THESIS

INVESTIGATING OVERTURNING HIGH SIDED VEHICLES THROUGH MODELING HIGH REYNOLDS NUMBER INCOMPRESSIBLE FLOW AROUND A RECTANGULAR CYLINDER NEAR A PLANE WALL BOUNDARY

Submitted by

Daniel K. Sanchez

Department of Civil & Environmental Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2023

Master's Committee:

Advisor: S. Karan Venayagamoorthy
Co-Advisor: Suren Chen

Daniel Olsen

ABSTRACT

INVESTIGATING OVERTURNING HIGH SIDED VEHICLES THROUGH MODELING HIGH REYNOLDS NUMBER INCOMPRESSIBLE FLOW AROUND A RECTANGULAR CYLINDER NEAR A PLANE WALL BOUNDARY

Safety on public roadways is of paramount importance to road users, road authorities, the local economy, and the general wellbeing of society. High sided vehicles (commonly known as semi-trucks in the United States (US) or lorries in the European Union (EU)) are used throughout the world for transporting freight, but they are susceptible to roll-over accidents due to high crosswind. The overturning of high sided vehicles is of concern during extreme wind events. In Boulder, Colorado, it is estimated that eight high wind events (with gusts greater than 75 mph) occur every year.

The research field of overturning high sided vehicles is young compared to other areas of knowledge since CJ Baker of the United Kingdom (UK) opened the research field in 1986. The traditional method applied for evaluating the likelihood of a high sided vehicle to overturn is to use the predetermined rolling moment coefficient ($C_{rolling}$) and translate the wind speed into a rolling moment. The resulting rolling moment can be compared to the restoring moment to determine the force required to overturn the high sided vehicle. This methodology requires that $C_{rolling}$ be accurate with respect to the high sided vehicle being analyzed. A recent study conglomerated many papers that have investigated $C_{rolling}$, showing wide variation in the expected $C_{rolling}$ for yaw angles between 45° and 90° (a direct crosswind). Through this thesis, it was discovered that some of the variation is due to the fact that $C_{rolling}$ is Reynolds number dependent.

In this thesis a comprehensive verification analysis and validation of a computational fluid dynamics (CFD) model was completed. Verification and validation are key components to performing a quality CFD analysis. When referring to verification, this traditionally implies a grid indepen-

dence study to ensure the CFD results are accurate with respect to the mesh sizing. However, this study explores why a comprehensive verification study is necessary to evaluate the influence of the flow domain size for high Reynolds number incompressible flow around a bluff body.

Additionally, it was found for flow around a rectangular cylinder near a plane wall boundary with a gap ratio of 0.407, that the drag coefficient (C_{drag}) is dependent on Reynolds number. This fundamental field was connected to the application of overturning high sided vehicles, with the assumption that a 2D rectangular cylinder could represent the trailer section of a high sided vehicle. It was found that traditional studies on overturning high sided vehicles assume the aerodynamic coefficients are Reynolds number independent, whereas the fundamental field shows that there is a Reynolds number dependence. It is apparent that additional work on determining $C_{rolling}$ is needed due to the Reynolds number dependency.

ACKNOWLEDGEMENTS

The completion of this thesis would not have been possible without the support of many people. First, I would like to thank my advisor, Dr. Subhas Karan Venayagamoorthy, for his mentorship, guidance, and support as I have worked through the graduate studies. Thank you for the flexibility that you provided due to starting the program during the middle of COVID. It has been an exceptional journey and I have learned a considerable amount along the way from Dr. Karan. Thank you. I also wish to thank:

- The Mountains-Plains Consortium for funding this work (project number MPC #644).
- Dr. Suren Chen and Dr. Daniel Olsen: thank you for being on my committee.
- My peers in the Environmental Fluid Mechanics Lab, both past and present: Harshit, Joe Pugh, Kasun Aluthwalage, Jessica Baker, Matthew Klema, and Joey Sinclair. Thank you for your readiness to discuss research, or just to enjoy time together in friendship. Thank you for being flexible and available, even with the COVID influence on my graduate studies journey. I appreciated the involvement and support each of you had in my journey.
- My family: mom and dad, Juliana and Truen, grandma and grandpa, the Campbells, and Matthew and Sydney for their support and love. I especially want to thank mom and dad - for your constant encouragement and prayers over me. Thank you for taking on the challenge of proofreading a new type of writing. I am thankful for your support at every turn.
- Jonathan Olds: thank you for your close friendship even though we are now separated by a 1,600-mile drive. I am thankful that the start of my masters coincided with your last semester of your CSU bachelors which allowed us to continue the in-person relationship. Thank you for the life-giving conversations and supporting me every step of the way.
- Jeremiah Corrado, my sole roommate who understood the dynamics of higher education and the completion of graduate studies: thank you for being available for questions, bouncing

ideas, and getting a general understanding of your perspective on different parts of the masters pursuit. To my other roommates: Josh Dooley, Robbie Wagner, Trent Kindvall, and Kaleb Johns, thank you for the camaraderie and fun we have had over the years.

- My community at Revive Church: thank you for your friendship, mentorship, and encouragement throughout this thesis. A special thanks to Micah Hinson for her timely encouragement when I was down, and especially when my girlfriend was not in Wheat Ridge.

- My mentors: Joe Bahr, Steve Rannells and Steve Gunning. Thank you for your encouragement, availability as wise sages, and flexibility in meeting as the masters ebbed and flowed.

- My colleagues at Jacobs: thank you for your support as I pursued my graduate studies and took a break from full time work. Thank you for continuing to allow me to come into the office even though I was not working on Jacobs' projects; I appreciated the workplace camaraderie, connections, a different environment to work, and the occasional free food. Rick Fell, thank you for your continued interest in the thesis; I am looking forward to continued mentorship and friendship with you.

- And the anchor of this thesis, my girlfriend Molly Chart: thank you for being the ultimate support. Even though you came into the picture during the last year of my program, you became my biggest cheerleader and support on the good days and not so good days. Thank you for your overwhelming dedication to helping me proofread 75% of this thesis. Since you do not come from an engineering background, it is incredibly impressive that you read through my many chapters and concretely understood what I was communicating, which enabled you to provide valuable feedback and edits. I am blown away by your dedication to me and looking forward to more of life with you, and I'll have more free time now!

And lastly but by no means the least, to my Lord and Savior Jesus Christ, thank you for Your gift of free salvation, Your grace, constant encouragement, and being the Prince of Peace. I am grateful for all You have done for me, and how You have placed me right where I am, to go and live and love like Jesus.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF CODES

LIST OF ACRONYMS

CAD	Computer Aided Design
CDOT	Colorado Department of Transportation
CFD	Computational Fluid Dynamics
CFL	Courant–Friedrichs–Lewy
CMV	Commercial Motor Vehicle
CoE	Cab over Engine
CPU	Central Processing Unit
EF	Enhanced Fujita scale
EU	European Union
FHWA	Federal Highway Administration
GAMG	Geometric agglomeration with an Algebraic Multi-Grid
GCM	Generic Conventional Model
GTS	Ground Transportation System
M-GCM	Modified-GCM
M-GTS	Modified-GTS
NWS	National Weather Service
OpenFOAM	Open-source Field Operation And Manipulation
PIMPLE	Combined solver of PISO & SIMPLE methods
PISO	Pressure Implicit with Splitting of Operators
RANS	Reynolds Averaged Navier-Stokes
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
SSH	Secure Shell
SST	Shear Stress Transport
UK	United Kingdom
URANS	Unsteady RANS
US	United States

LIST OF VARIABLES

A	(L^2)	Reference area
C_{drag}		Drag coefficient
C_{lift}		Lift coefficient
$C_{rolling}$		Rolling moment coefficient
C_{side}		Side coefficient
$Eu = \frac{P}{\rho V^2}$		Euler number
F_D	$(\frac{ML}{T^2})$	Drag force
F_L	$(\frac{ML}{T^2})$	Lift force
F_S	$(\frac{ML}{T^2})$	Side force
h	(L)	Reference height
H	(L)	Height
I		Turbulence intensity
k	$(\frac{L^2}{T^2})$	Turbulent kinetic energy
$k_{fs} = (3/2) * UI^2$	$(\frac{L^2}{T^2})$	Freestream turbulent kinetic energy (Wolf Dynamics, a, pg. 160)
L	(L)	Length
P or p	$(\frac{M}{LT^2})$	Pressure
$Re = \frac{\rho HV}{\mu}$		Reynolds number
$Re_{critical}$		Critical Reynolds number, above which the characteristics of the flow are Reynolds number independent
R_L	$(\frac{ML^2}{T^2})$	Overturning moment
$St = \frac{\omega H}{V}$		Strouhal number
$u_\tau = \sqrt{\frac{\tau_w}{\rho}}$	$(\frac{L}{T})$	Friction velocity (Pope, 2000, Eqn. 7.25)
V_r or V or U	$(\frac{L}{T})$	Wind velocity relative to vehicle
W	(L)	Width
$y^+ = \frac{u_\tau y}{\nu}$		y+ (Pope, 2000, Eqn. 7.28)

δ	(L)	Rectangular cylinder height
ΔT	(T)	Time step for OpenFOAM Simulation
μ	$(\frac{M}{LT})$	Dynamic viscosity
$\frac{\mu_t}{\mu}$		Viscosity ratio
ν_t	$(\frac{L^2}{T})$	Turbulent viscosity
ω	(T^{-1})	Shedding frequency AND Specific dissipation rate
$\omega_{fs} = \frac{\rho k}{\mu} * \frac{\mu_t}{\mu}^{-1}$	(T^{-1})	Freestream specific dissipation rate (Wolf Dynamics, a, pg. 160)
ψ		Relative wind direction (with respect to the vehicle length axis)
ρ	$(\frac{M}{L^3})$	Density of the wind
τ_w	$(\frac{M}{T^2 L})$	Wall shear stress (Pope, 2000, Eqn. 7.24)
$\frac{dU}{dt}$	$(\frac{L}{T^2})$	Velocity gradient

Chapter 1

Introduction

1.1 Background

Safety on public roadways is of paramount importance to road users, road authorities, the local economy, and the general wellbeing of society. High sided vehicles (commonly known as semi-trucks in the United States (US) or lorries in the European Union (EU)) are used throughout the world for transporting freight, but they are susceptible to roll-over accidents due to high crosswind as illustrated in the sequence of Figure 1.1. Roll-over accidents are also known as overturning accidents, blow-over accidents, and flipping accidents; this thesis will refer to these accidents as overturning high sided vehicles.

Figure 1.1: High sided vehicle overturning due to high crosswind. Images from The Weather Network.

The overturning of high sided vehicles is of concern during extreme wind events. In Boulder, Colorado, it is estimated that eight high wind events (with gusts greater than 75 mph) occur every year (NWS Boulder priv. comm.). The author estimates that the average person in the Colorado

Front Range region comes across 2 to 5 overturned high sided vehicles every year. Even though these events are relatively infrequent, they are of concern because the effects of an overturned high sided vehicle impact the safety of the driver, the transport of the goods and the general traffic flow since other vehicles tend to be less susceptible to overturning.

1.2 Thesis Impetus

The research field of overturning high sided vehicles is young compared to other areas of knowledge. 67 years ago, the National Interstate and Defense Highway Act of 1956 was passed resulting in the creation of the interstate highway system (The U.S. National Archives and Records Administration). Because of this, trucking became more common and was ingrained into American culture (Hale, 2015). Since that time, a significant amount of research has investigated the fuel efficiency of high sided vehicles with a focus on headwinds. This research typically involves some variations in the wind direction, but the wind direction is never investigated as a direct crosswind because the research focus is on fuel efficiency. However, in 1986, CJ Baker of the United Kingdom (UK) opened the research field into overturning high sided vehicles (Baker, 1986). Baker focused on using wind tunnel models, but the research performed to date by Baker and others has included full-size studies and computational fluid dynamics (CFD) models.

The traditional method applied for evaluating the likelihood of a high sided vehicle to overturn is to use the predetermined rolling moment coefficient ($C_{rolling}$) and translate the wind speed into a rolling moment. The resulting rolling moment can be compared to the restoring moment to determine the force required to overturn the high sided vehicle. This methodology requires that $C_{rolling}$ be accurate with respect to the high sided vehicle being analyzed. Baker and Soper (2022) published a summary paper that conglomerated many studies that have investigated $C_{rolling}$ since 1986. Figure 1 from Baker and Soper (2022) is reproduced below as Figure 1.2.

This figure shows normalized $C_{rolling}$ plotted against the relative wind angle of attack (commonly known as the yaw angle). $C_{rolling}$ is calculated about the leeward wheels, and the data in the plot is normalized by $C_{rolling}$ at a yaw angle of 30°. From this overview graph, it is evident

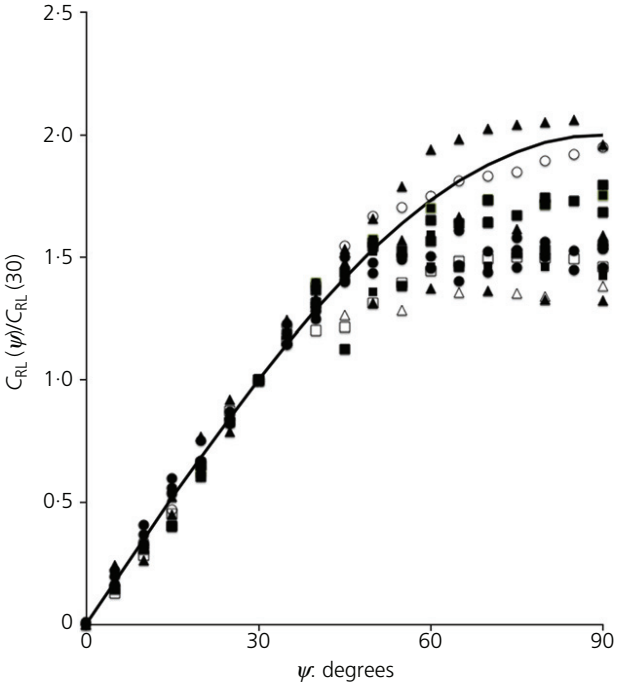


Figure 1.2: Reproduction from Baker and Soper (2022) – Leeward $C_{rolling}$ plotted against relative wind angle of attack, where 90° is a direct crosswind.

that there is a wide data spread for yaw angles between 45° and 90° (a direct crosswind). This increases the uncertainty that is present when calculating the likelihood of a vehicle to overturn.

Figure 1.2 is the impetus for this thesis, with a focus to understand why the data spread is present and significant. Some of this variation can be attributed to the various high sided vehicles used in the different analyses, however, not all the variation can be attributed to the different types of high sided vehicles. Through this thesis, it was discovered that $C_{rolling}$ is Reynolds number dependent.

1.3 Objectives

There are two main objectives with this thesis. First, a comprehensive verification analysis and validation of the CFD model will be completed. Verification and validation are key components to performing a quality CFD analysis. When referring to verification, this traditionally implies a grid independence study to ensure the CFD results are accurate with respect to the mesh sizing. However, this study explores why a comprehensive verification study is necessary to evaluate the

influence of the flow domain size for high Reynolds number incompressible flow around a bluff body.

In the second objective, it will be shown that flow around a rectangular cylinder near a plane wall boundary exhibits Reynolds number dependency. When comparing flow around a rectangular cylinder near a plane wall boundary to a rectangular cylinder in free flow, it becomes readily apparent that the flow vortical wake structures are asymmetric and symmetric, respectively. It was found for flow around a rectangular cylinder near a plane wall boundary with a gap ratio of 0.407, that the drag coefficient (C_{drag}) is dependent on Reynolds number. This fundamental field was connected to the application of overturning high sided vehicles, with the assumption that a 2D rectangular cylinder could represent the trailer section of a high sided vehicle. It was found that traditional studies on overturning high sided vehicles assume the aerodynamic coefficients are Reynolds number independent, whereas the fundamental field shows that there is a Reynolds number dependence.

1.4 New Contributions

This thesis has resulted in a few specific new contributions to the fields of CFD, fundamental fluid mechanics and the application field of overturning high sided vehicles:

1. For CFD models involving a bluff body in high Reynolds number incompressible flow, guidelines have been presented for a "comprehensive verification" analysis that goes beyond the typical verification method of a traditional grid independence study.

2. In the fundamental field of fluid mechanics, a gap in the literature has been filled for C_{drag} values for flow around a rectangular cylinder near a plane wall boundary, with a gap ratio of 0.407. These C_{drag} values have been provided for Reynolds numbers from 10^4 to 10^8.

3. In the application field of overturning high sided vehicles, it has been shown that Reynolds number dependency is a consideration that must be evaluated when looking at $C_{rolling}$.

1.5 Research Publications

The research presented in Chapter 4 has been submitted to the *Engineering Reports* journal (published by Wiley) under the title "A comprehensive verification study for computational modeling of high Reynolds number incompressible flow around a rectangular cylinder near a plane wall boundary". The research presented in Chapter 5 is being prepared for submission to the *Journal of Fluid Mechanics* under the title "Modeling high Reynolds number incompressible flow around a rectangular cylinder near a plane wall boundary, and an application to overturning high sided vehicles".

1.6 Organization of Work

This thesis has six additional chapters following this introduction. Chapter 2 presents the literature review associated with overturning high sided vehicles. Chapter 3 details the methods used in this thesis, including assumptions, the setup of the OpenFOAM CFD model, and the parametric framework used. Chapter 4 includes a paper that has been submitted on the comprehensive verification study conducted for this thesis. Chapter 5 includes a draft of the paper that is being prepared which focuses on the Reynolds number dependency for flow around a rectangular cylinder near a plane wall boundary and the resulting application to overturning high sided vehicles. Chapter 6 concludes this work and outlines future directions this research could go. Appendix A includes preliminary insights into a 3D analysis of overturning high sided vehicles. Appendix B contains general best practices that were learned throughout the course of this thesis. Appendix C includes details on the software versions used, details on the OpenFOAM and Linux workflow, and script details as a repository for the custom Linux, OpenFOAM and Matlab codes used throughout this thesis.

Chapter 2

Literature Review

The following review has been compiled to address the facets of literature, terminology, and research of high sided vehicles and flow around a rectangular cylinder near a plane wall boundary. Because high sided road vehicles are common throughout the world but are known by different names, the terminology will be clarified so that the language used in this paper is precise and understood. Following this, an overview of existing research on the overturning of high sided vehicles is discussed. Next, although the geographic focus of the majority of the existing research is in the EU, this study is funded in the US and as such, this thesis will focus on high sided vehicles found in the US; additional background differentiating the EU and US research and high sided vehicle designs is provided. Thereafter, the topic of aerodynamic coefficient dependency on Reynolds number is introduced, looking at the existing literature of high sided vehicles overturning and of a simplified 2D rectangular and circular cylinder, near a plane wall boundary and in free flow. To this end, a discussion on simplifying the trailer section of a high sided vehicle into a 2D rectangular cylinder near a plane wall boundary is presented.

2.1 Clarifying the Language Used to Address High Sided Articulated Road Vehicles (US: Tractor-Trailer Vehicles or Semi-Trucks, EU: Lorries)

In an international world, it can be difficult to coherently and consistently use the same terminology to describe similar or the same equipment. High sided vehicles, such as tractor-trailers, are no exception. Their terminology, the physical construction, and the geometry of tractor-trailers change depending on the geography and relevant regulations. However, their overall design is similar – reflecting a high sided articulated road vehicle that is prone to overturning in high cross-winds. Highlighting these differences is important because Baker (1986) spearheaded research into

overturning high sided vehicles in the UK. However, since the geographical focus of this thesis is on the US, it is appropriate to conduct the present study using vehicles that are found in the US.

To clarify the terminology, the following presents general yet imprecise terms that are used when referring to high sided vehicles in the US and the EU. Within the US, these terms include: 18-wheeler, semi-truck, high profile vehicle and tractor-trailer vehicle. Within the EU, these terms include: heavy goods vehicle, articulated lorry/artic, and non-articulated lorry/rigid lorry (Fleet Speak; Truck School Swindon). In this thesis, the term "high sided vehicle" will be used to refer to these aforementioned vehicles found in the US and EU. In the following section, a generic high sided vehicle model will be depicted and will be used as the basis for this thesis.

2.2 Background of Overturning of High Sided Vehicles

The field of overturning high sided vehicles is a newer field of research. In the US, the interstate system was created by the Interstate Defense Highway Act of 1956 (The U.S. National Archives and Records Administration) and commercial trucking grew to become a part of American culture during the 1950's & 1960's (Hale, 2015) To the author's knowledge, the earliest time when the overturning of high sided vehicles became of academic interest was in 1986 with CJ Baker's UK research focus (Baker, 1986). Baker acknowledged that there was a lack of data for ground vehicles, especially when compared to rail vehicles (Baker, 1991).

The field of knowledge has continued to expand over the decades. Notable papers include the following (Coleman and Baker, 1990, 1994; Cheli et al., 2011; Coleman, 1990) where a framework investigating the overall system force and moment balance on a high sided vehicle system is used. This involved experimentally determining aerodynamic coefficients which vary based on the yaw angle of the vehicle, as well as the specific vehicle of interest. Because of the importance of aerodynamic coefficients, many papers have been written to determine the coefficients for various geometries in various experimental settings including full-scale models, wind tunnel models and CFD models. Several of these are well summarized in Baker and Soper 2022. Most of the work involving high sided vehicles has revolved around experimental and CFD model studies to

investigate these aerodynamic coefficients of lift force, drag force, side-slip force, rolling moment, pitching moment and yawing moment.

2.3 Aerodynamic Coefficients

This CFD study will focus on investigating C_{drag}, side coefficient (C_{side}), lift coefficient (C_{lift}) and $C_{rolling}$. C_{drag}, C_{side} and C_{lift} are defined below (Blevins, 1984, pg. 342)(Sterling et al., 2010; Baker and Humphreys, 1996; Baker, 1991; Coleman and Baker, 1990, 1994; Baker, 1986):

$$C_{drag} = \frac{F_D}{0.5\rho A V_r^2} \tag{2.1}$$

$$C_{side} = \frac{F_S}{0.5\rho A V_r^2} \tag{2.2}$$

$$C_{lift} = \frac{F_L}{0.5\rho A V_r^2} \tag{2.3}$$

$C_{rolling}$ is defined below based on a revised definition from Baker and Soper (2022):

$$C_{rolling} = \frac{R_L}{0.5\rho A h V_r^2} \tag{2.4}$$

In equations 2.1 through 2.4, where F_D are the drag forces exerted on the entire body, and F_S and F_L are the forces exhibited on the reference area. R_L is the overturning moment on a vehicle about the leeward wheels, A is the reference area, h is the reference height, and V_r is the wind velocity relative to the vehicle (Baker and Soper, 2022).

2.4 US Focus on Overturning of High Sided Vehicles

The geometry of high sided vehicles vary throughout the world and within local geographies. The primary reason for the international physical differences of tractors and trailers is due to the influence of the EU and the US and the regulations that have been developed for each of these respective geographies. The following will describe the overarching regulations that govern each

of these countries, and then describe the distinct types of terminology that are used within these geographies.

The definition of the maximum vehicle length is the fundamental difference between US and EU regulations. As shown in Figure 2.1, in the US, the length of the trailer is restricted (US Department of Transportation) whereas in the EU, the total length of the tractor-trailer is restricted (UK Government).

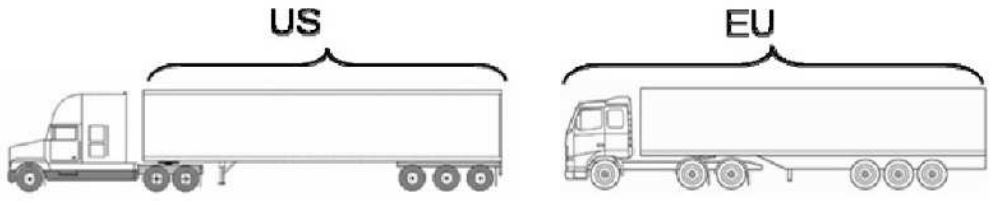


Figure 2.1: Schematic showing the regulatory differences between US and EU length regulations for high sided vehicles. Reproduced from Browand et al. (2009).

Additional regulations for the height and width of these vehicles also differ (Browand et al., 2009), but these differences are insignificant when compared to the length differences. The result is that a US tractor utilizes a conventional design where the engine sits in front of the cab. Whereas, in the EU the tractor is designed with the cab over the engine (CoE) to maximize the allowable space for the trailer. The CoE design is found outside of the EU in other countries such as Australia, Mexico, and other parts of Central America.

The previously referenced research that investigated the aerodynamic coefficients has primarily focused on CoE high sided vehicles. For example, looking at the compilation paper by Baker and Soper (2022), all the vehicle types are a CoE high sided vehicle. While research focused on CoE high sided vehicles provides a general understanding of the flow field surrounding high sided vehicles, it is not entirely applicable for the US conventional high sided vehicle. This knowledge gap must be filled with a focus on the overturning of conventional high sided vehicles.

Several types of simplified high sided vehicle models have been developed to standardize the analysis (Choi et al., 2014) as shown in Figure 2.2. The Ground Transportation System (GTS) and

Modified-GTS (M-GTS) (Croll et al., 1995) are similar in style to the EU's CoE high sided vehicle models. The Generic Conventional Model (GCM) and the Modified-GCM (M-GCM) (Storms et al., 2006) best resemble US Conventional style high sided vehicle models.

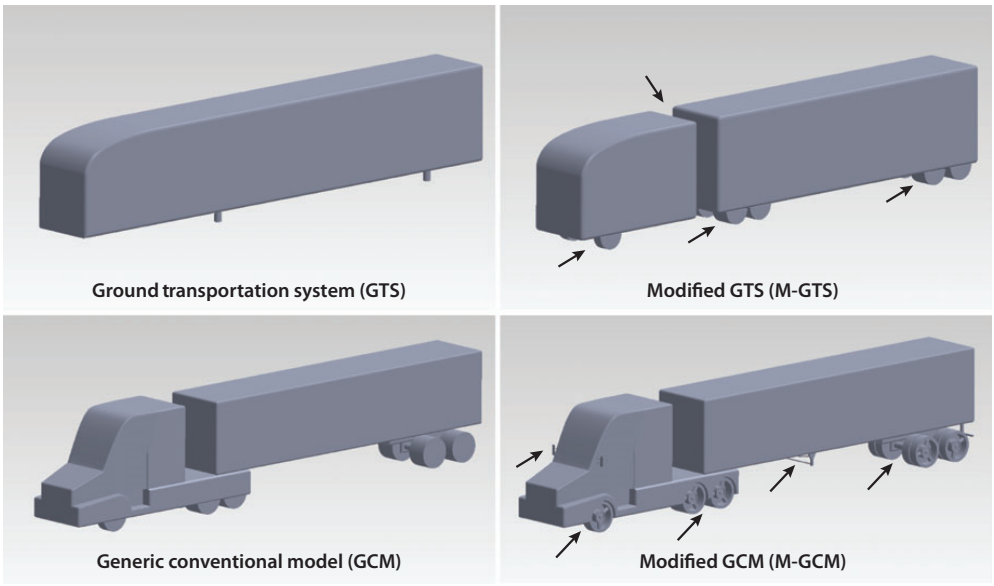


Figure 2.2: A schematic of different types of simplified high sided vehicle models. Reproduced from Choi et al. (2014).

For this thesis, a revised form of the GCM model (Figure 2.2, lower left corner) is utilized because the high level of geometric detail associated with the M-GCM model is not necessary for the purpose of this thesis. The GCM model is a Class 9 CMV (Commercial Motor Vehicle) because it has 5 axles (US Department of Transportation, Federal Highway Administration, Policy and Governmental Affairs, Office of Highway Policy Information). Based on measurements of a computer aided design (CAD) GCM model obtained from online (Macchesney, 2016), the trailer length was increased to 53 feet to reflect the typical trailer lengths found in the US. All other dimensions on this model were consistent with typical US high sided vehicle dimensions. This revised model used in this thesis is hereafter known as the 53-GCM model. More details about the revised 53-GCM geometry are discussed in Section 3.3.

2.5 Reynolds Number Dependency

As shown by Baker and Soper (2022) there is a wide variability in $C_{rolling}$ for yaw angles between 45° and 90° (a direct crosswind) as depicted in their compiled dataset of crosswind experiments for flat ground and unobstructed bridge scenarios (see Figure 1.2). These variations can largely be explained by the different types of vehicles that are being modeled and the Reynolds number used in the analysis. These studies were conducted with various Reynolds numbers, ranging from 8.5 x 10^4 to 1.25 x 10^5, which have been determined to be too low to be fully representative of realistic conditions in which overturning accidents may occur.

In Coleman and Baker (1994), a presentation of Reynolds number dependency is given. The conclusion was drawn that the effect of Reynolds number dependency is negligible, however, the Reynolds number only goes as high as 8.5 x 10^4. Other papers are similar in their low Reynolds number bounds: Coleman and Baker (1990) at 8.5 x 10^4, Baker (1991) up to 2.4 x 10^5, and Sterling et al. (2010) up to 1.25 x 10^5. When the Reynolds number is below a critical value, the flow characteristics exhibit Reynolds number dependency (Gerhart et al., 2016). In Baker (1991) and Coleman and Baker (1990), the Reynolds numbers are above a critical value and in Baker and Gawthorpe (1983) an argument is stated that the flow characteristics are independent of the Reynolds number. This argument is predicated upon an analysis of high Reynolds number crosswind with railroad trains, and this analysis is extrapolated to high sided vehicles. It is unknown if this extrapolation is accurate because there is a difference in the gap ratio for high sided vehicles compared to railroad trains.

The Reynolds number independence argument is corroborated in Gerhart et al. (2016) and Yang et al. (2021) where it is shown that C_{drag} asymptotically approaches a value as the Reynolds number increases, for both a circular and square cylinder. However, it must be noted that the asymptotic behavior of C_{drag} is for a cylinder in free flow, whereas an overturning high sided vehicle is in the presence of a plane wall boundary.

2.6 Trailer Section Simplification into 2D Rectangular Cylinder

For the purpose of comparison with the literature and simplification of a complex problem, the 3D problem of overturning high sided vehicles is simplified into flow around a rectangular cylinder near a plane wall boundary, in which the rectangular cylinder represents the trailer section of the high sided vehicle. It was determined that a simplification of this problem would be appropriate to expedite the analysis and focus on the Reynolds number dependency effects. Since the trailer section of a high sided vehicle is the most prominent, it is appropriate to conduct a 2D analysis using a cross section of the trailer section. This resulted in configuring a model with a 2D rectangular cylinder near a plane wall boundary.

There is a broad body of research for cylinders of various shapes in free flow, however, the scope of research associated with a cylinder near a plane wall boundary is more limited, especially for a rectangular cylinder near a plane wall boundary. For a circular cylinder near a plane wall boundary, most of the research has not ventured into the realm of Reynolds number dependency and has been limited to investigating the effect of the gap ratio on the characteristics of the flow (Zdravkovich, 1985; Bearman and Zdravkovich, 1978; Roshko et al., 1975). For rectangular cylinders near a plane wall boundary, much of the research has been restricted to Reynolds numbers in the range of laminar and transitioning to turbulent with a range from 50 to 4 x 10^5 (Bhattacharyya and Maiti, 2004; Cheng et al., 2007; Mahir, 2009; Yang et al., 2022; Forouzi Feshalami et al., 2022). There is a noticeable gap in the research of this field for higher Reynolds numbers. This paves the way for research investigating high Reynolds number incompressible flow with a rectangular cylinder near a plane wall boundary.

2.7 Literature Review Summary

Because of the influence of a plane wall boundary and previous Reynolds number dependency validation relying on low Reynolds numbers, a new question surfaces: Is flow around a high sided

vehicle, or more generally around a rectangular cylinder near a plane wall boundary, subject to Reynolds number dependency? It will be imperative to evaluate the Reynolds numbers up to realistic high wind speeds that could cause overturning accidents.

Chapter 3

Methods

With the research focus established, it is important to consider the means and methods used to implement this research. Some of the details provided below have been introduced in the remaining chapters of this thesis, however, this section is provided to expound on these preliminary introductions. This section discusses simplifications and assumptions used in analyzing the Reynolds number dependency associated with flow against high sided vehicles and a rectangular cylinder near a plane wall boundary, the selected Reynolds number bounds, additional details on developing the 53-GCM geometry, and the general OpenFOAM model setup include the initial condition and boundary condition configurations.

3.1 Simplifications and Assumptions

Several assumptions were made to simplify the dynamics of overturning high sided vehicles to allow the study to isolate the facet of aerodynamic coefficient dependency on Reynolds number. The dynamics of high sided vehicles overturning has many factors that influence the overall system. These include the wind direction, local topography (such as wind breaks, tree canopy, open country or mountain valleys that channel flow or significantly obstruct the flow), various bridge configurations (such as in mountains in channelized flow, over the ocean with unobstructed flow, or over open low lying land), other local structures, roadway configurations (such as multiple vehicles in a row, multiple lanes of traffic, or the camber of the road), and the weather conditions. All these factors are significantly simplified for the purpose of this present study to allow the specific effects of the wind velocity in a direct crosswind to be realized on the overall system and aerodynamic coefficients. These factors are assumed as follows: no land topology changes; no other structures or roadway configurations are included in the system, only the high sided vehicle is modeled; the high sided vehicle is modeled as a rigid body, neglecting the dynamics of the suspension system; the interaction of the driver with the high sided vehicle is neglected; the ground surface is assumed

to be smooth, neglecting the ground surface interaction with trees, grass, and manmade structures; the wind direction is specified as a direct crosswind; no adverse weather conditions are modeled.

3.2 Reynolds Number Bounds

As previously stated in the literature review, the Reynolds numbers typically used in prior analyses were found to be inadequate, ranging from 8.5 x 10^4 to 1.25 x 10^5 which correlate to a full-size wind velocity of 0.981 mph to 1.442 mph. Based on the following analysis to determine the Reynolds number bounds, this previously evaluated range was found to be deficient by a factor of 10 to 100. The desired Reynolds number bounds were found through the following procedure:

1. Evaluating typical wind speeds seen in Wyoming and Colorado, US by various agencies, including the National Weather Service (NWS) (NWS Boulder; NWS Cheyenne) and Colorado Department of Transportation (CDOT). See Table 3.1.

2. Corroborating item #1 with recent high wind events seen in Colorado's Front Range region (Denver metro and southern Colorado areas).

3. Establishing a desired Reynolds number bound based on items #1 and #2.

4. Expanding the study bound of item #3 to generate a broader dataset that encompasses the lower Reynolds numbers presented in previous literature.

Wind speeds are typically reported as sustained winds at an average number, with gusts up to a given maximum speed. Table 3.1 highlights the sustained winds and gusts relevant to different agencies that result in the specified restriction.

This table highlights that wind becomes a concern to the NWS Boulder and NWS Cheyenne when sustained winds are 40 mph. In Colorado, there have been a handful of recent wind events that have been well documented, namely the December 2021 wind events in northwest metro Denver and across the southern Colorado region (NWS Boulder, 2021; NWS Pueblo, 2021) and the June 2023 tornado event in Douglas County, Colorado (NWS Boulder, 2023). The December

Table 3.1: Sustained Winds and Gusts of Interest per various Agencies.

Agency	Restriction Type	Wind Speed at which Restriction is Enacted for	
		Sustained Winds	Gusts
NWS Boulder and NWS Cheyenne (priv. comm.)	High Wind Warning, for the Mountains	50 mph	75 mph or more
NWS Boulder and NWS Cheyenne (priv. comm.)	High Wind Warning, for the Plains	40 mph for 1 hour or more	58 mph or more
CDOT (Beedie, 2021)	High Wind Restriction		60 mph or more

2021 wind event reports show maximum wind gusts up to 115 mph, with winds over 100 mph reported in several locations. The June 2023 Douglas County tornado was rated at EF1 (Enhanced Fujita scale), with estimated peak winds of 105 mph. This provides a realistic upper boundary for expected windspeeds in the Colorado Front Range.

Based on these agency criteria and local observations, a wind speed range of approximately 22 mph to approximately 130 mph is proposed for this study. This range contains the minimum value of 40 mph and an upper bound of 130 mph is provided to allow the data to be more broadly studied beyond the measured maximum values of 115 mph. The Reynolds number range for this core dataset is 1.94×10^6 to 1.12×10^7. In order to expand this dataset to cover the Reynold numbers presented in the experimental literature, the lower bound must be extended below 8.5×10^4 to a Reynolds number of 10^4; this correlates to a full-size velocity of 0.115 mph. Additionally, it was determined to set the upper Reynolds number bound to a value higher than 1.12×10^7; 10^8 was selected as an arbitrary value that is one order of magnitude larger than 1.12×10^7. 10^8 correlates to a full-size velocity of 1,154 mph.

Using this bounding guidance, 17 Reynolds numbers were tested and are shown in Table 3.2. This table shows the full-size velocity in mph and m/s, the Reynolds number, model velocity in m/s and Model Mach number. The model is 1:8 scale geometry, using water as the model fluid with a kinematic viscosity of 1.004×10^{-6} m^2/s. The model characteristic length is 0.365 m, which is the height of the 53-GCM model trailer. Since the model fluid was specified as water, this maintained the Mach number within the recommended bounds of less than 0.3 (Gerhart et al., 2016).

Table 3.2: Parameters used for model.

Velocity (Full Size)	Velocity (Full Size)	Reynolds Number	Velocity (Model Scale)	Model Mach Number
m/s	mph		m/s	
0.052	0.115	1.00E+04	0.028	0.000
0.129	0.288	2.50E+04	0.069	0.000
0.258	0.577	5.00E+04	0.138	0.000
0.387	0.865	7.50E+04	0.206	0.000
0.516	1.154	1.00E+05	0.275	0.000
1.290	2.885	2.50E+05	0.688	0.000
2.579	5.770	5.00E+05	1.376	0.001
3.869	8.655	7.50E+05	2.063	0.001
5.159	11.540	1.00E+06	2.751	0.002
10.000	22.369	1.94E+06	5.333	0.004
18.000	40.265	3.49E+06	9.600	0.006
26.000	58.160	5.04E+06	13.867	0.009
34.000	76.056	6.59E+06	18.133	0.012
42.000	93.951	8.14E+06	22.400	0.015
50.000	111.847	9.69E+06	26.667	0.018
58.000	129.742	1.12E+07	30.933	0.021
515.864	1153.954	1.00E+08	275.127	0.186

3.3 3D Geometry: Developing the 53-GCM Model

As discussed in the literature review (Chapter 2), the base model for the 53-GCM was obtained online (Macchesney, 2016). The sketch in Figure 3.1 is the basis for the CAD GCM model (Storms et al., 2006), set to a scale of 1/8. Table 3.3 compares the CAD GCM model obtained online and the Federal Highway Administration (FHWA) and state specific regulations, as well as more specifics on changes made to the CAD GCM model to facilitate this study. The resulting 53-GCM model more accurately reflects the size of trailers that are used in the US. This geometry resizing only applies to the 3D model; the 2D model is discussed in the following section.

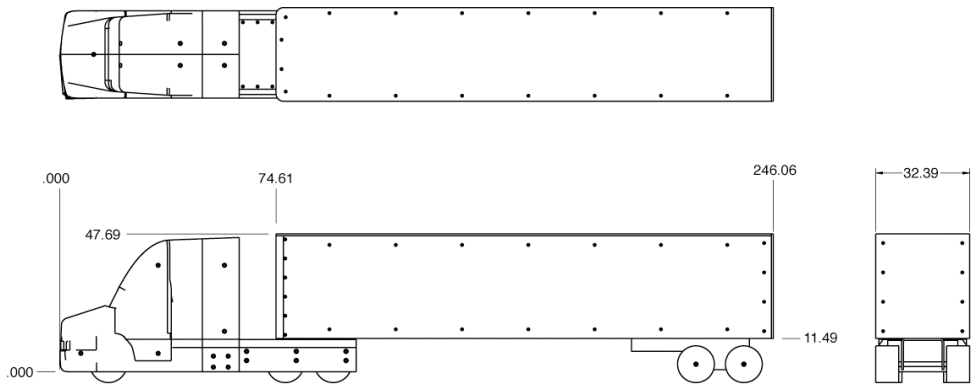


Figure 3.1: Basis for GCM Model, measurements in cm (Originally published in Storms et al. (2006)).

Table 3.3: Changes made to the CAD GCM model based on typical US regulations (US Department of Transportation).

	CAD GCM	US Regulations	Changes to CAD GCM to create 53-GCM model
Trailer length	44.9 ft	48 ft – 53 ft. This is a typical length range. FHWA and state specific regulations allow some variance, but most trailers are 53 ft in length.	Increase length of trailer to 53 ft because 48 ft is minimum and the existing 44.9 ft is too short.
Total Tractor-Trailer assembly height	13.46 ft	FHWA allows states to limit the max height which is typically 13.5 ft to 14 ft.	None.
Trailer width	8.5 ft	FHWA regulates the max width to 8.5 ft.	None.
Tractor CAD Geometry	N/A	N/A	Removed 2 unnatural and sharp corners that will be an easy location for flow separation to occur, as well as difficult for the meshing software to pick up and accurately mesh.

3.4 2D Geometry: Rectangular Cylinder near a Plane Wall Boundary

To simplify the analysis, it was determined that a 2D model would be adequate to focus the study on analyzing the Reynolds number dependency. The rectangular cylinder was developed based on a simple cross section of the 53-GCM trailer section, resulting in a suspended rectangular cylinder with dimensions that resembled a typical trailer. The dimensions of this rectangular cylinder are presented in Chapters 4 and 5.

3.5 OpenFOAM Setup

Since OpenFOAM was selected as the CFD software, this section details initial conditions, boundary conditions, numerical schemes, system settings, and function objects used in configuring OpenFOAM. A balanced approach to this section has been taken, with portions being detailed and using OpenFOAM specific language, and other sections using generic terminology that is applicable to general CFD applications. Appendix C contains minute details on the OpenFOAM setup, including specific software versions used, details on the typical workflow used in creating, running and post processing simulations, as well as a comprehensive script repository.

3.5.1 Initial Conditions and Boundary Conditions for 2D Model

Chapters 4 and 5 provide a general overview of the initial conditions and boundary conditions used for the 2D model. This section provides an explicit use of OpenFOAM terminology. The $k-\omega\ SST$ model was used to close the Navier-Stokes equation because it is a y^+ insensitive model (Wolf Dynamics, a, pg. 154), which is beneficial since the speed of the flow varies significantly in the model (Wolf Dynamics, a, pg. 117).

The following considerations were used at each patch surface for the respective initial conditions and boundary conditions. The ground surface and surface of the rectangular cylinder are the model's walls; on these surfaces, the $k-\omega\ SST$ model was implemented with a no-slip velocity boundary condition, a zero-gradient boundary condition for pressure and turbulent kinetic energy,

and a log-law blending function for the specific dissipation rate. At the atmospheric boundary, a free-slip condition was allowed. At the outlet, the flow was assigned a zero-gradient condition. At the flow inlet, a zero-pressure gradient condition was specified with a fixed value condition specified for the velocity, specific dissipation rate, turbulent kinetic energy and turbulent viscosity. The turbulence intensity was set at 1%. Two different inlet flow profiles were used depending on the model. 1) For the validation model and to generate developed flow profiles, uniform flow profiles were used. 2) Once the vertical domain height was established, developed flow profiles were used for the verification models.

Table 3.4 details these initial conditions and boundary conditions with OpenFOAM methodology. Since two different types of inlet profiles were used, this table includes a differentiation for the different settings used for the uniform flow profile and developed flow profile. This table is laid out using Linux notation, where a variable is defined by "variable=1", and this variable can be referenced in subsequent rows by "$variable". For example, in Table 3.4, $k_{freestream}$ is marked as $k_{fs} = (3/2) * UI^2$, and later is referenced as in "uniform $\$k_{fs}$". For the sake of visual clarity, a blank cell in this table correlates to a value that is not applicable.

Table 3.4: Detailed description in OpenFOAM terms of parameters applied to each boundary and wall to facilitate the implementation of the $k-\omega\ SST$ model (Wolf Dynamics, a, pg. 160, 161).

OpenFOAM Patch	Parameter types	Velocity (U)	Pressure (p)	Specific Dissipation Rate (ω)	Turbulent Kinetic Energy (k)	Turbulent Viscosity (ν_t)
Freestream	value			$\omega_{fs} = \frac{\rho k}{\mu} * \frac{\mu_t}{\mu}^{-1}$	$k_{fs} = (3/2) * UI^2$	
internalField		uniform ($U 0 0)	uniform 0	uniform ω_{fs}	uniform k_{fs}	uniform 0
Inlet: uniform flow	type intensity value	uniform fixedValue $internalField	zeroGradient	uniform fixedValue $internalField	turbulentIntensity-KineticEnergyInlet 0.01 uniform 1	calculated uniform 0
Inlet: developed flow	type	Non-uniform fixedValue	zeroGradient	Non-uniform fixedValue	Non-uniform fixedValue	Non-uniform fixedValue
Outlet	type inletValue value	pressureInlet-OutletVelocity uniform (0 0 0)	totalPressure p0 = uniform 0	inletOutlet uniform ω_{fs} uniform ω_{fs}	inletOutlet uniform k_{fs} uniform k_{fs}	calculated uniform 0
Rectangular cylinder & ground surface	type value	noSlip	zeroGradient	omega-WallFunction uniform ω_{fs}	kqR-WallFunction uniform 1.00E-10	nutU-WallFunction uniform 0
Atmosphere	type value	slip	slip	slip	slip	calculated uniform 0

3.5.2 Numerical Schemes

fvSchemes. Time Discretization

The Crank-Nicolson method was implemented because it can be a second order accurate scheme. Additionally, based on recommendations found through an OpenFOAM training course, it was suggested to use the Crank-Nicolson method with a blending factor of 0.7 which results in an almost second order accurate scheme (Wolf Dynamics, b, pg. 59)).

fvSchemes. Spatial Discretization optimized for $k-\omega\ SST$

Because the $k-\omega\ SST$ model was selected, a general discretization scheme optimized for the $k-\omega\ SST$ model was preferred. This discretization scheme was found through recommendations from an OpenFOAM training course (Wolf Dynamics, a, pg. 324). It was suggested to use the following first order accurate methods for the convective terms because turbulence is a diffusive process (Kundu et al., 2016); this is shown in Code Listing 3.1.

Code Listing 3.1: divSchemes methods, for convective terms. Reproduced from Wolf Dynamics (a), pg. 324.

```
divSchemes
{
    div(phi,U) Gauss linearUpwind default;
    div(phi,k) Gauss upwind;
    div(phi,omega) Gauss upwind;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}
```

For the gradient terms, a bounded gradient discretization was selected as recommended by the OpenFOAM training course because this simulation is URANS (Unsteady Reynolds Averaged Navier-Stokes) (Wolf Dynamics, a, pg. 325). Code Listing 3.2 shows the methods that were implemented.

Code Listing 3.2: gradSchemes methods, for gradient terms. Reproduced from Wolf Dynamics (a), pg. 325.

```
gradSchemes
{
    default cellLimited Gauss linear 1;
    grad(k) cellLimited Gauss linear 1;
    grad(omega) cellLimited Gauss linear 1;
}
```

In order to facilitate the implementation of the $k-\omega\ SST$ model, a method for wallDist had to be specified to allow the calculation of the approximate distance from all cells and boundaries to the nearest patch. The meshWave method was implemented based on recommendations from the OpenFOAM training course (Wolf Dynamics, a, pg. 326) as shown in Code Listing 3.3.

Code Listing 3.3: wallDist method, for calculation of cell distance from nearest wall. Reproduced from Wolf Dynamics (a), pg. 326.

```
wallDist
{
    method meshWave;
}
```

fvSchemes. Non-Orthogonality Requirements

The laplacianSchemes and snGradSchemes (surface normal gradient) are implemented based on non-orthogonality requirements provided in the OpenFOAM training course (Wolf Dynamics, b, pg. 38). For the Laplacian scheme, a simple linear central differencing scheme is implemented as it is second order accurate. Both the Laplacian and surface normal gradient schemes use a limited blending factor of 1 due to the low non-orthogonality of the rectangular cylinder near a plane wall boundary mesh, where the non-orthogonality was typically less than 50° (Greenshields, 2022). The Laplacian scheme is shown in Code Listing 3.4 and the surface normal gradient is in Code Listing 3.5.

Code Listing 3.4: laplacianSchemes method, for shear stress divergence. Reproduced from Wolf Dynamics (b), pg. 38.

```
laplacianSchemes
{
    default Gauss linear limited 1.0;
}
```

Code Listing 3.5: snGradSchemes method, for gradient component normal to a cell face. Reproduced from Wolf Dynamics (b), pg. 38.

```
snGradSchemes
{
    default limited 1.0;
}
```

fvSolution. PISO Method

The PISO (Pressure Implicit with Splitting of Operators) method uses an iterative method to solve the Navier-Stokes equations. This requires a general tolerance for the solution, before converging on a final tolerance. The solver used the GAMG method (Geometric agglomeration with an Algebraic Multi-Grid), and the settings were specified based on recommendations from the OpenFOAM training course (Wolf Dynamics, b, pg. 52-53) and as shown on Code Listing 3.6. The PIMPLE solver (combined solver of PISO and SIMPLE methods [Semi-Implicit Method for Pressure-Linked Equations] methods) was used because it has a dynamic ΔT which is restricted by the maximum CFL (Courant–Friedrichs–Lewy) number. However, since the number of outer correctors was set to 1, this sets the solver equal to the PISO method with the added functionality of a dynamic ΔT.

Code Listing 3.6: PISO Settings. Reproduced from Wolf Dynamics (b), pg. 52-53.

```
solvers
{
    p
    {
        solver                  GAMG;
        tolerance               1e-4;
        relTol                  0.01;
        smoother                GaussSeidel;
        nPreSweeps              0;
        nPostSweeps             2;
        cacheAgglomeration      on;
        agglomerator            faceAreaPair;
        nCellsInCoarsestLevel   1000;
        mergeLevels             1;
    }

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

    "(U|k|omega)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-05;
        relTol          0.1;
    }

    "(U|k|omega)Final"
    {
        $U;
        relTol          0;
    }
}

PIMPLE
{
    momentumPredictor           yes;
    nOuterCorrectors            1;
    nCorrectors                 3;
    nNonOrthogonalCorrectors    1;
    turbOnFinalIterOnly         false;
}
```

3.5.3 Control Dictionary Settings and Function Objects

The system control dictionary file was specified with the typical parameters needed for an OpenFOAM simulation. The two parameters of interest that were necessary to tweak were the end time and ΔT. The PIMPLE solver was used which invokes a dynamic ΔT, however, the initial ΔT must be prescribed before the solver dynamically chooses the next ΔT. ΔT for all simulations was set to 10^{-10} which was between 100 to 10,000 orders of magnitude lower than the resulting ΔT that the simulation oscillated around. This low ΔT was chosen because it was always below the final average ΔT and allowed the simulation to approach convergence by itself. In the few cases where the initialization ΔT was too high, the simulation never converged.

The end time was specified in different manners depending on the simulation type. There were four types of simulations that were used throughout this thesis. 1) Initial validation of the CFD model was completed with the rectangular cylinder in free flow, 2) Verification of the CFD model and final results were obtained with flow around a rectangular cylinder near a plane wall boundary, 3) Flow development simulations were run for generating the developed flow inlet condition for #2, and 4) Test 3D simulations of the 53-GCM were completed. Various function objects were used to facilitate the collection of data for each simulation. For each type of simulation, Table 3.5 outlines the procedures used to calculate the end time and Table 3.6 details the function objects used.

Table 3.5: A description of the procedures used to calculate the end time for each type of simulation.

Simulation Type	Procedure to Determine End Time	Procedure to Determine Beginning of Sample Time
Rectangular cylinder in free flow	It was determined that the simulation time needed to be long enough to allow the flow to pass from the inlet to the outlet, multiplied by 10.	The beginning of the sample timeframe took the end time divided by 2.
Flow around a rectangular cylinder near a plane wall boundary	These simulations used a developed flow profile, which resulted in the ground surface velocity taking longer to propagate through the domain compared to the relatively uniform velocity above the cylinder. The simulation end time was computed using the minimum inlet velocity located at the cell immediately above the ground surface. This minimum velocity ranged between 20% - 48% of the uniform model velocity. The end time was calculated by multiplying the time required for the minimum inlet velocity to traverse the horizontal domain by approximately 8.8. For some flow velocities, the value was increased up to 10.087 based on the individual requirements to end on a visually pleasing number.	The beginning of the sample timeframe started at two-thirds of the end time value.
Flow development	The flow development simulations continued until the simulation progressed such that the developed flow was at a distance of 0.6m above the ground surface. The flow achieved the developed state between 40-75% of the time required for the flow to propagate across the 100m horizontal domain length.	N/A
53-GCM	Due to the high computational resources required for the 3D simulation, the end time was set to an arbitrary value to allow any amount of time to be completed.	N/A

Table 3.6: A description of the function objects for data collection for each type of simulation.

Simulation Type	Data Collection
All Simulations	All simulations gathered y^+ data on all surfaces.
Flow around a rectangular cylinder in either flow condition	The vorticity function was implemented in these simulations, to allow the vortical structures in the flow to be observed. The field average function was included to allow a RANS (Reynolds Averaged Navier-Stokes) type result to be viewed. Additionally, the forces and aerodynamic coefficients were collected on each individual face of the cylinder geometry and on the collective cylinder. This allowed the automatically computed aerodynamic coefficients to be back-checked by the force values.
Flow development	Data was uniformly collected along a vertical line to generate data to compute the non-dimensional flow characteristics and plot them against the log-law Kundu2016. The data collected included the τ_w, velocity, and $\frac{dU}{dt}$. Approximately 2010 data points were collected at every write time interval.
53-GCM	The vorticity function was implemented in these simulations, to allow the vortical structures in the flow to be observed. The field average function was included to allow a RANS type result to be viewed. Additionally, the forces and aerodynamic coefficients were collected, as outlined above in the data collection description for flow around a rectangular cylinder.

Chapter 4

A comprehensive verification study for computational modeling of high Reynolds number incompressible flow around a rectangular cylinder near a plane wall boundary[1]

4.1 Introduction

This study explores the traditional concept of computational fluid dynamics (CFD) model verification with additional considerations as it pertains to 2D incompressible high Reynolds number flow (order 10^5 to 10^7) around a bluff body. In CFD studies, it is best practice to perform a model verification test, typically a grid independence test, to verify that the CFD results are accurate with respect to the mesh sizing (Versteeg and Malalasekera, 2007). This study shows model verification examined solely through grid independence is not adequate. A comprehensive verification study is necessary in high Reynolds number incompressible flows, especially for flows around a bluff body; additional care must be taken when constructing the computational fluid domain and evaluating the CFD results.

The layout of this paper is as follows. Section 4.2 introduces the study motivation, parametric framework, numerical methodology and the parameters used in OpenFOAM, an opensource CFD platform (Weller et al., 1998). Section 4.3 details the validation analysis used to support the use of OpenFOAM. Section 4.4 outlines the comprehensive verification study domain parameters and geometry used. Section 4.5 contains the results of the comprehensive verification study, with

[1]The research presented in this chapter has been submitted to the *Engineering Reports* journal (published by Wiley) under the same title. Background information and literature relevant to this chapter are presented again so the chapter may be read as a stand-alone work.

Section 4.6 discussing these results. Section 4.7 discusses recommendations to accelerate future comprehensive verification studies and concludes this study.

4.2 Model Parametric Framework & Numerical Methodology

This study is based on work (Sanchez, 2023) which was inspired by investigating high sided vehicles (known in different parts of the world as, tractor-trailer vehicles, semi-trucks or articulated lorries) overturning due to high crosswinds. The characteristic length scale is the height of the rectangular cylinder (0.365m) at model scale. Verification of the CFD model was completed at the Reynolds numbers of 1.94 x 10^6 and 1.12 x 10^7. Validation of the CFD model used a Reynolds number of 100,000.

4.2.1 Governing Equations

The governing equations for this parametric study are the Navier-Stokes equations. Assuming incompressible flow ($\nabla \cdot \mathbf{u} = 0$), the simplified Navier-Stokes equations are as follows (Kundu et al., 2016):

$$\rho \left(\frac{\partial u_j}{\partial t} + u_i \frac{\partial u_j}{\partial x_i} \right) = -\frac{\partial P}{\partial x_j} + \rho g_j + \mu \frac{\partial^2 u_j}{\partial x_i^2} \tag{4.1}$$

The following section discusses the discretization and solution methodology.

4.2.2 CFD Solver – OpenFOAM

OpenFOAM was selected as the CFD solver because it is customizable. These configurable and programming capabilities were deemed beneficial to conduct a separate parametric study (Sanchez, 2023). OpenFOAM has been investigated and validated for incompressible flows around a bluff body using the finite volume method (Robertson et al., 2015). This study utilized OpenFOAM version 9 and sought to implement best practices in the OpenFOAM environment. Because the Reynolds number is high, an unsteady flow field was expected. As a result, a URANS (Unsteady Reynolds Averaged Navier-Stokes) simulation was desirable and the PISO (Pressure Im-

plicit with Splitting of Operators) method (Issa, 1986) was implemented to facilitate URANS. To close the Navier-Stokes equation, the $k-\omega\ SST$ model was utilized because it is a y^+ insensitive model, which is necessary when performing traditional grid refinement studies (Menter et al., 2003) (where y^+ is a nondimensional measure of wall distance units; $y^+ = \frac{u_\tau y}{\nu}$ (Pope, 2000)). The PISO method was selected to solve the Navier-Stokes equations because of the transient nature of the unsteady simulations (Greenshields, 2021). Because turbulence is a diffusive process (Kundu et al., 2016), a first order upwind spatial scheme was implemented for the turbulence variables. The time discretization used the Crank-Nicolson method (Crank and Nicolson, 1947) with the blending factor set to 0.7, resulting in an approximately second order accurate time scheme.

The following considerations were used at each patch surface for the respective initial conditions and boundary conditions. The model walls are the ground surface and surface of the rectangular cylinder; on these surfaces, the $k-\omega\ SST$ model was implemented with a no-slip velocity boundary condition, a zero-gradient boundary condition for pressure and turbulent kinetic energy, and a log-law blending function for the specific dissipation rate. At the atmospheric boundary, a free-slip condition was allowed. At the outlet, the flow was assigned a zero-gradient condition. At the flow inlet, a zero-pressure gradient condition was specified, with a fixed value condition specified for the velocity, specific dissipation rate, turbulent kinetic energy and turbulent viscosity. The turbulence intensity was set at 1%. Two different inlet flow profiles were used depending on the model. 1) For the validation model and to generate developed flow profiles, uniform flow profiles were used. 2) Once the vertical domain height was established, developed flow profiles were used for the verification models. The developed flow profile is discussed in detail below.

4.3 Validation Geometry and Analysis

First, a validation study certified the use of OpenFOAM and the model setup. A schematic for the flow domain used in OpenFOAM is shown in Figure 4.1. A cell size of 0.01m was used. The flow domain was sized as follows: (a) vertical domain height of 12m, and total horizontal domain length of 17m with (b) 10m length upstream from the cylinder centroid and (c) 7m length

downstream from the cylinder centroid. A rectangular cylinder, simulating the trailer section of a high sided vehicle at 1/8 scale, was used with (d) a cylinder height of 0.365m and (e) a cylinder width of 0.324m.

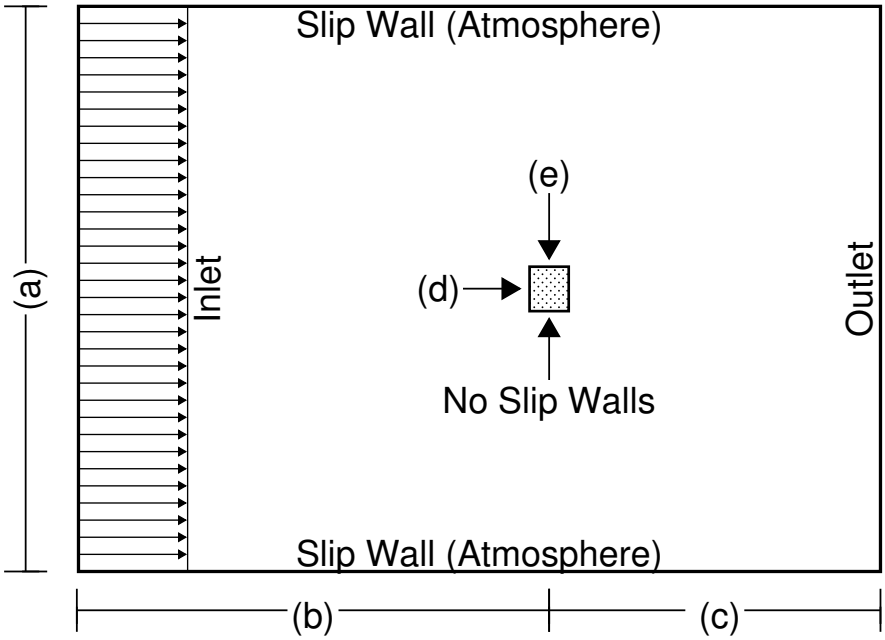


Figure 4.1: Validation Geometry – rectangular cylinder in free flow.

This validation study used an interpolated drag coefficient (C_{drag}) based on an experimental C_{drag} dataset utilizing rectangular cylinders with varying width to height ratios (Blevins, 1984; Courchesne and Laneville, 1979; Hoerner, 1965). Figure 4.2 shows the data (Blevins, 1984) with an interpolated target C_{drag} of 2.256 highlighted by the triangle.

Table 4.1 shows the measured C_{drag} from the OpenFOAM model, which is dependent on the number of inflation layers used, and the percent error relative to the interpolated C_{drag}. The error ranges from -0.742% to 6.245%, but is within a reasonable limit for all cases.

Based on best practices for developing a mesh with appropriate y^+ bounds, 0 inflation layers is ideal for the validation model. For the verification geometry, 6 inflation layers were selected

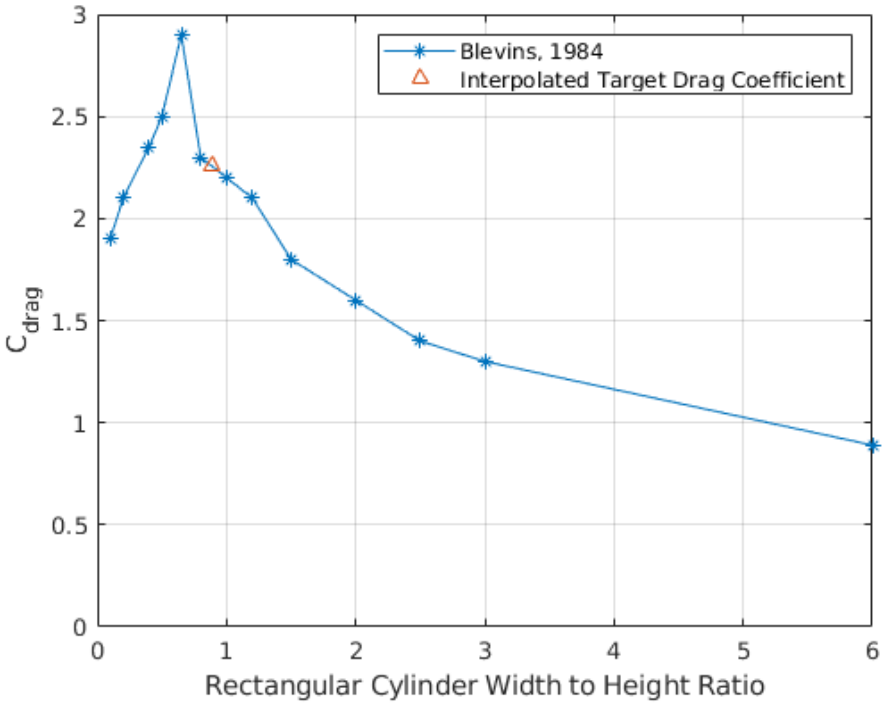


Figure 4.2: C_{drag} for varying rectangular cylinder width to height ratio, Re = 100,000.

Table 4.1: C_{drag} for free flow analysis with varying inflation layers, Re = 100,000.

Number of Inflation Layers	Average C_{drag}	Percent Error Relative to Interpolated C_{drag} Value (%)
0	2.397	6.246
2	2.301	1.974
4	2.239	-0.743
6	2.336	3.529
8	2.198	-2.57

according to the y^+ constraints. In both cases, the percent error is reasonable at 6.246% and 3.529% respectively.

When compared against the interpolated value, this free flow analysis shows OpenFOAM is generating realistic results, with an error up to approximately 7%. This provides a reasonable validation of the OpenFOAM model setup and provides confidence that the comprehensive verification study can proceed.

4.4 Comprehensive Verification Domain and Geometry

The comprehensive verification domain and geometry was constructed with a rectangular cylinder at 1/8 scale simulating the trailer section of a high sided vehicle near the ground surface. This rectangular cylinder was placed inside a 2D flow domain. Figure 4.3 shows a schematic for the flow domain with the key dimensions presented below.

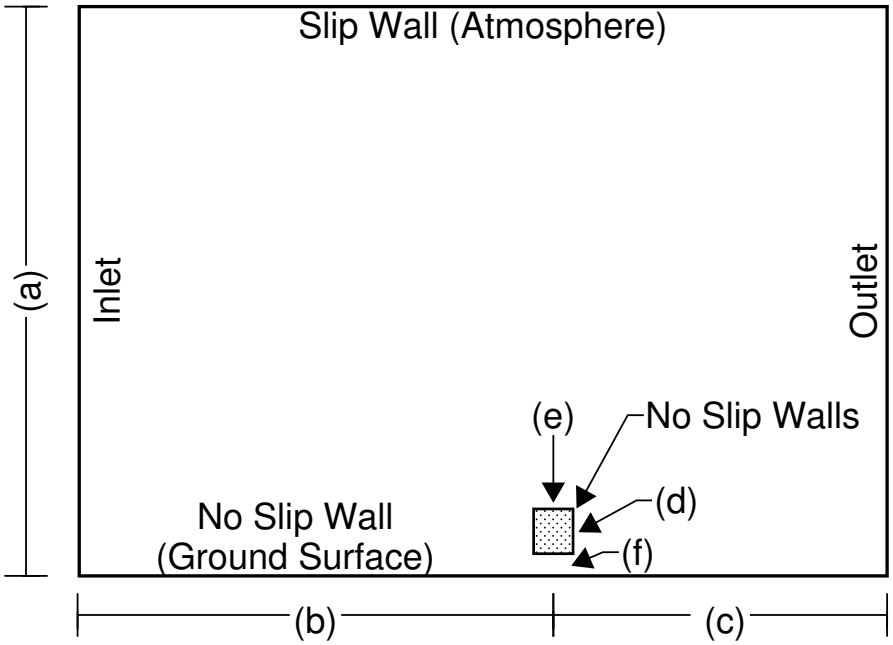


Figure 4.3: Comprehensive Verification Geometry – rectangular cylinder near a plane wall boundary.

The parameters (a), (b) and (c) are referenced in Table 4.2 and were varied within the given bounds during the comprehensive verification study. The dimensions of the rectangular cylinder are as follows: (d) cylinder height of 0.365m, (e) cylinder width of 0.324m and (f) gap height of 0.1485m between the ground surface and bottom of cylinder with a resulting gap ratio (gap height/cylinder height) (Forouzi Feshalami et al., 2022) of 0.407.

Table 4.2 provides the initial flow domain parameters, the value ranges explored for the comprehensive verification study, and the final parameters based on the comprehensive verification

study conclusion. These parameters are the focal point of this study and are discussed further below.

Table 4.2: Parameters explored in the comprehensive verification study.

	Initial Prescribed Value	Range Explored	Final Parameters from Comprehensive Verification Study
Number of Inflation Layers on Ground Surface	Not Prescribed	6-10	10
Number of Inflation Layers on Rectangular Cylinder	Not Prescribed	0-8	6
a) Vertical Domain Height	2m	2m-20m	12m
b) Horizontal Domain – Length Upstream from Cylinder Centroid	5m	2m-20m	10m
c) Horizontal Domain – Length Downstream from Cylinder Centroid	5m or 2m	2m-20m	7m
Uniform Cell Size	0.01m	0.005m-0.05m	0.01m

4.5 Comprehensive Verification Study

It was found the typical methods employed to conduct model verification are not satisfactory to ensure model accuracy. Often, model verification involves refining the cell mesh and inspecting how specific quantities throughout the system may change. Once the specified quantities have stabilized with a certain grid size, the mesh is independent of the final solution. This method is adequate if, and only if, the CFD practitioner is confident the CFD domain sizing is independent of the results.

The initial model prescribed values (Table 4.2) which were too small for the high Reynolds number flow as it became apparent the domain size was significantly influencing the model results. Because of this, variances in the domain size were investigated to determine the appropriate domain sizing in each coordinate direction such that the simulation results were independent of the domain sizing.

To perform the domain sizing verification, models were run using two distinct Reynolds numbers: 1.94 x 10^6 and 1.12 x 10^7 which are one order of magnitude apart. A parametric analysis was conducted, changing the parameters of one domain boundary at a time to determine when the model was domain independent.

4.5.1 Inflation Layer Selection

Because the $k-\omega\ SST$ model was selected to resolve the Navier-Stokes equations, the inflation layers had to be refined adequately to be within the prescribed bounds necessary for using the $k-\omega\ SST$ model, as well as optimizing computational expense so extra inflation layers were not needlessly refined. y^+ was confined to an absolute minimum of no less than 1, a preferred range of 40 to 300, with a preferred average of 150 (Menter et al., 2003).

The verification domain walls are the ground surface and the rectangular cylinder. Because these walls serve different purposes, of 1) confining the flow and 2) being the bluff body study object, it was found that independent inflation layer optimization would be needed. Once the inflation layers were selected, they were kept constant to remove a variable from the following analysis. The inflation layer selection simulations used a cell size of 0.01m.

Ground Surface Inflation Layers

The ground surface inflation layers were checked with the flow development simulations which are discussed in the Section 4.5.3, "Developed Flow Profile". The inflation layers were varied at 6, 8, and 10, and were evaluated at the previously stated high and low Reynolds numbers with a flow development section of 100m in length and 2m in height. Based on this analysis, 10 ground surface inflation layers provide an adequate y^+ range on the ground surface boundary.

Rectangular Cylinder Inflation Layers

The rectangular cylinder inflation layers were checked using a generic validation simulation with a 10m vertical domain height, 5m length upstream from cylinder centroid and 9m length downstream from cylinder centroid. In the same manner, this simulation was evaluated at the high

and low Reynolds numbers. Based on this analysis, 6 rectangular cylinder inflation layers provide the optimal means to stay within the prescribed y^+ boundaries.

4.5.2 Vertical Domain Height

The vertical domain size was investigated first. To accelerate the computational methods, a uniform flow profile was selected because the flow type (uniform flow versus a developed flow profile) is a second order variable with respect to the vertical domain height. Because the horizontal domain size had not been investigated yet, an arbitrary domain length was selected: 7m total, 5m upstream from the cylinder centroid and 2m downstream from the cylinder centroid. The vertical domain height varied from 2m up to 20m. Figure 4.4a shows the resulting average C_{drag} experienced by the body. Figure 4.4b shows the percent error relative to the minimum value for each Reynolds number (at 12m for each Reynolds number).

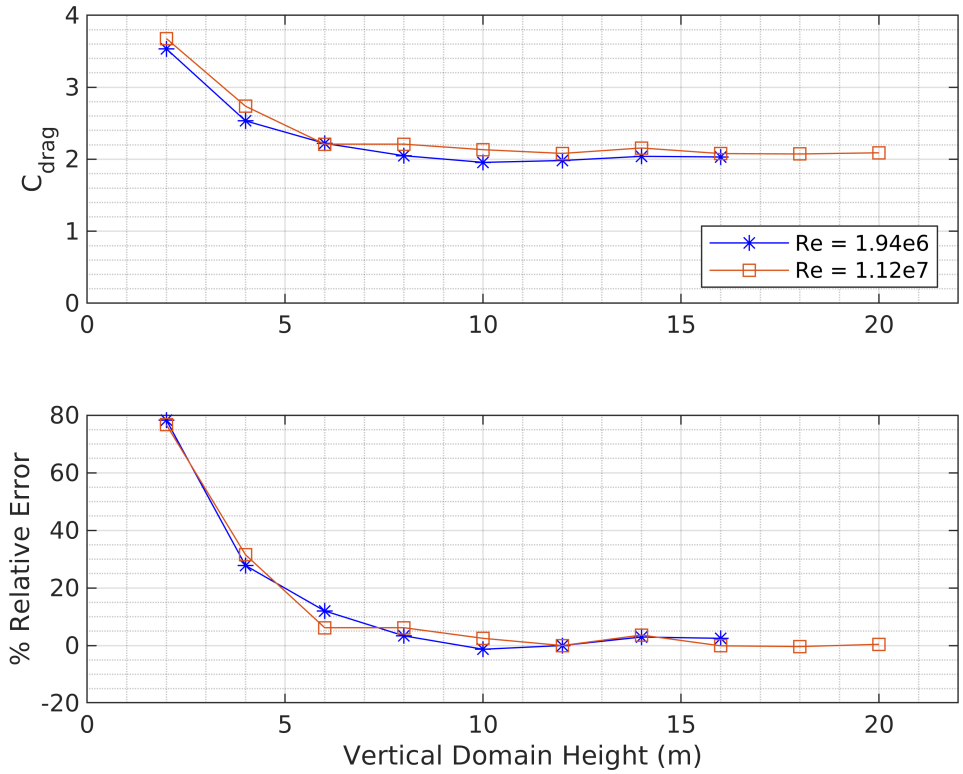


Figure 4.4: Comprehensive Verification Study Parameter of Interest, Vertical Domain Height: (a) C_{drag} versus Vertical Domain Height; (b) Percent error relative to selected converged value of 12m.

Based on these results, a qualitative analysis of Figure 4.4a shows a minimum vertical domain height of at least 8m is required for either Reynolds number. Looking at Figure 4.4b, it is quantitatively evident that C_{drag} is independent of the vertical domain height after 12m.

Figure 4.5 shows the difference in the instantaneous flow structure for a vertical domain height of 2m versus 12m. This figure depicts the compressed streamlines in the 2m vertical domain, leading to amplified velocity as evidenced by comparing the high velocity regions in each figure. The localized accelerated velocity artificially increases C_{drag} that is experienced by the body.

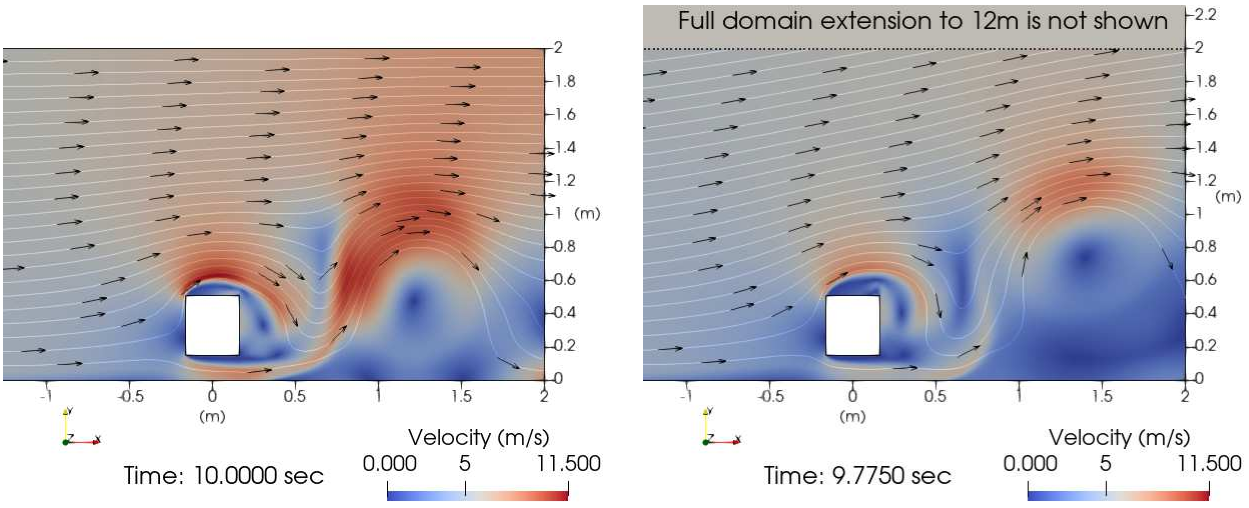


Figure 4.5: Compressed streamlines and accelerated flow evident in a vertical domain height of (a) 2m versus (b) 12m, Re = 1.94 x 10^6

4.5.3 Developed Flow Profile

To facilitate the repetitive parametric foundation of the comprehensive verification study, a developed flow profile was generated with the selected vertical domain height of 12m. The following models were fed a developed flow profile. Figure 4.6 shows the developed flow profile, which was generated using a 100m length section, 12m in height with a uniform flow profile at the inlet. The developed flow profile was taken from the outlet when the developed layer was approximately 0.6m which was high enough to sufficiently cover the total rectangular cylinder height.

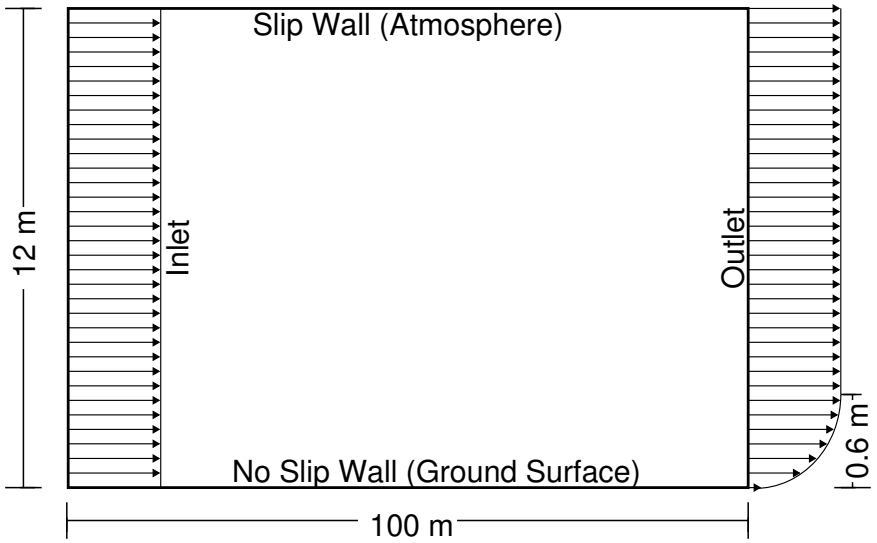


Figure 4.6: Using 100m section with uniform flow at inlet to generate developed flow profile at outlet.

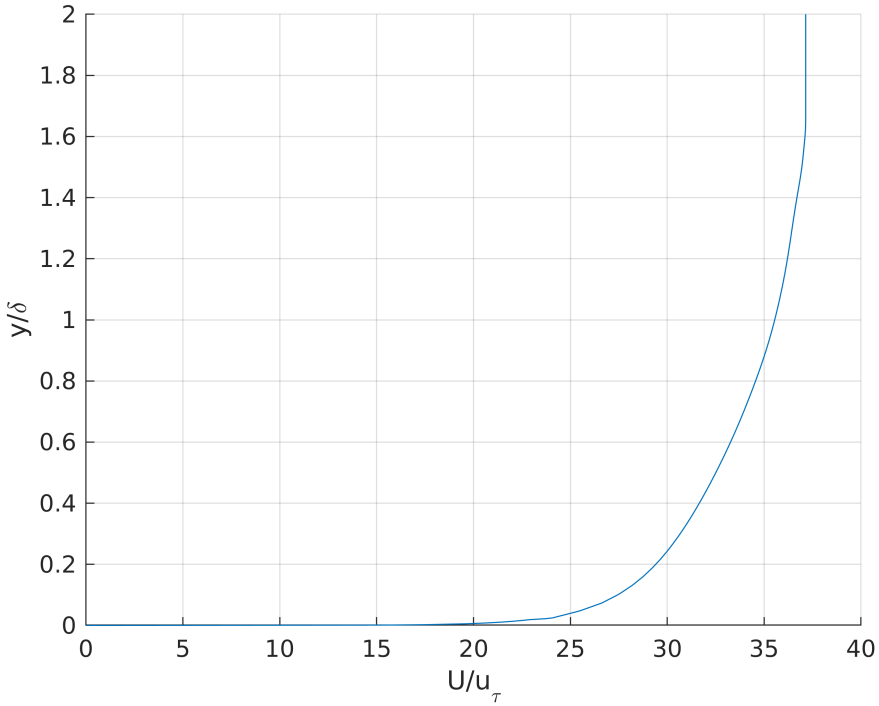


Figure 4.7: Non-dimensional plot of developed flow profile, where δ is the rectangular cylinder height and u_τ (friction velocity) $=\sqrt{\frac{\tau_w}{\rho}}$ (Pope, 2000). Shows developed flow up to a height of y = 0.6m, Re = 1.94 x 10^6.

Figure 4.7 shows a developing flow profile up to 0.6 m which is sufficient height to cover the entirety of the rectangular cylinder. The flow was not developed to cover the entirety of the 12m domain because the computational expense increases exponentially with the additional length necessary to increase the developed flow height. For example, a section 200m in length is required to generate developed flow at a depth of 0.8m, but this gain of 0.2 m will take approximately twice as long to computationally generate with respect to the 100m section flow development.

The developed flow profile was imposed on the subsequent verification models. When compared with a uniform flow profile, this reduced the computational cost expended, allowing the following simulations to focus on the flow dynamics associated with the rectangular cylinder.

4.5.4 Horizontal Domain: Length Upstream from Cylinder Centroid

The horizontal domain was investigated beginning with the length upstream from the cylinder centroid. As previously discussed, a developed flow profile was generated and used in this investigation. The length downstream from the cylinder centroid was set at an arbitrary 5m, while the upstream length varied from 2m up to 20m. Figure 4.8a shows the resulting average C_{drag} experienced by the body. Figure 4.8b shows the percent error relative to the selected upstream length's average C_{drag} (at 10m for each Reynolds number).

These results show an oscillatory behavior trending towards an average value. 10m was selected for both datasets because it is after the oscillations in the dataset have tempered down and it is the average value the system is converging towards as the parametric analysis continues beyond 10m. Figure 4.8b shows that this data collapses onto each other indicating similar behavior in the upstream length variation.

In tandem with the quantitative methods, a qualitative method was used to estimate the required horizontal domain length upstream from cylinder centroid. The results of a simulation with a 20m horizontal domain length upstream from cylinder centroid are shown on Figure 4.9, which depicts a pressure plot from the stagnation point on the centroid of the rectangular cylinder inlet face to the inlet boundary. This horizontal domain length of 20m was chosen as an arbitrary value that

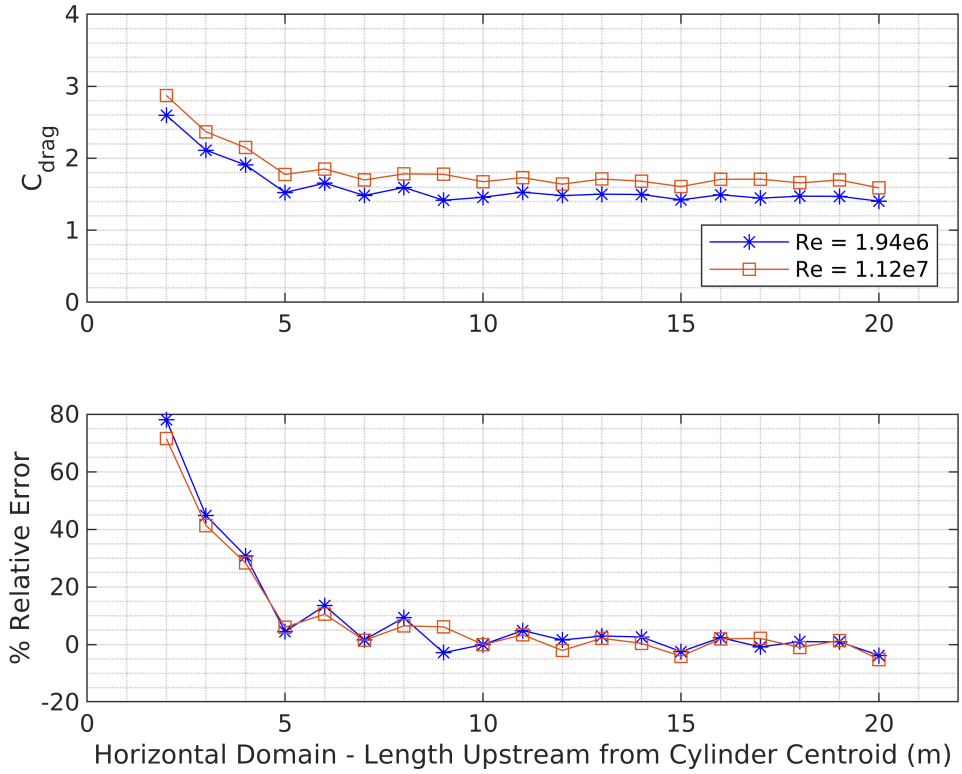


Figure 4.8: Comprehensive Verification Study Parameter of Interest, Horizontal Domain Length Upstream from Cylinder Centroid: (a) C_{drag} versus Horizontal Domain Length Upstream from Cylinder Centroid; (b) Percent error relative to selected converged value of 10m.

was assumed long enough to encompass the final domain length. Based on the pressure graph, approximately an 8m length is necessary to allow the stagnation pressure effects to dissipate prior to the inlet boundary implementation. Figure 4.9 qualitatively validates the quantitative results shown in Figure 4.8.

4.5.5 Horizontal Domain: Length Downstream from Cylinder

The final horizontal domain component was investigated with the length downstream from the cylinder centroid. A developed flow profile was used in this investigation. The length upstream from the cylinder centroid was set at an arbitrary 5m while the downstream length varied from 2m up to 20m. Figure 4.10a shows the resulting average C_{drag} experienced by the body. Figure 4.10b

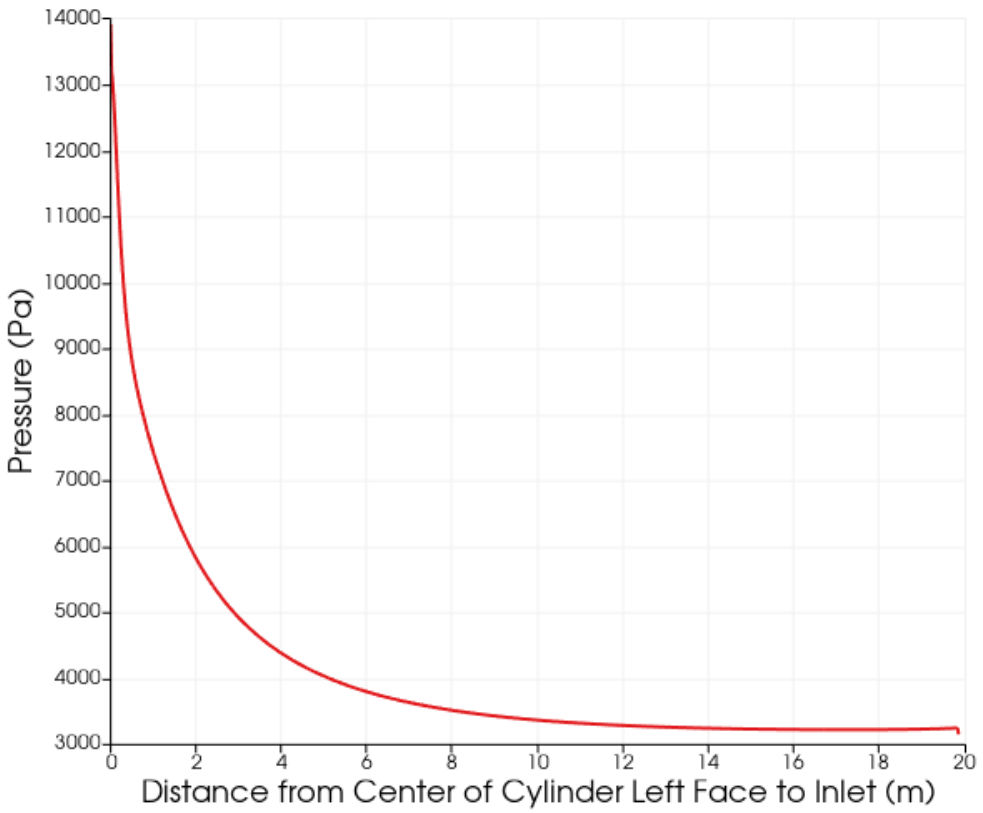


Figure 4.9: Averaged pressure along a line from the stagnation point on the rectangular cylinder to inlet, Re = 1.94 x 10^6.

shows the percent error relative to the selected downstream length's corresponding average C_{drag} (at 7m for each Reynolds number).

There was not an explicit trend that made a specific number ideal. However, it is best practice in wind tunnel experiments that the minimum downstream length is adequate such that the wake expands to its largest width before exiting the wind tunnel (Rae et al., 1999). In accordance with this perspective, 7m was selected because as seen in Figure 4.11, the wake width is the largest at this location and the flow exits the simulation with approximately horizontal velocity.

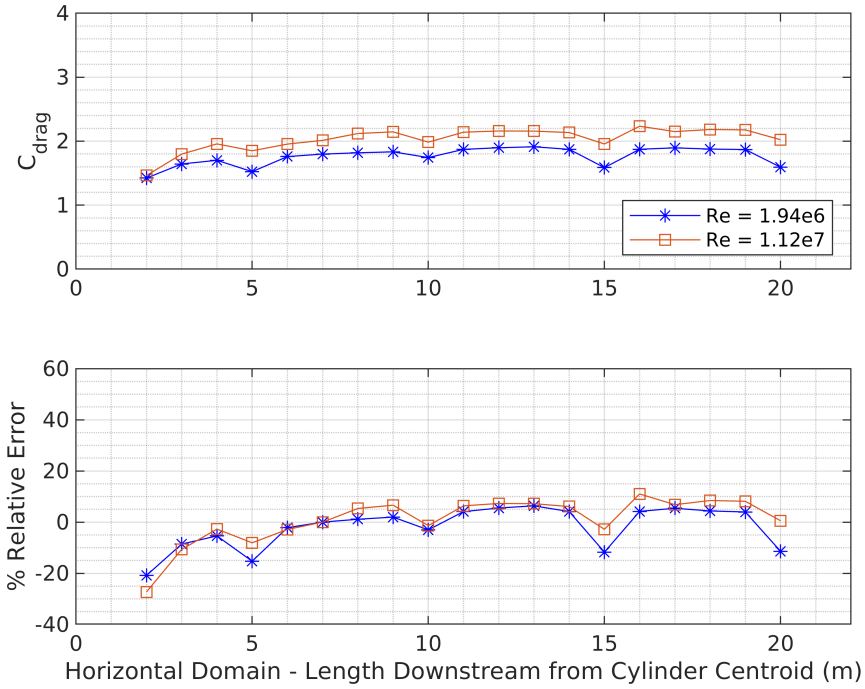


Figure 4.10: Comprehensive Verification Study Parameter of Interest, Horizontal Domain Length Downstream from Cylinder Centroid: (a) C_{drag} versus Horizontal Domain Length Downstream from Cylinder Centroid; (b) Percent error relative to selected converged value of 7m.

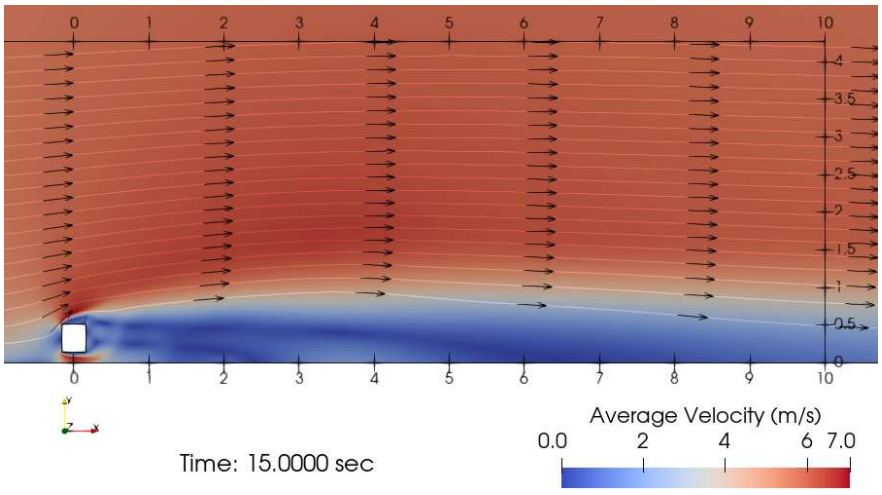


Figure 4.11: Average velocity with streamlines. Exiting flow at x = 7m is where the wake is at its largest width (Rae et al., 1999) and the velocity is approximately horizontal, Re = 1.94 x 10^6.

4.5.6 Traditional Grid Refinement Independence

Once the final domain parameters were established, a traditional grid refinement check was performed. Figure 4.12 shows that a mesh of about 2 x 10^6 cells is adequate; this correlates to a standard cell size of 0.01m. Therefore, the preceding results are independent of the grid sizing.

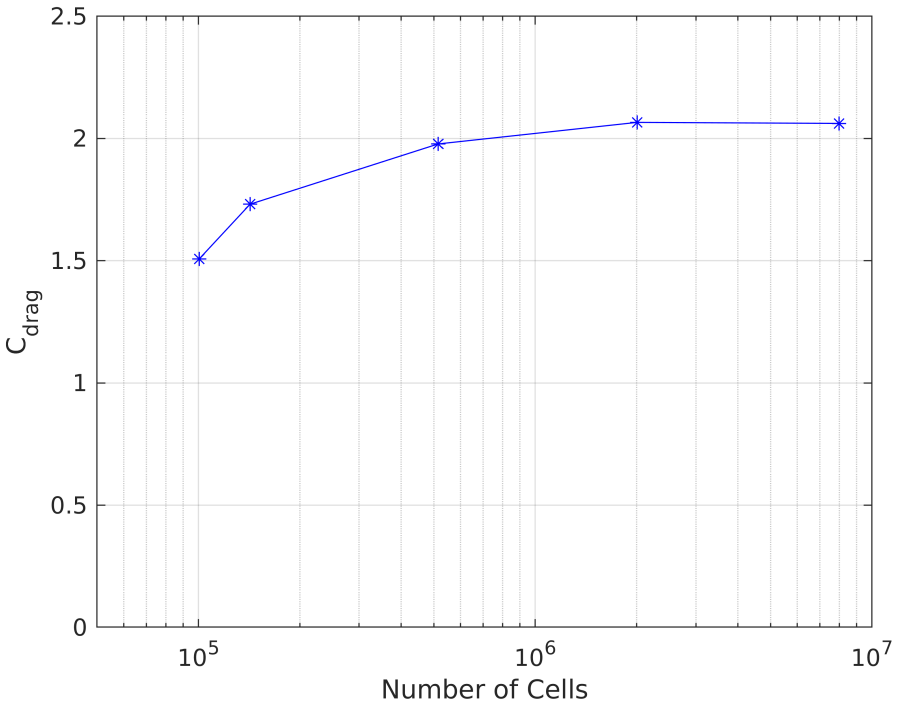


Figure 4.12: Comprehensive Verification Study Parameter of Interest, Traditional Grid Refinement, Re = 1.94 x 10^6.

4.6 Discussion of Comprehensive Verification Study Results

Based on this analysis, the final domain parameters presented in Table 4.2 were selected. The importance of performing this type of comprehensive verification study is evident in analyzing the error graphs. Figure 4.4b shows an 80% error if 2m was selected as the vertical domain height, down to 10% if 6m was selected. In Figure 4.8b, a 2m length upstream from cylinder centroid results in a 70% error. Lastly, Figure 4.10b shows an underprediction error of about -20% for a

2m length downstream from cylinder centroid. These potential errors are present because of the following reasons:

- If the vertical domain height is too small, the streamlines are compressed, as shown in Figure 4.5.
- If the length upstream from the cylinder centroid is too short, the stagnation pressure effect is compressed and not allowed to sufficiently develop, as shown in Figure 4.9.
- If the length downstream from the cylinder centroid is too short, the flow feature development is impeded and the flow wake is prevented from expanding to its full-size, as shown in Figure 4.11.

These effects are important to recognize for high Reynolds number flow, otherwise an improperly sized flow domain may significantly skew the simulation results.

4.7 Recommendations & Conclusion

Throughout the validation and comprehensive verification study, recommendations were identified for accelerating future comprehensive verification studies and to ensure accurate results are obtained. An initial validation study was performed using generic domain sizing. Once the final domain dimensions were found in the comprehensive verification study, this generic domain size was updated and the validation study was rerun to reflect the domain bounds of the comprehensive verification study.

A qualitative means to assess the appropriate domain sizing is required. Figures 4.5, 4.9 and 4.11 show different approaches to gauge the domain sizing effectiveness by running simulations with domain boundaries that are larger than the requirements. Once qualitative estimates of the domain boundaries are obtained, a comprehensive verification study must be conducted. To reduce the time required to complete the comprehensive verification study, varying the domain size centered around the qualitative estimates aids in quickly converging on the model behavior. Additionally, evaluate the data to generate plots similar to those in Figures 4.4, 4.8 and 4.10, initially

using larger increments such as 5m, and reducing the increments to 2m then 1m once the overall trends in the model behavior have been coarsely established.

A case for a comprehensive verification study as it pertains to 2D incompressible high Reynolds number flow around a bluff body near a plane wall boundary has been presented. Because of the complex flow dynamics associated with this incompressible flow, the domain sizing is of utmost importance in tandem with a traditional grid independence analysis. Without domain size verification, an error rate up to 80% is possible due to improper domain sizing in any coordinate direction. Qualitative means for assessing the domain boundary sizing have been presented for a twofold purpose: 1) To allow the user to understand the flow dynamics that are influenced with each boundary, and 2) to provide a starting estimate for the required domain boundary sizing for a new CFD model.

Chapter 5

Modeling high Reynolds number incompressible flow around a rectangular cylinder near a plane wall boundary, and an application to overturning high sided vehicles[2]

5.1 Introduction

Flow around bluff bodies has been a consistent topic of interest in fluid mechanics as evidenced by our continual fascination with von Kármán type flow structures. However, flow around a rectangular cylinder near a plane wall boundary has not been as thoroughly studied. This paper fills two identifiable knowledge gaps in two distinct fields of study: a fundamental field and an application field.

First, in the fundamental field, previous research investigating incompressible flow around rectangular cylinders near a plane wall boundary has been constrained to Reynolds numbers between 50 to 4.12 x 10^5 in the laminar and transitionary range (Bhattacharyya and Maiti, 2004; Cheng et al., 2007; Mahir, 2009; Yang et al., 2021, 2022; Forouzi Feshalami et al., 2022). This study goes beyond these values and investigates high Reynolds number incompressible flow up to 10^8, filling a substantial gap in the literature.

Second, the impetus of this paper was derived from an application of high sided vehicles overturning due to high crosswind, as shown by seminal research (Baker, 1986) and recent news events (Beedie, 2021). In this application, bulk parameter coefficients are used to correlate the wind speed to the force exerted on the high sided vehicle. Previous work on investigating the overturning of

[2]The research presented in this chapter is being prepared for submission to the *Journal of Fluid Mechanics*. Background information and literature relevant to this chapter are presented again so the chapter may be read as a standalone work.

high sided vehicles and the bulk parameter coefficients has typically involved limited full-size studies and a variety of wind tunnel models (Baker and Soper, 2022). Baker and Soper (2022) showed in their compiled dataset of crosswind experiments for flat ground that there is wide variability of the rolling moment coefficient ($C_{rolling}$) for yaw angles between 45° and 90° (equivalent to a direct crosswind) (see Baker and Soper (2022), and a reproduction here as Figure 5.1).

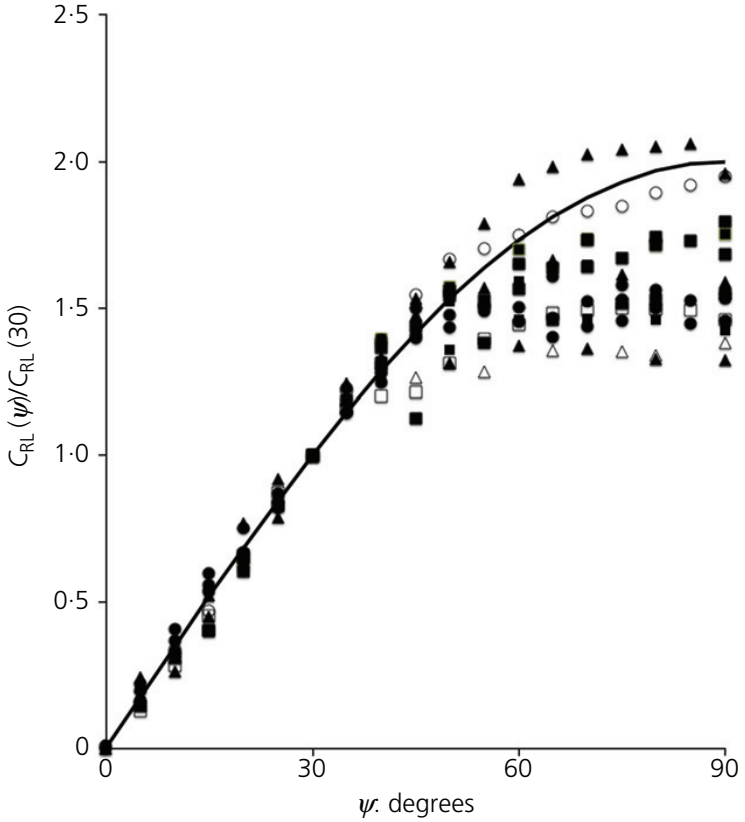


Figure 5.1: Reproduction from Baker and Soper (2022) – Leeward $C_{rolling}$ plotted against relative wind angle of attack, where 90° is a direct crosswind.

These variations can be explained by the models reflecting distinct types of high sided vehicles and the Reynolds numbers used in the analyses. The Reynolds numbers ranged from 8.5 x 10^4 to 1.25 x 10^5; these values were found to be too low to fully represent realistic Reynolds numbers such that overturning accidents may occur. These Reynolds numbers were used because select studies from Baker and Soper assert that the flow dynamics are independent of Reynolds number;

this assertion is based on an understanding that the Reynolds numbers used in the analysis are above a $Re_{critical}$ value, such that above $Re_{critical}$, the characteristics of the flow are Reynolds number independent (Baker, 1991; Coleman and Baker, 1994; Baker and Humphreys, 1996). It is pertinent to acknowledge that most studies on the overturning of high sided vehicles do not conduct a Reynolds number dependency test. This study aims to close this research gap by providing an analysis of the flow characteristics and aerodynamic coefficients at Reynolds numbers that reflect real-world conditions.

The analysis of overturning high sided vehicles has been generalized in order to correlate the study with the literature of high Reynolds number incompressible flow around a rectangular cylinder near a plane wall boundary. This generalization involves focusing on the trailer section of a high sided vehicle, approximating the trailer as a 2D cross section. The simplification will demonstrate the dependency of flow characteristics on the Reynolds number due to the rectangular cylinder being near a plane wall boundary, compared to a rectangular cylinder in free flow.

The layout of this paper is as follows: Section 5.2 presents a nondimensional analysis and introduces the numerical methodology, the parameters used in OpenFOAM, the study framework and briefly outlines the comprehensive verification and validation analysis completed for this research. Section 5.3 highlights the results and provides a discussion. Section 5.4 concludes this study.

5.2 Theory, Methods, and Model Setup

5.2.1 Non-dimensional Analysis

A non-dimensional analysis was performed using the Buckingham pi theorem. Figure 5.2 identifies some of the system variables.

These variables have been stated in their base dimensions using the units of mass (M), length (L) and time (T). First, the dimensions of length [L], width [W], and height [H] are represented using the unit (L). Additional variables include the relative wind velocity $[V] = \frac{L}{T}$, the relative wind direction (with respect to the vehicle length axis) [ψ] = 1, the density of air $[\rho] = \frac{M}{L^3}$, the dynamic viscosity of air $[\mu] = \frac{M}{LT}$, pressure (such as lift or drag) $[P] = \frac{M}{LT^2}$, and the shedding frequency

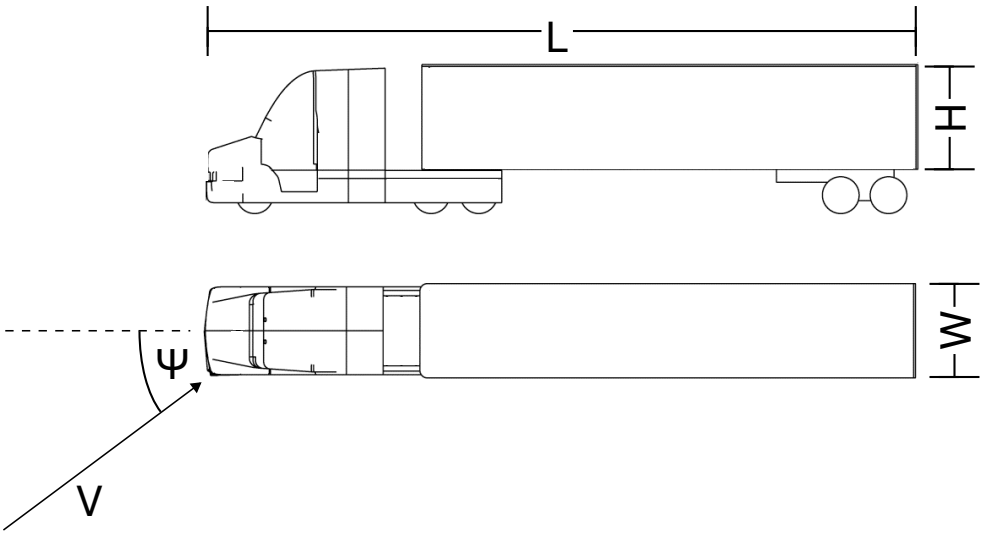


Figure 5.2: Schematic of some variables identified for Buckingham pi theorem non-dimensional analysis.

$[\omega] = T^{-1}$. There are 7 distinct variables based on the identification of the unit combinations (i.e. length, width and height act as a single variable with respect to a unit perspective). This results in 4 expected Π terms (7 variables minus 3 base dimension types).

Scaling variables are used to define the Π term. It was determined that the scaling variables M, L and T would be defined as follows: $L = [H]$, $T = [\frac{H}{V}]$, and $M = \rho H^3$. Using these scaling variable definitions, the Π terms were defined as: $[\psi] = 1$, $[\mu] = \rho HV$ where $Re = \frac{\rho HV}{\mu}$, $[P] = \rho V^2$ where $Eu = \frac{P}{\rho V^2}$, and $[\omega] = \frac{V}{H}$ where $St = \frac{\omega H}{V}$. The resulting non-dimensional equation is:

$$Eu = f(Re, St, \psi) \tag{5.1}$$

The Euler number is a general form of aerodynamic coefficients. ψ is left constant at a direct crosswind, resulting in an analysis between the Euler number, Reynolds number, and the Strouhal number. The focus of this study is the variance of the aerodynamic coefficients (Euler) and the Reynolds number. The Strouhal number will be evaluated with respect to Reynolds number, however, it was found that the variance was not significant.

Because the Euler number is a general form of aerodynamic coefficients, the aerodynamic coefficients will be given here to explicitly define their use in this paper (Blevins, 1984):

$$C_{drag} = \frac{F_D}{0.5\rho A V_r^2} \tag{5.2}$$

$$C_{side} = \frac{F_S}{0.5\rho A V_r^2} \tag{5.3}$$

$$C_{lift} = \frac{F_L}{0.5\rho A V_r^2} \tag{5.4}$$

$$C_{rolling} = \frac{R_L}{0.5\rho A h V_r^2} \tag{5.5}$$

In equations 5.2 through 5.5, where F_D are the drag forces exerted on the entire body, and F_S and F_L are the forces exhibited on the reference area. R_L is the overturning moment on a vehicle about the leeward wheels, A is the reference area, h is the reference height, and V_r is the wind velocity relative to the vehicle (Baker and Soper, 2022).

5.2.2 Governing Equations

The governing equations for this parametric study are the Navier-Stokes equations. Assuming incompressible flow ($\nabla \cdot \mathbf{u} = 0$), the simplified Navier-Stokes equations are as follows (Kundu et al., 2016):

$$\rho \left(\frac{\partial u_j}{\partial t} + u_i \frac{\partial u_j}{\partial x_i} \right) = -\frac{\partial P}{\partial x_j} + \rho g_j + \mu \frac{\partial^2 u_j}{\partial x_i^2} \tag{5.6}$$

The following section discusses the discretization and solution methodology.

5.2.3 CFD Solver - OpenFOAM

OpenFOAM, an opensource computational fluid dynamics (CFD) platform (Weller et al., 1998), was selected as the CFD solver because it is customizable. These configurable and programming capabilities were deemed beneficial to conduct a separate parametric study (Sanchez and Venayag-

amoorthy, 2023). OpenFOAM has been investigated and validated for incompressible flows around a bluff body using the finite volume method (Robertson et al., 2015). This study utilized OpenFOAM version 9 and sought to implement best practices in the OpenFOAM environment. Because the Reynolds number is high, an unsteady flow field was expected. As a result, a URANS (Unsteady Reynolds Averaged Navier-Stokes) simulation was desirable and the PISO (Pressure Implicit with Splitting of Operators) method (Issa, 1986) was implemented to facilitate URANS. To close the Navier-Stokes equation, the $k-\omega\ SST$ model was utilized because it is a y^+ insensitive model, which is necessary when performing traditional grid refinement studies (Menter et al., 2003) (where y^+ is a nondimensional measure of wall distance units; $y^+ = \frac{u_\tau y}{\nu}$ (Pope, 2000)). The PISO method was selected to solve the Navier-Stokes equations because of the transient nature of the unsteady simulations (Greenshields, 2021). Because turbulence is a diffusive process (Kundu et al., 2016), a first order upwind spatial scheme was implemented for the turbulence variables. The time discretization used the Crank-Nicolson method (Crank and Nicolson, 1947) with the blending factor set to 0.7, resulting in an approximately second order accurate time scheme.

The following considerations were used at each patch surface for the respective initial conditions and boundary conditions. The model walls are the ground surface and surface of the rectangular cylinder; on these surfaces, the $k-\omega\ SST$ model was implemented with a no-slip velocity boundary condition, a zero-gradient boundary condition for pressure and turbulent kinetic energy, and a log-law blending function for the specific dissipation rate. At the atmospheric boundary, a free-slip condition was allowed. At the outlet, the flow was assigned a zero-gradient condition. At the flow inlet, a zero-pressure gradient condition was specified, with a developed flow profile implemented using a fixed value condition for the velocity, specific dissipation rate, turbulent kinetic energy and turbulent viscosity. The turbulence intensity was set at 1%. The developed flow profile is further discussed in the comprehensive verification and validation analysis (Sanchez and Venayagamoorthy, 2023).

5.2.4 Model Parametric Framework

The 3D model shown in Figure 5.2 has been simplified into a rectangular cylinder near a plane wall boundary and is shown in Figure 5.3, with the dimensions of the rectangular cylinder given by the trailer section of a high sided vehicle. The characteristic length scale is the height of the rectangular cylinder (0.365m) at model scale. Additional details are provided in the following section.

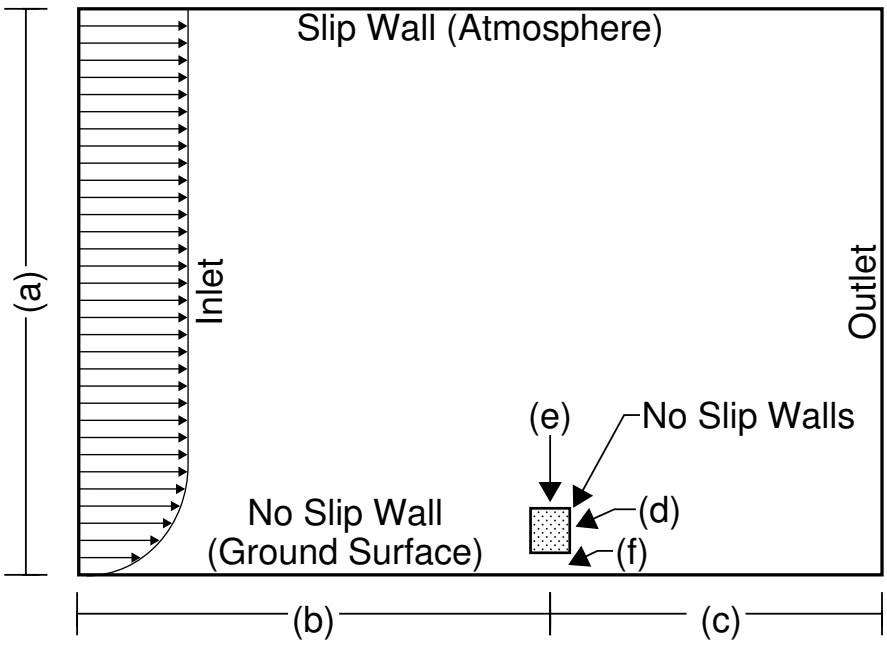


Figure 5.3: Rectangular cylinder near a plane wall boundary with a developed inlet flow profile.

Verification of the CFD model was completed at the Reynolds numbers of 1.94 x 10^6 and 1.12 x 10^7. Validation of the CFD model used a Reynolds number of 100,000. For this study, Reynolds numbers ranging from 10^4 to 10^8 were evaluated. This range partially envelopes the Reynolds numbers used in the fundamental field of study (50 up to 4.12 x 10^5), completely covers the Reynolds numbers used in existing application research (8.5 x 10^4 up to 1.25 x 10^5), and includes Reynolds numbers that are at realistic wind speeds such that overturning accidents could occur (between 1.94 x 10^6 to 1.12 x 10^7).

5.2.5 Verification and Validation

Upon investigating various CFD studies on overturning of high sided vehicles, it was found that the CFD studies lacked a rigorous and comprehensive numerical analysis, making the accuracy and realism of the results unknown (Grm and Batista, 2017; Salati et al., 2018; Tunay et al., 2020; Zhang et al., 2020). This section includes the pertinent details and results of the comprehensive verification and validation analysis (see Sanchez and Venayagamoorthy (2023) for details).

The initial validation of OpenFOAM was completed in Sanchez and Venayagamoorthy (2023) with the conclusion drawn that OpenFOAM is generating realistic results, with an error of approximately 7%. This resulted in a reasonable validation of the OpenFOAM model setup and provided confidence that the comprehensive verification analysis and this study can proceed. The comprehensive verification analysis was conducted because it was determined that evaluating model verification solely through a traditional grid independence study was inadequate. The comprehensive verification study evaluated the sizing of the domain until it was determined that the flow characteristics were independent of the domain sizing. After this, a traditional grid independence study was performed to finalize the comprehensive verification analysis.

5.2.6 Model Setup

The final dimensions for the rectangular cylinder in free flow and near a plane wall boundary are as follows based on the results in Sanchez and Venayagamoorthy (2023) and are seen on Figure 5.3: (a) vertical domain height of 12m, (b) horizontal domain length upstream from the cylinder centroid of 10m, (c) horizontal domain length downstream from the cylinder centroid of 7m, (d) cylinder height of 0.365m, (e) cylinder width of 0.324m with a resulting width-to-height ratio of 0.887 and (f) gap height of 0.1485m between the ground surface and bottom of cylinder with a resulting gap ratio (gap height/cylinder height) (Forouzi Feshalami et al., 2022) of 0.407. A cell size of 0.01m was used based on the results of the grid independence study.

The number of inflation layers required on each wall surface was dependent on the Reynolds number and evaluated by measuring y^+. To this end, y^+ was confined to an absolute minimum of

no less than 1, a preferred range of 40 to 300, with a preferred average of 150 (Menter et al., 2003). Because of these bounding criteria, there was flexibility in the number of layers prescribed for a given range of Reynolds numbers. For Reynolds numbers from 10^4 to 7 x 10^4, no inflation layers were specified. For Reynolds numbers from 10^5 to 1.12 x 10^7, six inflation layers were specified for the rectangular cylinder surface, and ten inflation layers were specified for the ground surface. Lastly, for the Reynolds number of 10^8, twenty inflation layers were specified for the rectangular cylinder surface, and twenty-four inflation layers were specified for the ground surface.

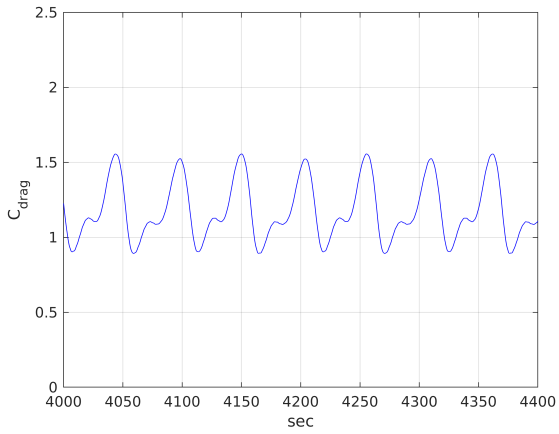
5.3 Results and Discussion

With the comprehensive verification and validation analysis complete (Sanchez and Venayagamoorthy, 2023), the focus of this paper can proceed. It will be demonstrated that the characteristics of flow around a rectangular cylinder near a plane wall boundary is dependent on Reynolds number and the wake is asymmetric, whereas the characteristics of a rectangular cylinder in free flow is not dependent on Reynolds number and the wake is symmetric. Therefore, the application to the rolling moment coefficient ($C_{rolling}$) is dependent on Reynolds number.

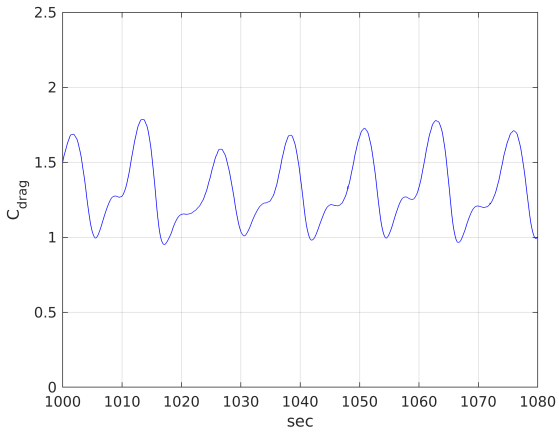
5.3.1 Symmetric and Asymmetric Flow Structures

Figures 5.4 and 5.5 show the time signature for the drag coefficient (C_{drag}) for flow around a rectangular cylinder near a plane wall boundary and in free flow respectively; this is plotted against time for Reynolds numbers 2.5 x 10^4, 10^5, 1.94 x 10^6, and 1.12 x 10^7, respectively.

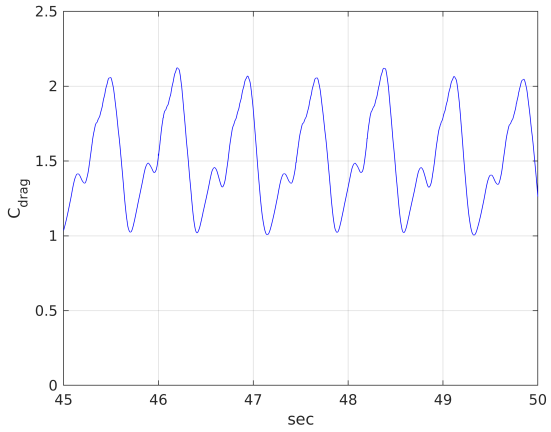
When comparing the graphs between Figures 5.4 and 5.5, the dynamics of the flow are cyclical and similar in that there are peaks and valleys in each time signature. The differences arise in that the dynamics of the time signature are suppressed in the flow around a rectangular cylinder near a plane wall boundary (Figure 5.4) and are more dynamic with a rectangular cylinder in free flow (Figure 5.5). This suggests that the influence of the ground surface is significant and dampens the oscillations present in the flow field.



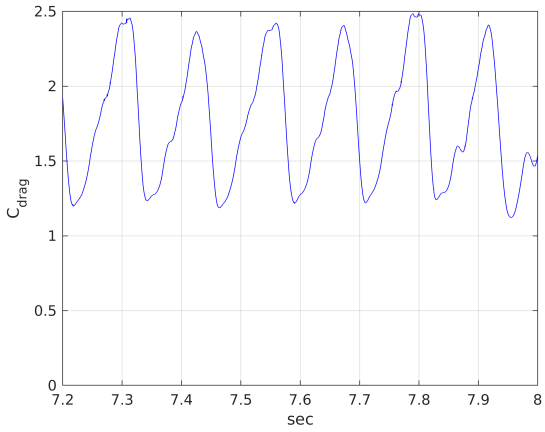
(a) Re = 2.5 x 10^4



(b) Re = 10^5

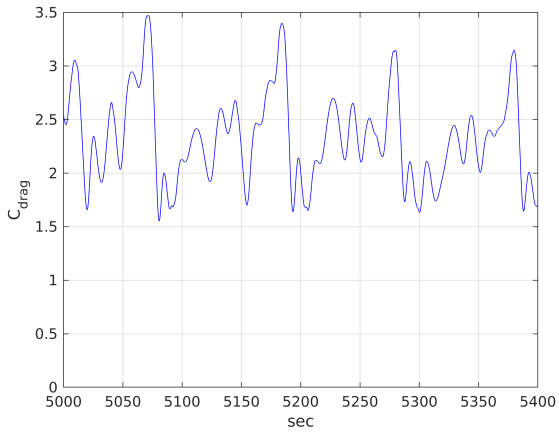


(c) Re = 1.94 x 10^6

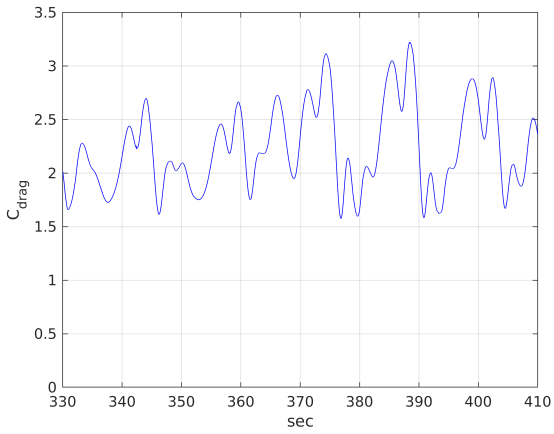


(d) Re = 1.12 x 10^7

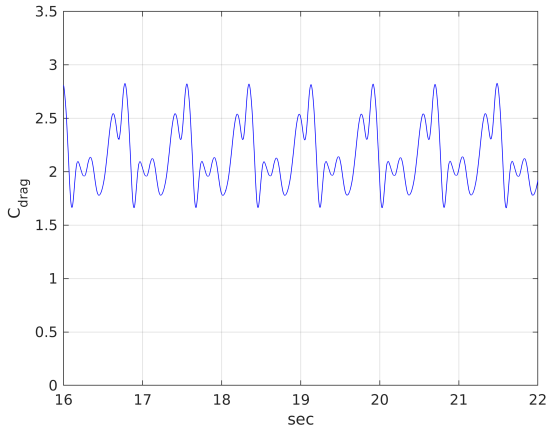
Figure 5.4: Time signature for C_{drag} for flow around a rectangular cylinder near a plane wall boundary, at Reynolds numbers (a) 2.5 x 10^4, (b) 10^5, (c) 1.94 x 10^6, and (d) 1.12 x 10^7.



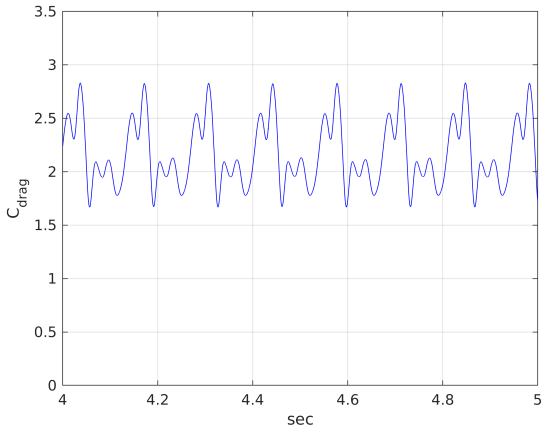
(a) Re = 2.5 x 10^4



(b) Re = 10^5



(c) Re = 1.94 x 10^6



(d) Re = 1.12 x 10^7

Figure 5.5: Time signature for C_{drag} for a rectangular cylinder in free flow, at Reynolds numbers (a) 2.5 x 10^4, (b) 10^5, (c) 1.94 x 10^6, and (d) 1.12 x 10^7.

The following analysis graphically looks at the flow field at various timesteps, correlating the flow field snapshots with the equivalent points on their respective C_{drag} plots. Figures 5.6 and 5.7 show the oscillatory von Kármán type behavior that persists in flow around a rectangular cylinder near a plane wall boundary at Reynolds numbers of 1.94 x 10^6 and 1.12 x 10^7, respectively. These figures show the instantaneous URANS velocity field with the blue color denoting low velocity flow and a red color denoting high velocity flow, the streamlines in white, and the Q-criterion in black at a value of 0.5; where the Q-criterion is the second invariant of the velocity gradient tensor, or more simply, the Q-criterion defines vortices as areas where the vorticity magnitude is greater than the magnitude of the rate of strain (Zhan et al., 2019). In Figures 5.6 and 5.7, each image iteration shows the flow progressing through the oscillatory state, with image (a) identifying accelerated flow in the gap between the rectangular cylinder and the ground surface (correlating to the peak of the C_{drag} time history), (b) highlighting the accelerated flow jetting up on the backside of the rectangular cylinder (correlating to about halfway down the falling limb of the C_{drag} time history), (c) showing the accelerated flow impinging on the top face of the rectangular cylinder (correlating to the C_{drag} trough), and (d) exhibiting the underflow from the beginning of the cycle detaching from the cylinder and propagating along the ground surface (correlating to the midpoint of the rising limb).

In Figures 5.8 and 5.9, each image iteration shows the flow progressing through an oscillatory state, similar to that shown in Figures 5.6 and 5.7, but for the dynamics of a rectangular cylinder in free flow. Image (a) shows accelerating flow on the top of the cylinder (correlating to the C_{drag} peak), (b) identifies the high flow rate transitioning towards the bottom of the cylinder (and at the C_{drag} trough), (c) exhibits the accelerated flow greatest at the bottom of the cylinder (and C_{drag} is slightly recovered), (d) shows the flow again accelerating at the top of the cylinder (and C_{drag} has moderately recovered) and (e) highlights the flow at the beginning of the oscillatory cycle, but the accelerated flow is mirrored and now at the bottom of the rectangular cylinder.

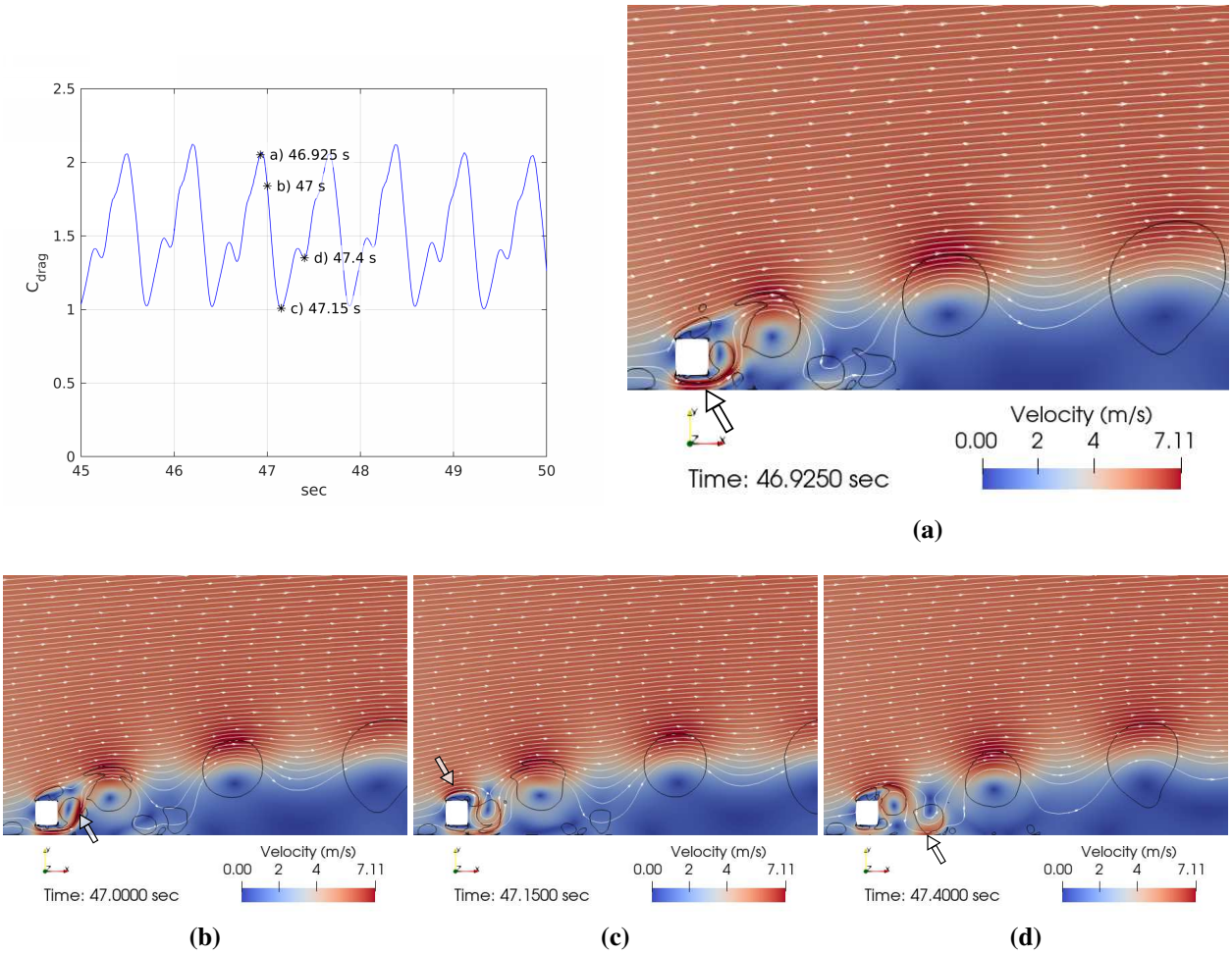


Figure 5.6: A sequence of images showing the oscillatory and asymmetric nature of flow around a rectangular cylinder near a plane wall boundary, at a Reynolds number of 1.94 x 10^6.

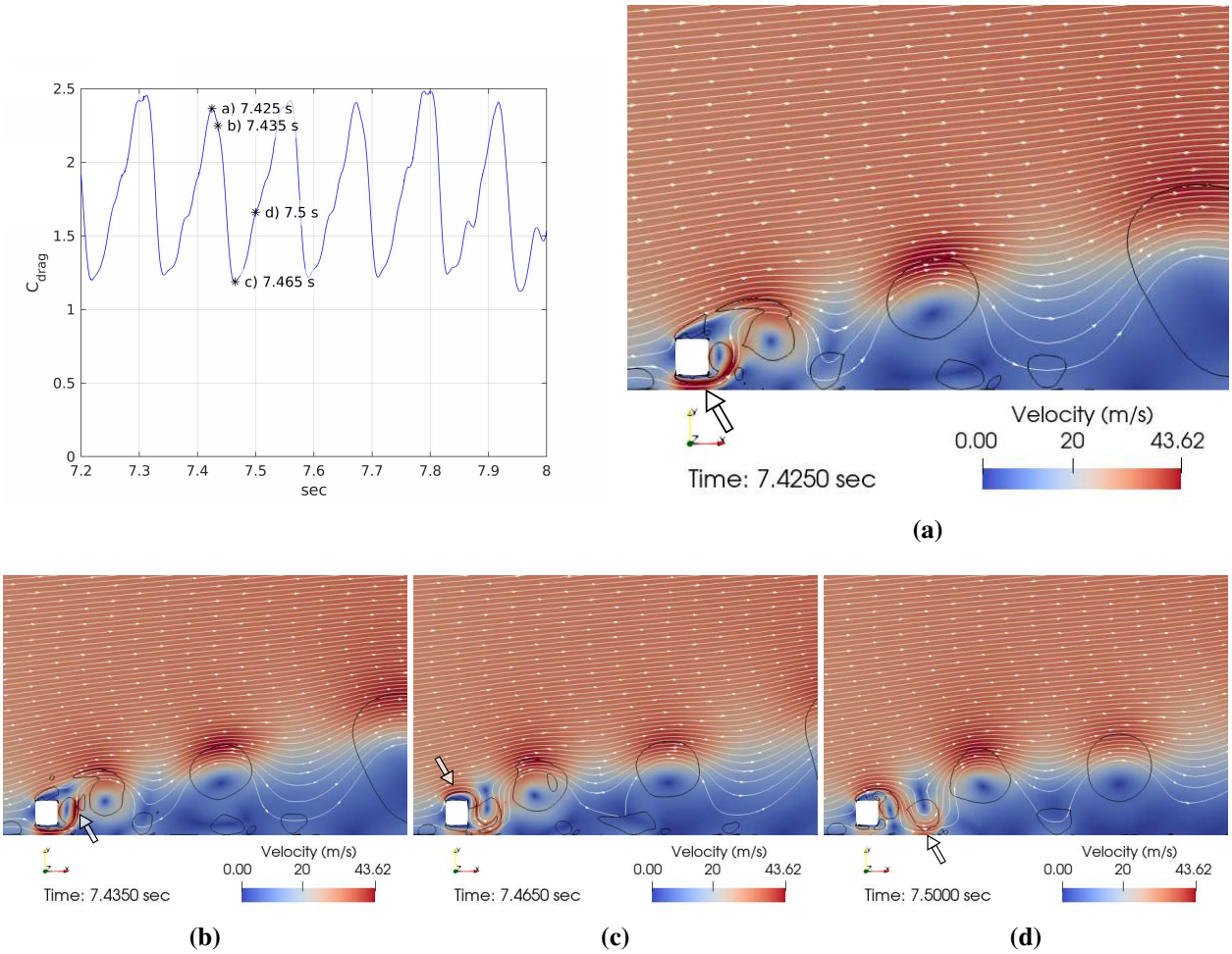


Figure 5.7: A sequence of images showing the oscillatory and asymmetric nature of flow around a rectangular cylinder near a plane wall boundary, at a Reynolds number of 1.12 x 10^7.

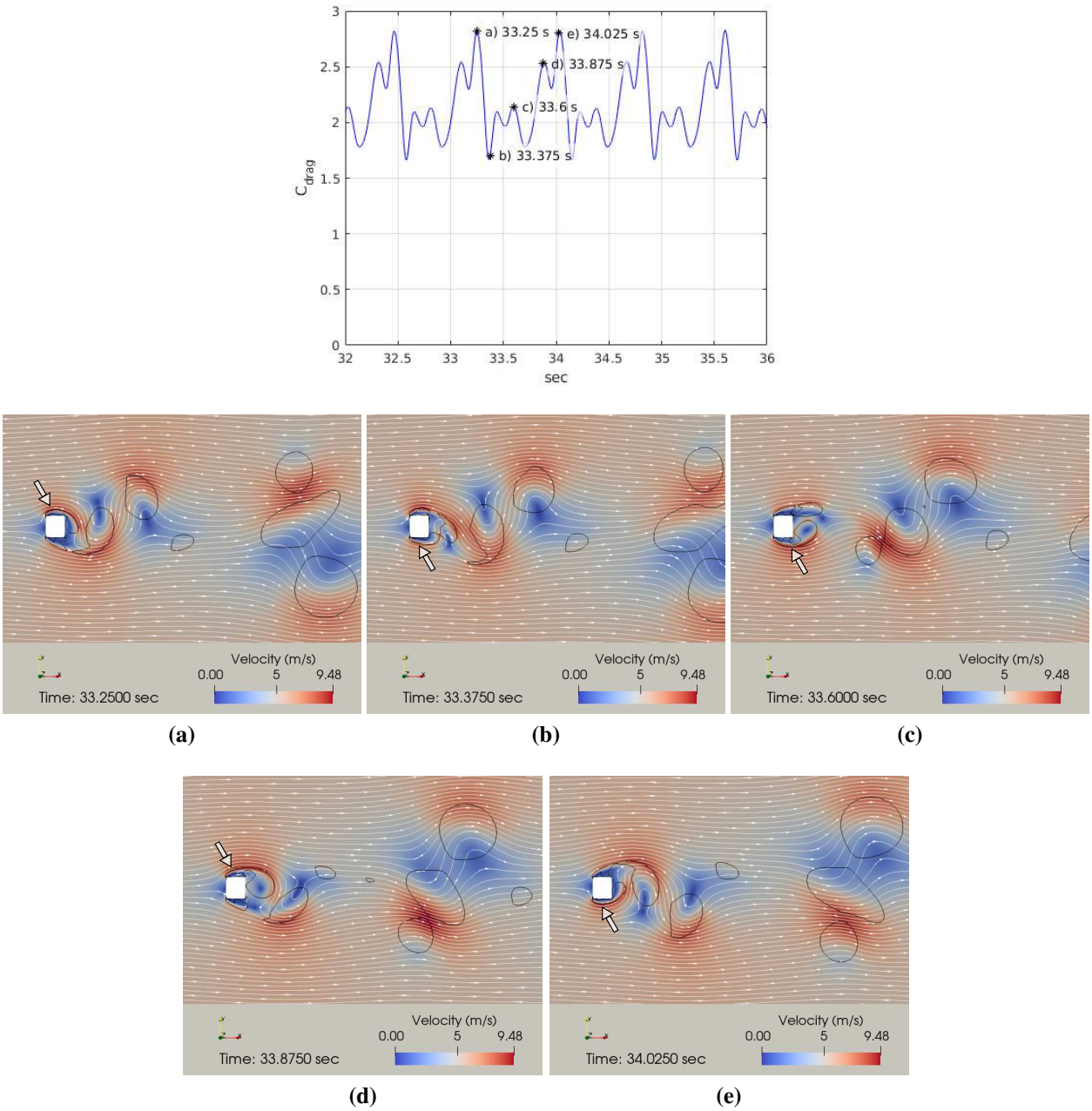


Figure 5.8: A sequence of images showing the oscillatory and symmetric nature of a rectangular cylinder in free flow, at a Reynolds number of 1.94 x 10^6.

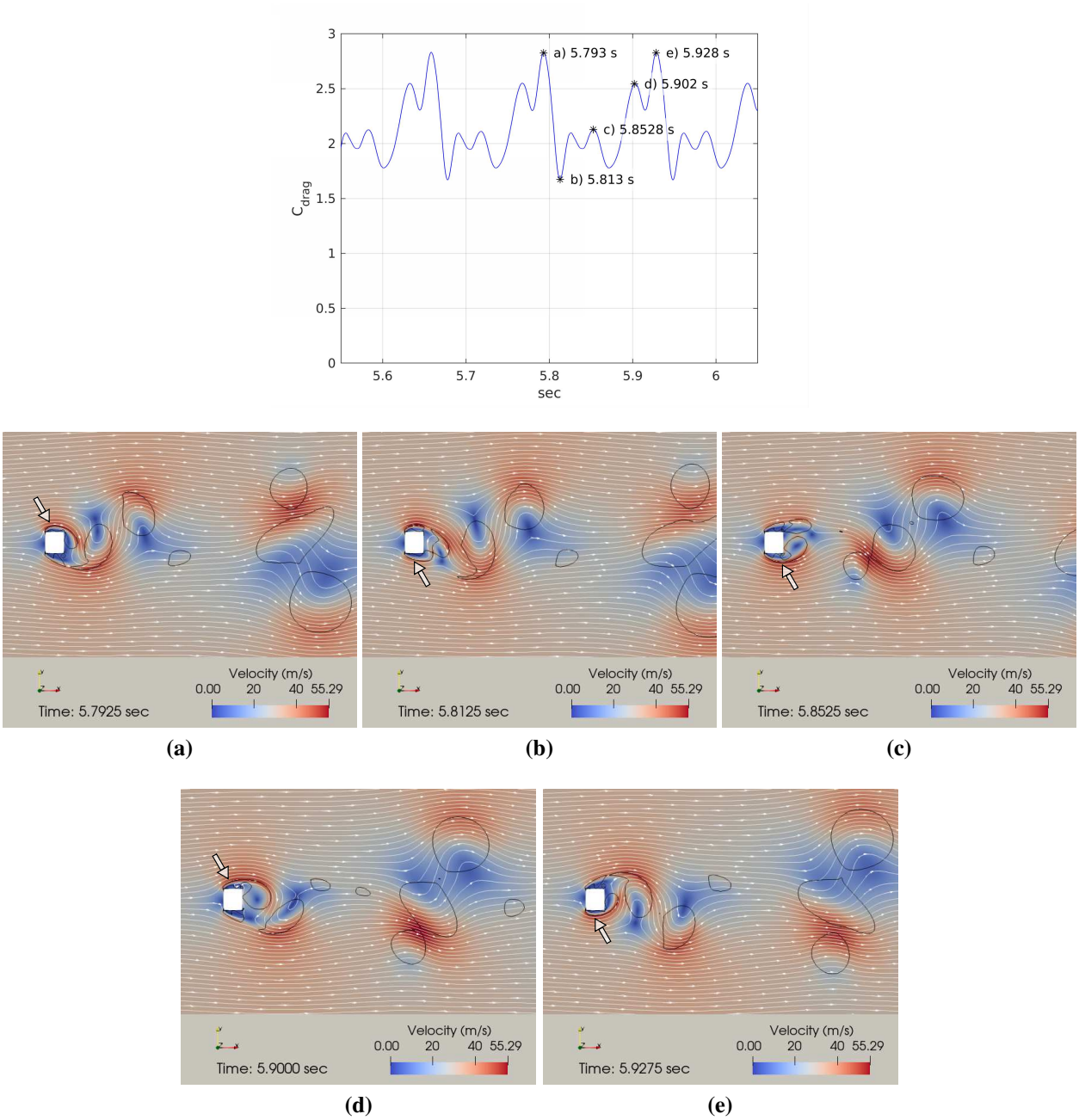
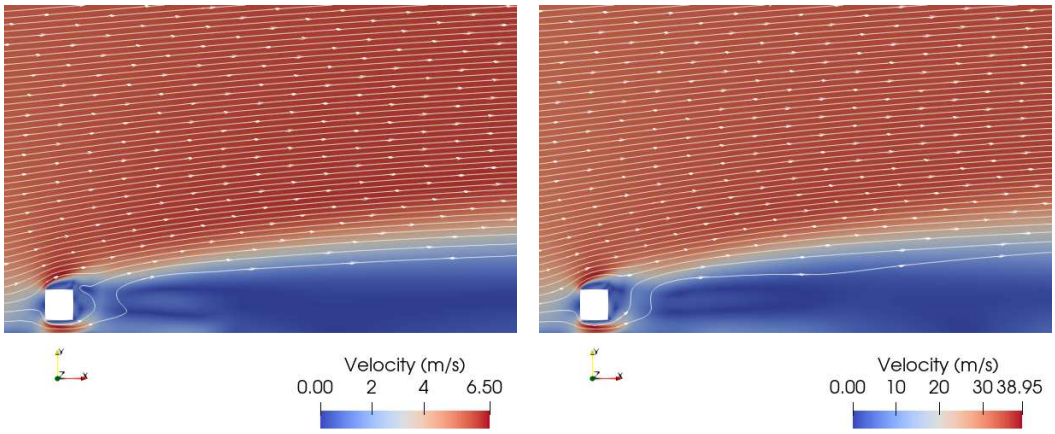


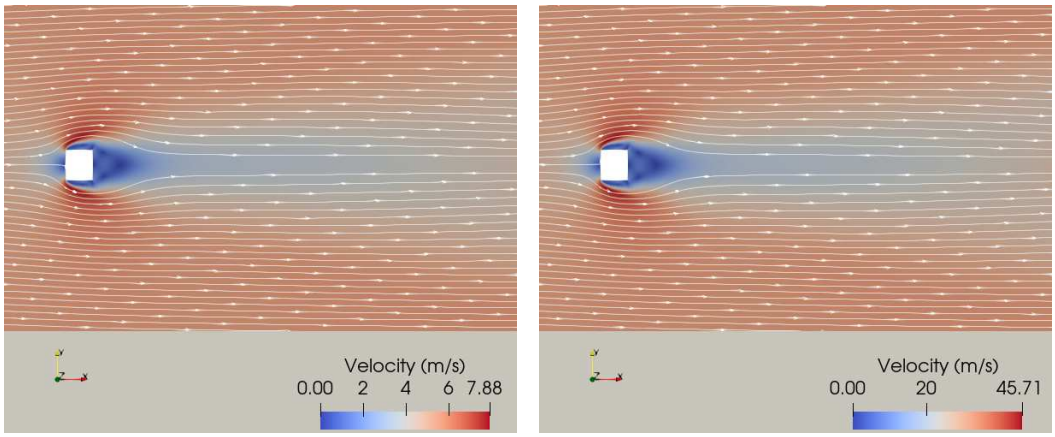
Figure 5.9: A sequence of images showing the oscillatory and symmetric nature of a rectangular cylinder in free flow, at a Reynolds number of 1.12 x 10^7.

Figures 5.10 and 5.11 show the average velocity fields. By comparing these figures, it becomes apparent the difference in the asymmetric wake versus the symmetric wake. Figure 5.10 faintly features the high velocity jets that were highlighted in Figures 5.6b and 5.7b. From a qualitative standpoint, flow around a rectangular cylinder near a plane wall boundary is asymmetric, whereas free flow around a rectangular cylinder is symmetric.



(a) Re = 1.94 x 10^6 **(b)** Re = 1.12 x 10^7

Figure 5.10: The asymmetric average velocity field for flow around a rectangular cylinder near a plane wall boundary, at Reynolds numbers (a) 1.94 x 10^6 and (b) 1.12 x 10^7.

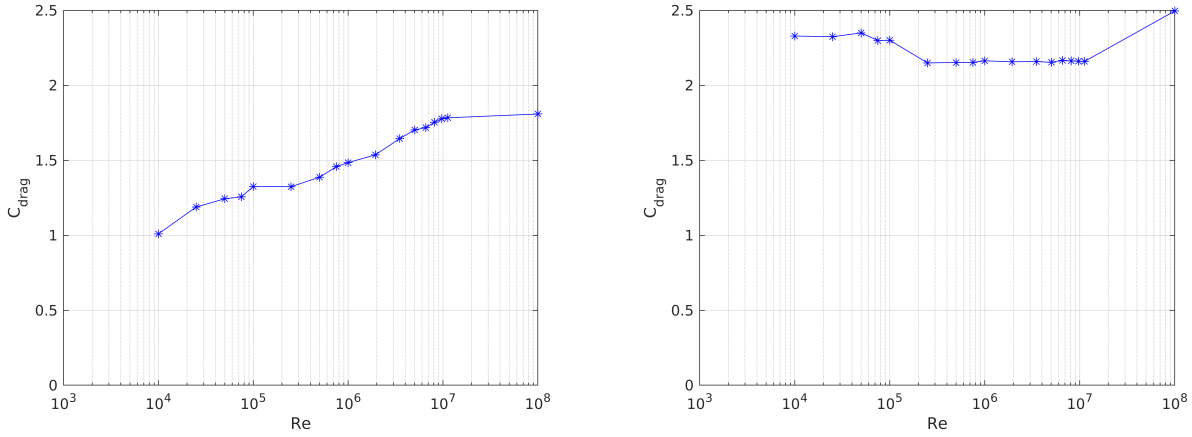


(a) Re = 1.94 x 10^6 **(b)** Re = 1.12 x 10^7

Figure 5.11: The symmetric average velocity field for a rectangular cylinder in free flow, at Reynolds numbers (a) 1.94 x 10^6 and (b) 1.12 x 10^7.

5.3.2 Aerodynamic Coefficients

The primary focus of this study is to show that the characteristics of flow around a rectangular cylinder near a plane wall boundary are dependent on Reynolds number, up to at least 1.12 x 10^7. Figure 5.12 shows C_{drag} on the rectangular cylinder for (a) flow near a plane wall boundary and (b) free flow. It is apparent that the flow around a rectangular cylinder near a plane wall boundary is dependent on Reynolds number, as suggested by Equation (5.1).



(a) Near a plane wall boundary

(b) In free flow

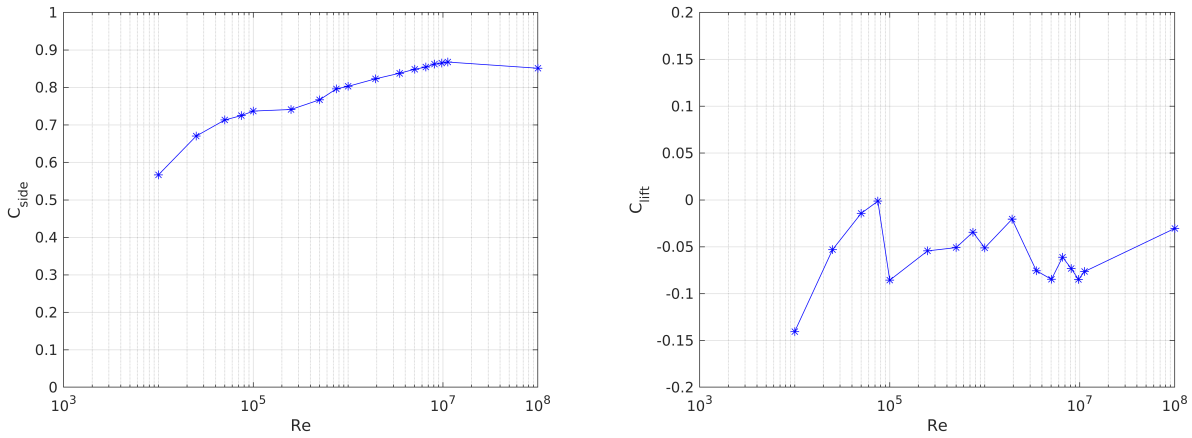
Figure 5.12: C_{drag} measured for flow around a rectangular cylinder (a) near a plane wall boundary and (b) in free flow.

Figure 5.12a shows strong Reynolds number dependence for all Reynolds numbers less than or equal to 1.12 x 10^7. For the single analysis at a Reynolds number of 10^8, C_{drag} appears to become Reynolds number independent as Reynolds number approaches 10^8, however, further simulations in this range are required to validate the Reynolds number independence observed above 1.12 x 10^7.

Figure 5.12b is corroborated against traditional C_{drag} literature (Blevins, 1984, Fig. 10-22) which shows C_{drag} for a circular cylinder is independent of Reynolds number above approximately 1000, but goes through a drag crisis around 5 x 10^5 and recovers after 10^6. The significant difference is that the asymptotic C_{drag} values are not the same, at an average of 2.22 for the rectan-

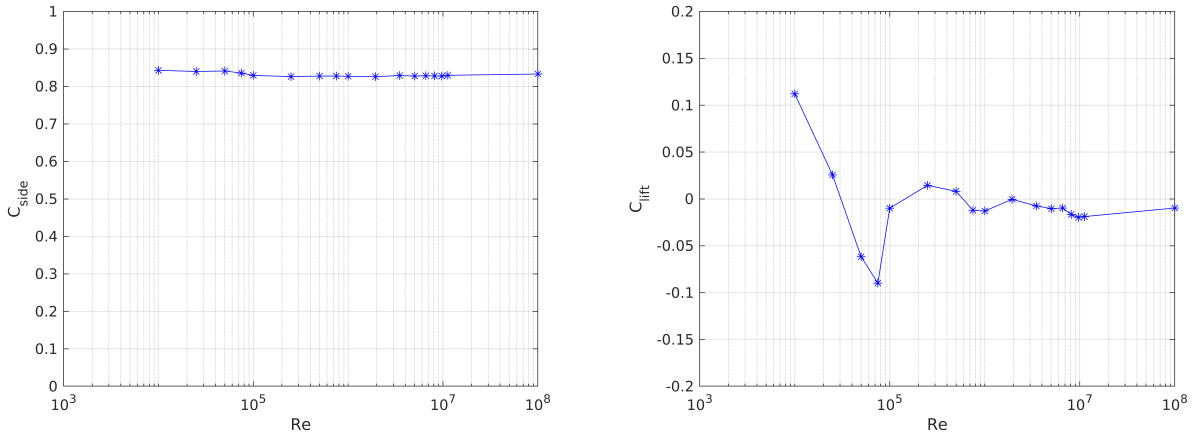
gular cylinder and approximately 1.2 for the circular cylinder. Even though C_{drag} is known to be dependent on the shape of an object (Blevins, 1984), note that the asymptotic behavior is present for both cylinder shapes in free flow.

Figures 5.13 and 5.14 show the variation of the side coefficient (C_{side}) and the lift coefficient (C_{lift}) with respect to Reynolds number. Reynolds number dependence is exhibited in C_{side} for flow around a rectangular cylinder near a plane wall boundary as seen in Figure 5.13a. C_{lift} does not seem to show dependency on Reynolds number in either model, as shown in Figures 5.13b and 5.14b. Figure 5.14a shows clear Reynolds number independence for a rectangular cylinder in free flow.



(a) C_{side} **(b)** C_{lift}

Figure 5.13: Variation of side and C_{lift} for flow around a rectangular cylinder near a plane wall boundary.



(a) C_{side} **(b)** C_{lift}

Figure 5.14: Variation of side and C_{lift} for a rectangular cylinder in free flow.

5.3.3 Application to Overturning High Sided Vehicles

Figure 5.15 depicts $C_{rolling}$ which is pertinent to the application of overturning high sided vehicles. Figure 5.15 also demonstrates that the Reynolds number dependency carries into $C_{rolling}$, which supports Equation (5.1) and highlights errors in previous Reynolds number dependency evaluations for analyses of overturning high sided vehicles (Baker, 1991; Coleman and Baker, 1994; Baker and Humphreys, 1996).

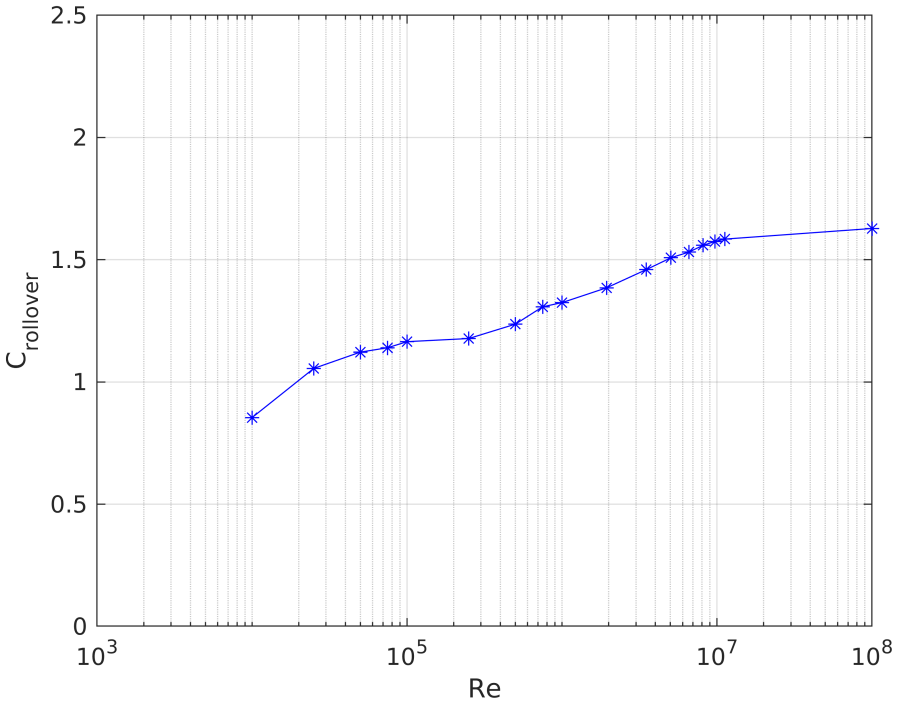


Figure 5.15: $C_{rolling}$ for flow around a rectangular cylinder near a plane wall boundary is shown varying with respect to Reynolds number.

The results presented in Figure 5.1 have been evaluated at Reynolds numbers ranging from 8.5 x 10^4 to 1.25 x 10^5. Figure 5.15 illustrates that $C_{rolling}$ is dependent on Reynolds number up to at least 1.12 x 10^7, which explains a portion of the data spread present in Figure 5.1. Additionally, Figure 5.15 highlights that Figure 5.1 is underpredicting the overturning values because the Reynolds numbers of 8.5 x 10^4 to 1.25 x 10^5 correlate to a full-size wind velocity of 0.44 m/s (0.981 mph) to 0.64 m/s (1.442 mph).

Overturning accidents are more likely to happen at Reynolds numbers that are about 20 to 110 times larger than the preceding Reynolds numbers. Realistic wind conditions reflect a full-size wind speed of 10 m/s (22.4 mph) to 58 m/s (129.7 mph) which correlate to Reynolds numbers between 1.94 x 10^6 to 1.12 x 10^7. Quantitatively, $C_{rolling}$ for a Reynolds number of 100,000 is 1.1655 and is underpredicted by 19% (with respect to Re = 1.94 x 10^6, where $C_{rolling}$ = 1.385) and by 36% (with respect to Re = 1.12 x 10^7, where $C_{rolling}$ = 1.584). Due to the Reynolds number dependency, the low Reynolds numbers that were previously used and the resulting underprediction of $C_{rolling}$, this supports the need for additional studies to update the research associated with overturning high sided vehicles to ensure that the Reynolds numbers are realistic and the resulting analysis is applicable to overturning high sided vehicles.

5.4 Conclusion

Through this study, a holistic analysis bridging two fields of study has been conducted, connecting the fundamental field of flow around rectangular cylinders near a plane wall boundary and in free flow to an application field of overturning high sided vehicles. Equation (5.1) suggested that it is premature to assume $Re_{critical}$ has been surpassed, such that the flow dynamics are Reynolds number independent.

In the fundamental field, C_{drag} for both flow conditions have been evaluated with Reynolds numbers spanning 10^4 to 10^8, filling a significant gap in the research. It has also been shown that there is a strong Reynolds number dependence for flow around a rectangular cylinder near a plane wall boundary, specifically with a length-to-height ratio of 0.887 and a gap ratio of 0.407.

In the application field, previous work on overturning high sided vehicles must be reevaluated considering the Reynolds number dependency exhibited in this analysis using a 2D rectangular cylinder (Figure 5.15). It is assumed that Reynolds number dependency carries into work investigating a 3D high sided vehicle.

Chapter 6

Conclusions and Future Work

6.1 Summary of Investigation

This thesis addressed the problem of overturning high sided vehicles through a fundamental fluid mechanics perspective. The results of this work necessitate a deeper analysis into the existing body of work on rolling moment coefficients ($C_{rolling}$) with the understanding of Reynolds number dependency.

Chapter 1 opened by providing the framework for this thesis and presented the issue of investigating Reynolds number dependency associated with overturning high sided vehicles. The literature review in Chapter 2 presented the background on overturning high sided vehicles, the aerodynamic coefficients of interest including $C_{rolling}$, the considerations for this thesis given the United States geographical focus, the background on Reynolds number dependency from an overturning high sided vehicle perspective and a fundamental fluid mechanics perspective with rectangular cylinders.

The methods used in this thesis were discussed in Chapter 3. This included the assumptions made in the analysis, the reasons for the utilized Reynolds number bounds, details on the 3D 53-GCM and 2D rectangular cylinder geometries and overall parameters used in setting up the OpenFOAM simulations.

Chapter 4 presented a comprehensive verification study that was proven necessary in addition to a traditional grid independence study. The drag coefficient (C_{drag}) was used as the basis to compare different flow domains. Due to flow around a rectangular cylinder near a plane wall boundary, it was found there are three main effects present that influence the domain sizing in each coordinate direction. An insufficient vertical domain leads to suppressed streamlines which locally accelerate the flow and artificially increases C_{drag}. An adequate horizontal domain length upstream from the cylinder allows the stagnation pressure on the rectangular cylinder to propagate

unimpeded to the flow inlet. A sufficient horizontal domain length downstream from the cylinder allows the flow wake to completely develop before exiting the outlet. Through appropriately sizing the CFD domain, a comprehensive verification study leads to accurate and useful results.

Chapter 5 discussed the Reynolds number dependence as it applied to a fundamental field in fluid mechanics involving flow around a rectangular cylinder near a plane wall boundary, and the application to overturning high sided vehicles. It was found for flow around a rectangular cylinder near a plane wall boundary with a gap ratio of 0.407, that C_{drag} is dependent on Reynolds number. This fundamental field was connected to an application field involving the overturning of high sided vehicles, with the assumption that a 2D rectangular cylinder could represent the trailer section of a high sided vehicle. It was found that traditional studies on overturning high sided vehicles assume a Reynolds number independence of the aerodynamic coefficients, whereas the fundamental field shows that there is a Reynolds number dependence, as shown in Figure 5.15.

6.2 Major Conclusions

The following provides a brief discussion on the major conclusions obtained from this thesis:

Verification and validation are fundamental requirements to performing a CFD analysis. Typically, verification is conducted using a grid independence study. It was found that for high Reynolds number incompressible flow, especially for flow with a bluff body (such as a rectangular cylinder) near a plane wall boundary, it is imperative to conduct a comprehensive verification analysis. This comprehensive verification analysis ensures that the sizing of the CFD domain is independent of the results.

In the fundamental field of fluid mechanics, a gap in the literature has been filled for C_{drag} values for flow around a rectangular cylinder near a plane wall boundary, with a gap ratio of 0.407. These C_{drag} values have been provided for Reynolds numbers from 10^4 to 10^8. It has also been shown that the flow field is asymmetric due to the plane wall boundary.

In the application field of overturning high sided vehicles, Equation (5.1) suggested that it is premature to assume $Re_{critical}$ has been surpassed, such that $C_{rolling}$ is Reynolds number indepen-

dent. Figure 5.15 shows Reynolds number dependence for $C_{rolling}$ for flow around a rectangular cylinder near a plane wall boundary, up to at least Re = 1.12 x 10^7. Previous work on overturning high sided vehicles assumed Reynolds number independence, due to $Re > Re_{critical}$. Assuming the $C_{rolling}$ Reynolds number dependence shown in Figure 5.15 carries into work investigating a 3D high sided vehicle, previous work on overturning high sided vehicles must be reframed considering the $C_{rolling}$ Reynolds number dependence shown through this research.

6.3 Suggestions for Further Research

Regarding the fundamental field of fluid mechanics, there is a significant amount of research that could be conducted. It would be useful to determine the maximum gap ratio for the flow characteristics to become independent of Reynolds number, such that the flow is not explicitly free flow, but is not influenced by the plane wall boundary and acts like a rectangular cylinder in free flow. On the other hand, it would be beneficial to reverse the analysis and investigate how a gap ratio approaching zero influences the Reynolds dependency. This would be beneficial as it would support a discussion of the results discussed in Baker and Gawthorpe (1983) which are one of the foundations for the Reynolds number independence argument used in the various analyses presented in Baker and Soper (2022). Additionally, corroborating the results of this thesis with experimental studies would be interesting and useful.

In the application field, it is apparent that additional work on determining $C_{rolling}$ is needed because there is a Reynolds number dependency evident in the simplified 2D analysis of flow around a rectangular cylinder near a plane wall boundary. This work may include the following:

1. Because it was found that the Reynolds number influences $C_{rolling}$, it would be prudent to evaluate the range of Reynolds numbers that are probable to cause overturning accidents so that the focus of the analysis can be refined.

2. It would be beneficial to investigate the connection between the Reynolds number dependency evident in the simplified trailer section and the application of Reynolds number dependency to 3D high sided vehicles. It is possible that the Reynolds number dependence is

present in the trailer section, but not in the tractor section. This question must be specifically analyzed to determine the extent of the Reynolds number dependence exhibited by the simplified trailer section. This could be accomplished by emulating the tractor section of a high sided vehicle as a 2D rectangular cylinder which is closer to the ground surface and larger in size compared with the rectangular cylinder used in this study. Additionally, further simplifying the 53-GCM could bridge the study between fundamental fluid mechanics and the application to overturning high sided vehicles. These simplifications could include: a) removing the trailer wheel base, b) removing the tractor wheel base, and c) simplifying the tractor section to be a uniform construction in the crosswind direction. Each of these simplifications would help illuminate the Reynolds number dependency effect on the entire geometry of the high sided vehicle.

3. It could be useful to conduct CFD analyses at realistic overturning Reynolds numbers. If this is unfeasible due to the high computational cost of a 3D simulation (given the comprehensive verification analysis has been completed), it would be interesting to conduct a wind tunnel study at realistic overturning Reynolds numbers. A parametric CFD investigation on various high sided vehicle types would help elucidate more of the variation present in Figure 1.2/5.1. A preliminary 3D study has been completed and is presented in Appendix A.

4. Because of the Reynolds number dependency, it would be beneficial to normalize $C_{rolling}$ with respect to Reynolds number.

Bibliography

Baker, C. A simplified analysis of various types of wind-induced road vehicle accidents. *Journal of Wind Engineering and Industrial Aerodynamics*, 22(1):69–85, 1986. doi: 10.1016/0167-6105(86)90012-7.

Baker, C. Ground vehicles in high cross winds part I: Steady aerodynamic forces. *Journal of Fluids and Structures*, 5(1):69–90, 1991. doi: 10.1016/0889-9746(91)80012-3.

Baker, C. and Gawthorpe, R. G. Aerodynamics of transportation II, the effect of turbulence simulation on the wind-induced loads on ground vehicles. *The Fluids Engineering Division, ASME*, 7:1–10, Nov. 1983.

Baker, C. and Humphreys, N. Assessment of the adequacy of various wind tunnel techniques to obtain aerodynamic data for ground vehicles in cross winds. *Journal of Wind Engineering and Industrial Aerodynamics*, 60:49–68, 1996. doi: 10.1016/0167-6105(96)00023-2. The Wind Engineering Society's 2nd UK Conference.

Baker, C. and Soper, D. Calculation of the overturning wind speed of large road vehicles at exposed sites. *Proceedings of the Institution of Civil Engineers - Transport*, 175(2):97–104, 2022. doi: 10.1680/jtran.18.00102.

Bearman, P. W. and Zdravkovich, M. M. Flow around a circular cylinder near a plane boundary. *Journal of Fluid Mechanics*, 89(1):33–47, 1978. doi: 10.1017/S002211207800244X.

Beedie, D. CDOT stands behind timing of high-wind restriction despite 15 semi-truck wrecks in El Paso County. KRDO. Web, Dec. 2021.

Bhattacharyya, S. and Maiti, D. K. Shear flow past a square cylinder near a wall. *International Journal of Engineering Science*, 42:2119–2134, 11 2004. doi: 10.1016/J.IJENGSCI.2004.04.007.

Blevins, R. D. *Applied fluid dynamics handbook*. Van Nostrand Reinhold Co., New York, N.Y, 1984. ISBN 0442212968.

Browand, F., McCallen, R., and Ross, J. *The Aerodynamics of Heavy Vehicles II: Trucks, Buses, and Trains*, chapter European Truck Aerodynamics – A Comparison Between Conventional and CoE Truck Aerodynamics and a Look into Future Trends and Possibilities. Springer, 2009.

Cheli, F., Corradi, R., Sabbioni, E., and Tomasini, G. Wind tunnel tests on heavy road vehicles: Cross wind induced loads—part 1. *Journal of Wind Engineering and Industrial Aerodynamics*, 99(10):1000–1010, 2011. doi: 10.1016/j.jweia.2011.07.009.

Cheng, M., Whyte, D. S., and Lou, J. Numerical simulation of flow around a square cylinder in uniform-shear flow. *Journal of Fluids and Structures*, 23:207–226, 2 2007. doi: 10.1016/J.JFLUIDSTRUCTS.2006.08.011.

Choi, H., Lee, J., and Park, H. Aerodynamics of heavy vehicles. *Annual Review of Fluid Mechanics*, 46(1):441–468, 2014. doi: 10.1146/annurev-fluid-011212-140616.

Coleman, S. and Baker, C. High sided road vehicles in cross winds. *Journal of Wind Engineering and Industrial Aerodynamics*, 36:1383–1392, 1990. doi: 10.1016/0167-6105(90)90134-X.

Coleman, S. and Baker, C. An experimental study of the aerodynamic behaviour of high sided lorries in cross winds. *Journal of Wind Engineering and Industrial Aerodynamics*, 53(3):401–429, 1994. doi: 10.1016/0167-6105(94)90093-0.

Coleman, S. A. *The Aerodynamics of Ground Vehicles in Cross Winds*. PhD thesis, The University of Nottingham, 1990.

Courchesne, J. and Laneville, A. A comparison of correction methods used in the evaluation of drag coefficient measurements for two-dimensional rectangular cylinders. *Journal of Fluids Engineering*, 101(4):506–510, 1979. doi: 10.1115/1.3449019.

Crank, J. and Nicolson, P. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(1):50–67, 1947. doi: 10.1017/S0305004100023197.

Croll, R. H., Gutierrez, W. T., Hassan, B., Suazo, J. E., and Riggins, A. J. Experimental investigation of the ground transportation systems (GTS) project for heavy vehicle drag reduction. *US DOE Office of Scientific and Technical Information*, 12 1995.

Fleet Speak. The difference between HGV, LGV, rigid, artic and road train: What's best for the job? Web, Apr. 2021.

Forouzi Feshalami, B., He, S., Scarano, F., Gan, L., and Morton, C. A review of experiments on stationary bluff body wakes. *Physics of Fluids*, 34(1):011301, 2022. doi: 10.1063/5.0077323.

Gerhart, P. M., Gerhart, A. L., and Hockstein, J. I. *Munson, Young and Okiishi's Fundamentals of Fluid Mechanics*. John Wiley & Sons Inc., Hoboken, NJ, 8th edition, 2016.

Greenshields, C. *OpenFOAM v9 User Guide*. The OpenFOAM Foundation, London, UK, 2021.

Greenshields, C. OpenFOAM v10 User Guide - 4.5 Numerical schemes. CFD Direct Ltd. Web, Jul. 2022.

Grm, A. and Batista, M. Vehicle aerodynamic stability analysis under high crosswinds. *Strojniski Vestnik-journal of Mechanical Engineering*, 63:191–200, 2017.

Hale, R. History of trucking and transportation. TruckersLogic. Web, Nov. 2015.

Hoerner, S. F. *Fluid-dynamic drag; practical information on aerodynamic drag and hydrodynamic resistance*. Published by the author, Midland Park, N.J, 1965.

Issa, R. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62(1):40–65, 1986. doi: 10.1016/0021-9991(86)90099-9.

Kundu, P. K., Cohen, I. M., and Dowling, D. R. *Fluid Mechanics*. Academic Press, Boston, MA, 6th edition, 2016. ISBN 978-0-12-405935-1. doi: 10.1016/B978-0-12-405935-1.01001-7.

Macchesney, Z. Generic conventional model (GCM). GRABCAD. Web, Jun. 2016.

Mahir, N. Three-dimensional flow around a square cylinder near a wall. *Ocean Engineering*, 36: 357–367, 4 2009. doi: 10.1016/J.OCEANENG.2009.01.002.

Menter, F., Kuntz, M., and Langtry, R. Ten years of industrial experience with the SST turbulence model. *Heat and Mass Transfer*, 4, 01 2003.

Moin, P. *Fundamentals of Engineering Numerical Analysis*. Cambridge, 2nd edition, 2010.

NWS Boulder. High winds and marshall fire on December 30th, 2021. National Weather Service. Web, Dec. 2021.

NWS Boulder. Wind advisory and high wind warning/watch guidelines. Private communication, Nov. 2022.

NWS Boulder. Douglas county tornado June 22, 2023. National Weather Service. Web, Jun. 2023.

NWS Cheyenne. High wind warning/watch guidelines. Private communication, Jul. 2023.

NWS Pueblo. Extreme wind event. National Weather Service. Web, Dec. 2021.

Pope, S. B. *Turbulent flows*. Cambridge University Press, Cambridge, 2000. ISBN 0521591252. doi: 10.1017/CBO9780511840531.

Rae, W. H., Barlow, J., and Pope, A. *Low Speed Wind Tunnel Testing*. John Wiley & Sons, Inc, New York, NY, 3rd edition, 1999. ISBN 0471557749.

Robertson, E., Choudhury, V., Bhushan, S., and Walters, D. Validation of OpenFOAM numerical methods and turbulence models for incompressible bluff body flows. *Computers & Fluids*, 123: 122–145, 2015. doi: 10.1016/j.compfluid.2015.09.010.

Roshko, A., Steinolfson, A., and Chattoorgoon. Flow forces on a cylinder near a wall or near another cylinder. In *Second US National Conference on Wind Engineering Research*, pages IV–15, Jun. 1975.

Salati, L., Schito, P., Rocchi, D., and Sabbioni, E. Aerodynamic study on a heavy truck passing by a bridge pylon under crosswinds using CFD. *Journal of Bridge Engineering*, 23(9):04018065, 2018. doi: 10.1061/(ASCE)BE.1943-5592.0001277.

Sanchez, D. K. Investigating overturning high sided vehicles through modeling high reynolds number incompressible flow around a rectangular cylinder near a plane wall boundary. Colorado State University, 2023.

Sanchez, D. K. and Venayagamoorthy, S. K. A comprehensive verification study for computational modeling of high reynolds number incompressible flow around a rectangular cylinder near a plane wall boundary. Submitted for publication to the Engineering Reports journal, published by John Wiley & Sons Ltd, 2023.

Shotts, W. E. *The Linux Command Line: A Complete Introduction*. No Starch Press, USA, 2012. ISBN 9781593273897.

Sterling, M., Quinn, A., Hargreaves, D., Cheli, F., Sabbioni, E., Tomasini, G., Delaunay, D., Baker, C., and Morvan, H. A comparison of different methods to evaluate the wind induced forces on a high sided lorry. *Journal of Wind Engineering and Industrial Aerodynamics*, 98(1):10–20, 2010. doi: 10.1016/j.jweia.2009.08.008.

Storms, B. L., Satran, D. R., Heineck, J. T., and Walker, S. M. A summary of the experimental results for a generic tractor-trailer in the ames research center 7- by 10-foot and 12-foot wind tunnels. Technical report, NASA, 2006.

The U.S. National Archives and Records Administration. National interstate and defense highways act (1956). Web, Feb. 2022.

The Weather Network. Extreme alberta winds causes semi truck to tip over. YouTube, Jan. 2021.

Truck School Swindon. What are the different types of lorries? Web, Jan. 2021.

Tunay, T., Drugge, L., and O'Reilly, C. J. On coupling methods used to simulate the dynamic characteristics of heavy ground vehicles subjected to crosswind. *Journal of Wind Engineering and Industrial Aerodynamics*, 201:104194, 2020. doi: 10.1016/j.jweia.2020.104194.

UK Government. Maximum length of vehicles used in Great Britain. UK Government. Web, Oct. 2017.

US Department of Transportation. Federal size regulations for commercial motor vehicles. US Department of Transportation, Federal Highway Administration, Office of Freight Management and Operations. Web, Oct. 2004.

US Department of Transportation, Federal Highway Administration, Policy and Governmental Affairs, Office of Highway Policy Information. Traffic monitoring guide. US Department of Transportation, Federal Highway Administration, Policy and Governmental Affairs, Office of Highway Policy Information. Web, Nov. 2014.

Versteeg, H. K. and Malalasekera, W. *An introduction to computational fluid dynamics : the finite volume method*. Pearson Education Ltd., Harlow, England, 2nd edition, 2007. ISBN 9780131274983.

Weller, H. G., Tabor, G., Jasak, H., and Fureby, C. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12(6):620–631, 1998. doi: 10.1063/1.168744.

Wolf Dynamics. Turbulence modeling in OpenFOAM: Theory and applications. Web, Jul. 2021a.

Wolf Dynamics. Tips and tricks in OpenFOAM. Web, n.d.b.

Yang, F., Zhou, Z., Tang, G., and Lu, L. Steady flow around a square cylinder near a plane boundary. *Ocean Engineering*, 222:108599, 2 2021. doi: 10.1016/J.OCEANENG.2021.108599.

Yang, F., Peng, B., Liu, M., and Jin, X. Large eddy simulation on a square cylinder near a plane boundary. *Ocean Engineering*, 245:109953, 2 2022. doi: 10.1016/J.OCEANENG.2021.109953.

Zdravkovich, M. Forces on a circular cylinder near a plane wall. *Applied Ocean Research*, 7(4): 197–201, 1985. doi: 10.1016/0141-1187(85)90026-4.

Zhan, J.-m., Li, Y.-t., Wai, W.-h. O., and Hu, W.-q. Comparison between the Q criterion and Rortex in the application of an in-stream structure. *Physics of Fluids*, 31, 2019. doi: 10.1063/1.5124245.

Zhang, Q., Su, C., Zhou, Y., Zhang, C., Ding, J., and Wang, Y. Numerical investigation on handling stability of a heavy tractor semi-trailer under crosswind. *Applied Sciences*, 10(11), 2020. doi: 10.3390/app10113672.

Appendix A

Preliminary insights into a 3D analysis of overturning high sided vehicles

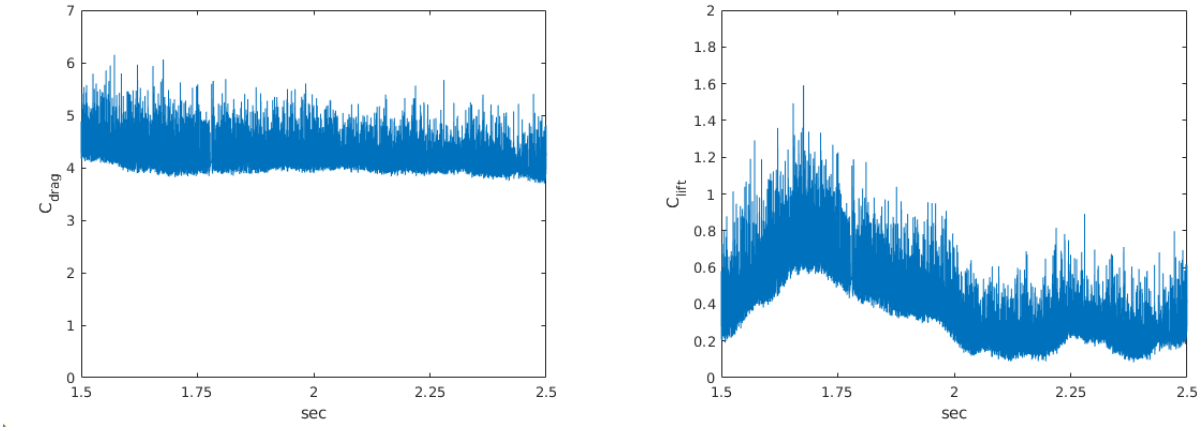
The focus of this thesis was on a simplified 2D analysis; however it was understood that a 3D analysis would directly connect to the application of overturning high sided vehicles. A preliminary analysis was conducted using the 53-GCM model. The purpose of this chapter is to detail the parameters for this preliminary analysis, the preliminary results, and some general guidelines for a future analysis with 3D geometry.

To set up the model, the following parameters were used. Because of the complexity of implementing a developed flow profile for a 3D geometry using the methodology conducted in Chapter 4, it was determined that this preliminary analysis would use a uniform inlet velocity profile. Using the results of the comprehensive verification analysis, the results of the Chapter 4 2D analysis were applied as the domain bounds (12m domain height, 10m length upstream from the center axis of the 53-GCM model, 7m length downstream from the center axis of the 53-GCM model), and preliminary bounds were specified in the length axis of 6m on either side of the high sided vehicle. The predominate cell size was 0.05m, with a refinement region using a cell size of 0.025m located in the region below the bottom of the trailer and the ground surface. This refinement region was included because it was found during the grid independence study (Chapter 4), that the flow oscillations dampen out with a low level of refinement. Six inflation layers were specified for the surface of the 53-GCM model, and ten inflation layers were specified for the ground surface. The resulting mesh was around 25 million cells.

Based on guidance, it is recommended that 1 GB of RAM allocated per 1 million cells (https://www.cfd-online.com/Forums/openfoam-solving/60879-memory-requirements-tools.html). Since the computer has 252 GB of RAM, 25 million cells equate to a usage of about 10% of the

RAM capacity. Since this is a preliminary analysis, it was determined that this was reasonable to generate preliminary results.

A preliminary analysis of the data shows the oscillatory behavior is moderately present in the 3D model, as seen in Figure A.1 for a Reynolds number of 1.94 x 10^6. The variations are less magnified than they were in the 2D simulations, but they are moderately evident.



(a) C_{drag} **(b)** C_{lift}

Figure A.1: Preliminary time series of aerodynamic coefficients plotted for 53-GCM model, Re = 1.94 x 10^6.

After a significant amount of computational time had completed, it was evident that optimization must be implemented. Using 50 cores, the computer requires approximately 200 hours to generate 1 model second. This is a substantial increase in the computational requirements compared to the 2D results. At this rate, it will take several months to generate the same type of results analyzed in Chapters 4 and 5.

One area for optimization is in the mesh construction. As described in Appendix B.2.2, it is imperative to use snappyHexMesh, especially for 3D geometries. The 2D meshing strategy used in Chapter 4 is not optimized for 3D geometries. It is important to note that the cell size used in this preliminary analysis is 5 times larger than in the 2D simulations. During the grid independence study in Chapter 4, it was found that for a cell size of 0.05m, the oscillations in the system disappear completely. This reiterates the need to develop a robust meshing methodology that is appropriate

for 3D geometries, such that the cell size immediately around and near the 53-GCM is 0.01m. It would be advantageous to determine the optimum mesh construction using snappyHexMesh to reduce the number of required cells while retaining the mesh refinement near the high sided vehicle to capture the wake dynamics. To accelerate this analysis, a pre-initialization of the flow domain would accelerate the parametric study of different mesh configurations.

Through these preliminary insights into a 3D analysis on high sided vehicles using the 53-GCM geometry, it was found that the 2D mesh construction strategy presented in Chapter 4 and discussed in Appendix B.2.2 is not transferable or optimized for a 3D mesh. Additional work must be completed to optimize the mesh construction so that adequate results may be generated and evaluated.

Appendix B

Best Practices

Throughout the course of this thesis, a plethora of best practices have been accumulated through online research, personal exploration, and trial and error. This appendix describes some of these best practices as it pertains to using OpenFOAM and the general workflow. Additional details are provided in Appendix C, expounding on the content presented here as well as providing new content that is not discussed here. These best practices are the result of the author's learning experience and are not meant to be authoritative as the best solution for a specific problem.

B.1 Before Diving into OpenFOAM

OpenFOAM was used as the CFD modeling software for this thesis. However, in order to learn OpenFOAM, a significant amount of upfront work was invested prior to diving into the opensource CFD software. Three graduate level courses and one book study were foundational to understanding the theory behind CFD: Numerical methods (Moin, 2010) the theory and application of CFD (Versteeg and Malalasekera, 2007), graduate level fluid mechanics (Kundu et al., 2016) and turbulent flows (Pope, 2000). After this foundation was laid, one semester was spent going through an introduction to Linux basics (Shotts, 2012). Finally, one more semester was spent integrating the knowledge basics of Linux and C coding with OpenFOAM which included learning mesh development, running a model and post processing the model.

B.2 Using OpenFOAM

B.2.1 OpenFOAM History

The development of OpenFOAM is justifiably confusing to a new user. There are 2 primary publishers of the OpenFOAM software: OpenFOAM.org (who releases software with incremental version increases; this thesis was created using this software with version 9; as of this writing,

the OpenFOAM Foundation has released up to version 11) and OpenFOAM.com (who releases software twice a year, with the format of vYYMM). These separate companies have the same root history, which is complicated but can explain on the surface level why there are two different versions of OpenFOAM (https://www.cfd-online.com/Forums/openfoam/152605-original-openfoam-author-s.html). For the purpose of this thesis, it was found beneficial to install both types of OpenFOAM as they each have the same base capabilities, but also have individual benefits. However, for consistency it is recommended that the results be obtained from one version or the other.

B.2.2 Meshing Takeaways

The CFD mesh is of the utmost importance. It is reasonable to generate a preliminary mesh and then run a model and evaluate the results, but this model and results are also preliminary. As the understanding of mesh development and evaluation grows, the mesh must be refined and improved. With respect to OpenFOAM, there are many types of mesh software available. Through this thesis, it was learned that cfMesh is a helpful tool for developing a preliminary mesh and expediting the learning curve for running a model and post processing the data. However, cfMesh is not adequate from a foundational standpoint, especially when it comes to boundary refinement for complex 2D geometries and 3D geometries in general. From evaluating various options, it appears that snappyHexMesh is the best software for generating foundationally resilient meshes, even though snappyHexMesh has a steeper learning curve compared to cfMesh. **The use of snappyHexMesh is imperative with 3D geometries.**

B.2.3 Simulations in Parallel, with mpi and numactl

When running simulations, there are a handful of best practices that are helpful when expediting the simulations. The software “mpi” is beneficial to allow the OpenFOAM simulation to be run on multiple processors. If mpi is not invoked, the simulation will be run on a single core. In addition to mpi, some computer hardware systems are designed with hardware separation within the CPU cores. By running the command “`numactl --hardware`", the user can find out the

construction of the CPU cores and how many "numa" there are in the system (https://www.cfd-online.com/Forums/hardware/100260-mpirun-best-parameters.html#post356954). The benefit of investigating the construction of the system's numa is that the simulations can be isolated to run on a specific numa, thereby isolating each simulation and increasing the efficiency. In the author's case, 4 separate computers were used throughout the course of this thesis. One custom-built computer (Feynman) had 2 numa, whereas the remaining three computers had 1 numa each. By following the directions in the previous reference using "`mpirun --bind-to none -np $numberOfProcessors numactl --cpunodebind=$numa FOAMapplication -parallel`", 2 simulations were continually run on Feynman since they were isolated from one another.

For example, a benchmark simulation on flow around a rectangular cylinder near a plane wall boundary was run on 14 processors and took 1.18 hours to complete 1 model second. 14 processors were selected as it was determined the optimum number of processors to run this type and size of simulation. This same benchmark simulation was run as two identical and simultaneous simulations in parallel, with 14 processors each. These simultaneous simulations were isolated to the same numa (which has 28 processors allocated to the single numa). The time required to run the simulations in parallel was 2.44 hours. If these two simulations had been run in series, they would have completed in 2.36 hours, a 3.2% time savings compared to the simulations run in parallel. This time savings is magnified over the course of long duration simulations.

In addition to isolating a single simulation within a specific numa and running simulations in series, it is important to perform processor optimizing tests. It was found that there seemed to be an optimum number of cells per processor that expedited the simulation. It was found that the simple decomposition method is the most effective. It is possible that other decomposition methods may be better for certain situations, but the simple method is a good starting point (https://www.cfd-online.com/Forums/openfoam/119612-scotch-ptscotch.html). Processor optimization tests were conducted by copying the base simulation that was going to be analyzed, changing the end time to a short and reasonable value, and copying this edited simulation into several sub simulations with

varying processor configurations. Since the simple decomposition method was used, the varying processor configurations typically involved two subdivisions in the Y axis, and 4-9 subdivisions in the X axis. In this thesis, it was normal for 14 processors to be used for larger 2D simulations (with a cell count ranging between 1.9 and 12 million cells), 12 processors used for smaller 2D simulations (with 0.4 million cells), and 50 processors used for the 3D 53-GCM simulations (with 24 million cells). 56 processors could have been specified for the 3D 53-GCM simulations, but it was determined that it was beneficial to leave a few processors unallocated so that other functions could run on the computer during the simulation.

When conducting multiple simulations in a single "series" of simulations, it is imperative to understand that the results are dependent on the number of processors used. For example, when comparing the bulk C_{drag} measured in the same simulation run on the same computer with 14 processors versus 3 processors, the result differs by about 0.806%. This difference is negligible but must be acknowledged. This analysis can be extended to running simulations on multiple computers. By running the same simulation on 3 cores on three different computers, the results are identical. Based on this analysis, it was recommended that simulations of the same "series" be run with the same number of cores to eliminate differences inherent in the processor distribution.

B.2.4 Control Dictionary Considerations

As previously described, the PIMPLE method was used in this analysis. The PIMPLE method has a built in dynamic CFL method which works primarily based on a provided maximum CFL number in the controlDict file ("maxCo") but is initialized by the ΔT variable. ΔT is then dynamically adjusted based on the condition of the simulation and within the constraints of the maximum CFL number. However, it is imperative that the initial ΔT is set below the required simulation ΔT, otherwise the simulation will not converge. For this thesis, the initial ΔT was set to a value of 1^{-10}, which was typically 10^4 to 10^6 orders of magnitude smaller than the average ΔT.

The writeInterval entry of the controlDict file was discovered to influence the speed at which a simulation was completed. If the writeInterval was significantly low, 10^{-4} for example, ΔT would

be artificially slowed down to allow the simulation to march forward in time to hit every requested writeInterval. This issue only occurred in simulations with the highest and lowest Reynolds numbers. An alternative to this writeInterval constriction is to set the writeInterval high enough, such as 0.1, and once the simulation completes at the desired endTime, the simulation can be restarted with a lower writeInterval for a short duration to allow data to be collected for visualization.

B.2.5 Evaluating Simulation Progress

It is important to have means to check the initial start of a simulation, and then periodically check the simulation progress. Both can be visually evaluated by graphically looking at the residual convergence, the progression of ΔT and CFL number, and selected post processing values. Code Listing B.1 shows the codes used. These codes were typically run near the beginning of the simulation to evaluate the startup behavior, and then throughout the run to ensure that the simulation is behaving as expected.

Code Listing B.1: Code used to generate graphs of a) residuals, b) deltaT c) CFL number, and d) selected post processing values. Code executed with gnuplot software. Code adapted from https://www.cfdsupport.com/OpenFOAM-Training-by-CFD-Support/node145.html

```
set title "Convergence Progress - Residuals"
set xlabel "Iterations"; set ylabel "Residuals"
set logscale y
plot    \
    "< cat log.pimpleFoam | grep Ux          | cut -d' ' -f9 | tr
       -d ','" title 'u', \
    "< cat log.pimpleFoam | grep Uy          | cut -d' ' -f9 | tr
       -d ','" title 'v', \
    "< cat log.pimpleFoam | grep Uz          | cut -d' ' -f9 | tr
       -d ','" title 'w', \
    "< cat log.pimpleFoam | grep omega       | cut -d' ' -f9 | tr
       -d ','" title 'omega', \
    "< cat log.pimpleFoam | grep 'for k'     | cut -d' ' -f9 | tr
       -d ','" title 'k'
pause 4 # pause mouse
-----------------------------------------------------------------
set title "Convergence Progress - deltaT"
set xlabel "Iterations"; set ylabel "deltaT (sec)
plot "< cat log.pimpleFoam | grep deltaT     | cut -d' ' -f3 "
       title 'deltaT'
pause 2 # pause mouse
-----------------------------------------------------------------
set title "Convergence Progress - Courant No."
set xlabel "Iterations"; set ylabel "Courant No."
set logscale y
plot    \
    "< cat log.pimpleFoam | grep Courant          | cut -d' ' -f6
       "      title 'Courant Max', \
    "< cat log.pimpleFoam | grep Courant          | cut -d' ' -f4
       "      title 'Courant Mean'
pause 2 # pause mouse
-----------------------------------------------------------------
set title "Convergence Progress - Force Coefficients"
set xlabel "Simulated Time (sec)"; set ylabel "Force Coefficients
   (non-dimensional)"
unset logscale y
plot    \
    "< cat postProcessing/forceCoeffsIncompressible/*/forceCoeffs
       .dat" using 1:2      title 'Cm', \
    "< cat postProcessing/forceCoeffsIncompressible/*/forceCoeffs
       .dat" using 1:3      title 'Cd', \
    "< cat postProcessing/forceCoeffsIncompressible/*/forceCoeffs
       .dat" using 1:4      title 'Cl'
pause 5 # pause mouse
```

B.2.6 Purpose of Developed Flow Profile

As presented in Chapter 4, a developed flow profile was implemented into the final simulations. It became apparent that the developed flow profile is beneficial for multiple reasons when compared to a uniform flow profile. First, it provides more accurate results. This manifested itself in a test simulation with a Reynolds number of 1.61 x 10^6, where C_{drag} for the uniform flow profile was 2.1789, and C_{drag} with the developed flow profile was 1.6935 resulting in a 28.7% inflated value. This difference makes sense because the velocity in the region of the rectangular cylinder is lower compared to the uniform velocity. Because of this behavior, it is imperative to use a developed flow profile to generate more accurate results. Second, the results are generated faster due to the developed profile and a lower velocity gradient between the cells at the ground surface and the no-slip wall. The developed flow profile must first be generated which typically took about 30-50 hours per developed flow simulation on 14 cores. The final simulation with developed flow for 50 model seconds took approximately 57.3 hours to generate on 14 cores compared to 67 hours to generate for the uniform flow simulation: a 17% decrease. Since 17 simulations were run for the final dataset of flow around a rectangular cylinder near a plane wall boundary as seen in Chapter 5, the use of a developed flow profile saved approximately 165 hours while generating more accurate results. Finally, the boundary layer does not readily grow when using a developed flow inlet profile. When observing the velocity cross section of a uniform flow profile at various lengths downstream of the inlet as seen in Figure B.1, it is obvious that the velocity profile is developing. However, the same analysis for a developed flow profile shows that the velocity profile is not further developing as shown in Figure B.2.

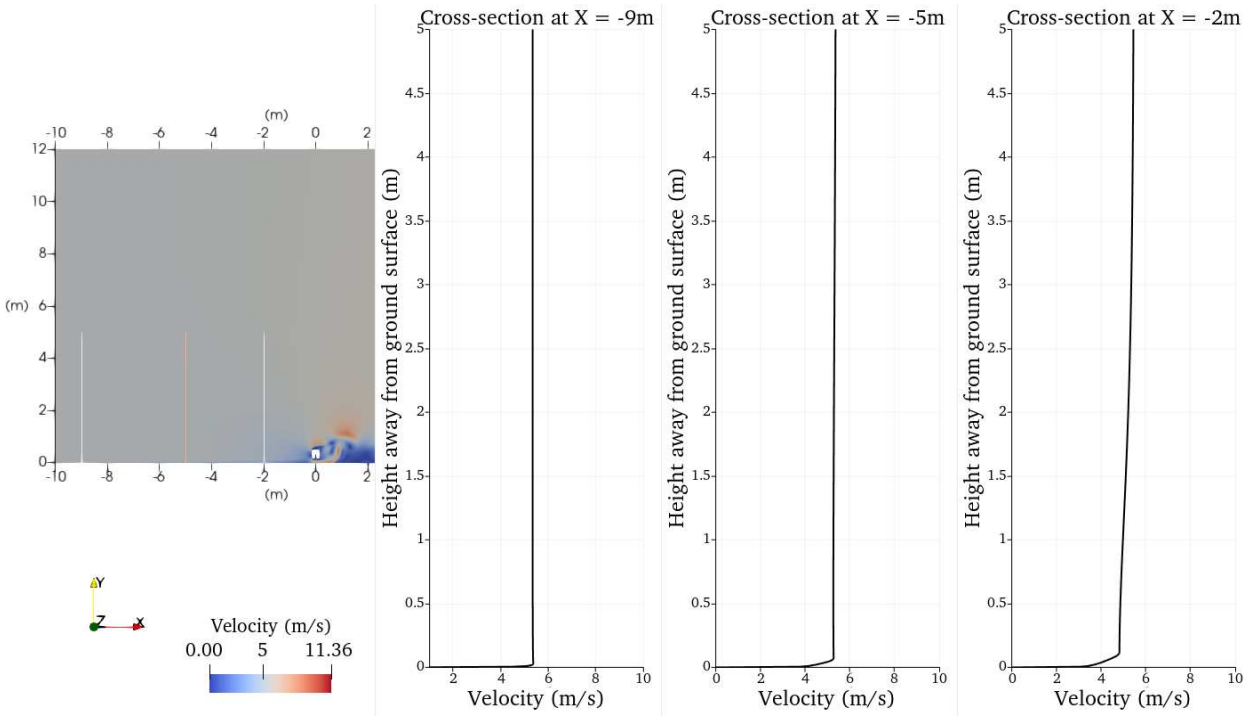


Figure B.1: Uniform inlet velocity profile is developing at 1m, 5m and 9m downstream of the inlet.

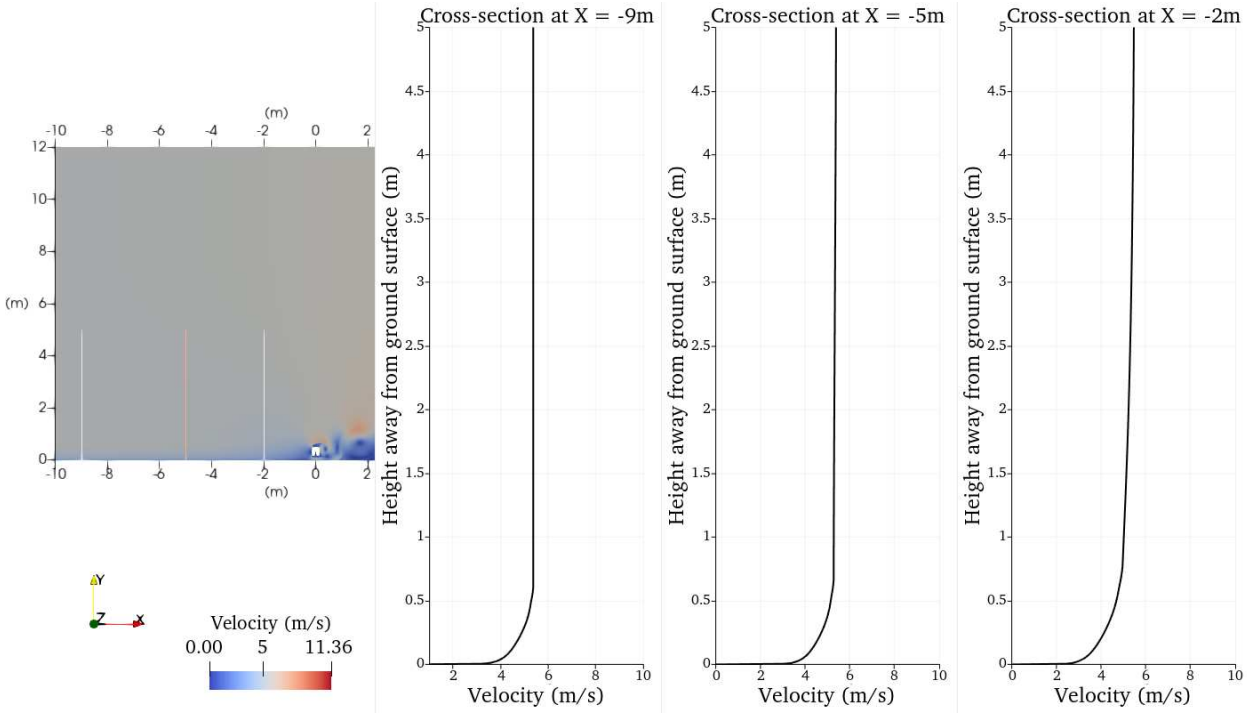


Figure B.2: Developed inlet velocity profile is not developing at 1m, 5m and 9m downstream of the inlet.

B.3 OpenFOAM Resources

There are many facets to learning how to conduct an OpenFOAM analysis, but there are also numerous resources for learning the various tools necessary and alternative methods. A shortlist of OpenFOAM resources used by the author is as follows:

- CFD Online. A general CFD forum for various CFD software, with a specific subdomain for OpenFOAM. https://www.cfd-online.com/Forums/openfoam/ This specific thread is pertinent for starting with OpenFOAM:
  https://www.cfd-online.com/Forums/openfoam/146697-getting-started-openfoam.html

- Wolf Dynamics. This publisher has created many tools and teachings. The tools are available on the website. The teachings are available on the website and as PDFs via an internet search, typically by searching for "wolf dynamics" and "item of interest" such as snappyHexMesh www.wolfdynamics.com/

- Fluid Mechanics 101. Bridges the gap between the CFD theory and the practical application to OpenFOAM (as well as ANSYS Fluent). Typically provides the sources for the material that is presented. https://www.youtube.com/channel/uCcqQi9LT0eTkRoUu8eYaEkg

- OpenFOAM in 15 days. This conglomerated course has been put together by the OpenFOAM community to facilitate learning OpenFOAM.
  https://wiki.openfoam.com/%223_weeks%22_series

- FOAMY Lectures, CFD Lectures with OpenFOAM by Dr. Cuneyt Sert. This lecture series developed in house is to facilitate the learning of OpenFOAM. It is a work in progress. https://users.metu.edu.tr/csert/foamyLectures/index.html

- OpenFOAM.com documentation has refences dynamically built into the documentation. For example, the PISO algorithm is mentioned (https://www.openfoam.com/documentation/guides/latest/doc/guide-applications-solvers-piso.html), and the reference is provided

(https://www.openfoam.com/documentation/guides/latest/doc/citelist.html#CITEREF_issa_solution_1986)

- Introduction to OpenFOAM with Prof. Hrvoje Jasak. The author could not find the source files that are referenced in the videos, so the examples and general knowledge is all that can be gleaned from these. The author would recommend watching these sometime after doing the OpenFOAM in 15 days crash course so that you have a general framework for what is going on in these videos:
  https://www.youtube.com/playlist?list=PLqxhJj6bcnY9RoIgzeF6xDh5L9bbeK3BL

The following includes additional resources that were found but not necessarily used by the author:

- OpenFOAM resources.
  https://www.cfdsupport.com/OpenFOAM-Training-by-CFD-Support/node1.html
- OpenFOAM Journal. A collection of journal articles for OpenFOAM simulations.
  https://journal.openfoam.com/index.php/ofj/index
- CFD Engine. A business, with actionable newsletters published weekly.
  https://www.cfdengine.com/
- A "cheat sheet" on OpenFOAM basics. https://seanbone.ch/openfoam-cheat-sheet/
- OpenFOAM Benchmarking. To test the speed of a computer relative to other computers.
  https://openbenchmarking.org/test/pts/openfoam

B.4 Workflow

As the OpenFOAM simulations were developed, the author created and implemented various workflow methods that were helpful with streamlining the creation of simulations. Additional details are presented in Appendix C.3.

Benchmark cases were used to evaluate the relative speed of different computers, and to test the same computer at a later point in time to determine if the computer was getting "tired" and needed to be cleaned or restarted. Since over 400 simulations were run throughout the course of this thesis, it was imperative to keep track of the different simulations. A spreadsheet was used to track each type of simulation (such as rectangular cylinder in free flow, flow around a rectangular cylinder near a plane wall boundary, or flow development), and the variations for each of these simulations (such as turbulence model, boundary definitions, discretization methods, varying velocity, lengths of the domain, number of inflation layers, or inlet flow profile type). This spreadsheet also tracked preliminary results and the time required to complete the simulation.

Because of the vast quantity of data generated, data management is paramount. To save data while a simulation was running, the writeInterval was set to at least 2 so that there was always a static set of data in the event that a simulation was abruptly ended (such as due to a power outage or the power button accidentally being clicked). In this situation, it was relatively easy to restart a simulation. Once the simulation completed, the data was backed up twice. First to a supplementary internal hard drive, and second to an external file server. These methods provided peace of mind for the security of the data.

Bash scripting was heavily used to automate the simulation processes. Approximately 59 scripts and codes were created throughout the development of the thesis to setup, run and post process each simulation. See Appendix C.5 for details on these scripts.

One script that was particularly useful is the helperRunningMPI.sh script as seen in Code Listing C.14, which was developed to preload a simulation into the background so that one simulation could run and a second simulation would be queued up. Once the mpi process on the active simulation concluded, the queued simulation would begin after a user specified delay. Another useful

script was the automatic email script status.sh, implemented with crontab (Code Listing C.62), as seen in Code Listing C.61. This script would check the system for active simulations, and then provide an email summary of dozens of parameters and calculate the estimated time of completion. In addition, this script was used to monitor the storage capacity of the computer, and to alert the user if the capacity was being reached. This script could be executed on demand and was executed at regularly specified intervals using the crontab functionality, as shown in Code Listing C.62. Both scripts allowed the author to mentally disconnect from the simulations for 1 to 3 days at a time, so that more focus could be put on other parts of the thesis and life in general.

Remote access to the computers was implemented to facilitate immediate access to the simulations. This remote access was implemented using two primary methods and a second backup method. First, SSH (Secure Shell) connections were used with KiTTY. KiTTY was preferred over PuTTY because of a bug associated with PuTTY where the CTRL key does not work (https://superuser.com/a/1533323). Finally, the software NoMachine visually displayed the remote machine, with the software x11vnc used as a backup to NoMachine. NoMachine was preferred over x11vnc because of the higher security protocol, and the ability to restart the remote machine and log back in. x11vnc only works once the user is logged in. However, using x11vnc as a backup was sometimes useful in case NoMachine was not working.

B.5 Validation and Comprehensive Verification

Validation and verification are key components to any CFD analysis. Chapter 4 details specifics on a "comprehensive verification" analysis, where the verification goes beyond the traditional grid independence study and looks wholistically at the independence of the domain. It is important to carry out the recommendations presented in Chapter 4 so that the user is confident that the simulation is accurate and the accuracy of the mesh has been evaluated.

B.6 Working Environment

Throughout the course of this thesis, many work environments were employed and it was found that each work environment was beneficial in different seasons and for different purposes. One key environment is the outdoors – whenever the author had a chance to work outside the chance was seized. This manifested itself in several ways. Early on, the software youtube-dl was used to download YouTube videos for offline use (youtube-dl has been abandoned by the developer as of 12/17/2021, but many alternatives have arisen, such as yt-dlp). Once these videos were downloaded for offline use, the author watched them at parks, on the backyard patio, and in public open spaces. Sometimes after watching a video at a public open space, the author would then go on a short run to also incorporate physical activity. Other environments that were beneficial included many public libraries and coffeeshops. Throughout this thesis, the author learned that incorporating varied work environments is beneficial to increasing productivity.

Appendix C

Software Versions, Workflow, Scripts and Writing Recommendations

This appendix includes details on the software versions used, workflow including a repository of scripts, and writing recommendations. The purpose of this appendix is to aid the reader in understanding the processes that were implemented throughout this thesis, and to give a starting point for the reader to implement similar or the same processes.

C.1 Software Versions

This section lists out the various different types of software that were used throughout the course of this thesis. The purpose of this section is that it was found that software versions are important to maintain compatibility with various workflows since various workflows were developed by the open-source community. As a result, some workflows have not been maintained as the software is upgraded. It is the users responsibility to determine if it is realistic to use the older software version, or to upgrade the workflow to be compatible the most recent software version.

The following list of software and the respective version is provided in no particular order. The purpose of the software may be optionally provided:

1. Linux system built using Mint, release 19.2 "Tina" (which runs on Ubuntu version 18.04 which lost Long-Term Support (LTS) on May 31, 2023).
   https://linuxmint.com/edition.php?id=267

2. OpenFOAM (.org), version 9. For running all simulations. https://openfoam.org/version/9/

3. OpenFOAM (.com), version 2112. For using the integrated cfMesh software (version 1.1).
   https://www.openfoam.com/news/main-news/openfoam-v2112

4. cfMesh, version 1.1. For generating 2D meshes and preliminary 3D meshes. Newer versions of cfMesh may not be free. https://cfmesh.com/

5. Salome, version 9.3.0. For developing "fms" geometry files for use with cfMesh.
   https://files.salome-platform.org/Salome/Salome9.3.0/SALOME_9_3_0_Release_Notes.pdf
   It appears that as of 8/25/2023, that the Salome download function has moved behind a request form and older verions of Salome are now longer publically available:
   https://www.salome-platform.org/?page_id=2430

6. Matlab, version 2022b. For post processing OpenFOAM data.
   https://www.mathworks.com/products/new_products/r2022b-transition.html

7. Paraview, built in with OpenFOAM 9 and stand alone version 5.11.1. For visualizing OpenFOAM data.
   https://discourse.paraview.org/t/paraview-5-11-1-is-available-for-download/11805.
   Installation issues: https://discourse.paraview.org/t/error-when-building-paraview-5-9-1-with-python-enabled/12589/4

8. Remote Access Software:

   (a) NoMachine. For remote visual access into system. This software is capable of remotely restarting the computer, and allowing login after system restart.
   https://www.nomachine.com/

   (b) KiTTY. For remote SSH access into systems. https://www.fosshub.com/KiTTY.html This software was preferred over PuTTY because of additional features and a bug associated with PuTTY where the CTRL key does not work (https://superuser.com/a/1533323).

   (c) Xming. For allowing copying of text from one KiTTY window into another, and for visualizing graphics generated via KiTTY/SSH session.

C.2 Commonly Used Commands

There are many commands and programs that can be used in the Linux system. In no particular order, the following includes various commands and programs that were helpful throughout the course of this thesis. This section is laid out with the following format: "program; relevant command(s). Description. url, if applicable"

1. `chown`; `sudo chown user:user folder/file-of-interest`. To change the owner settings of a file/folder. https://linuxize.com/post/linux-chown-command/

2. `cinnamon-session-quit`. To logout via terminal. Beneficial if GUI is malfunctioning. https://technowikis.com/3446/how-to-log-out-of-linux-from-terminal

3. `crontab -e`. To automatically execute commands or scripts at regular intervals or specified times. https://www.geeksforgeeks.org/crontab-in-linux-with-examples/

4. Custom `PS1`; `PS1="\[\033[s\033[0;70H\033[0;41m\033[K\033[1;33m\t \033[0m\033[u\]\[\033[1;33m\]$USER-$HOSTNAME-\$OFVERSION \w \$\[\033[0m\] "` Output = USER-HOSTNAME-OpenFOAMversion; with the current time at the top of the terminal. This custom PS1 is beneficial to explicitly state which computer is being used, and if a specific OpenFOAM version is currently activated. See .bashrc file for definition of OFVERSION variable.

5. `df`; `df -h`. To get an overview of the total storage capacity.

6. `diff`, or `colordiff`. To compare 2 different files or directories and evaluate differences.

7. `ffmpeg`; `ffmpeg -framerate $1 -i $2%04d.png -c:v libx264 -strict -2 -preset slow -pix_fmt yuv420p -vf "scale=trunc(iw/2)*2:trunc(ih/2)*2" -f mp4 $(basename $(dirname $(pwd)))_$3$1.mp4;`. Create a mp4 video from a series of PNGs. https://stackoverflow.com/a/44033863

(a) `ls | cat -n | while read n f; do mv "$f" 'printf "a.%04d.png" $n'; done`. Rename images sequentially because ffmpeg does not like skipped numbers. See Seweryn Niemiec comment for adding leading zeros. https://stackoverflow.com/a/34153342

8. `find`; `find -type f -name "regex-file-name-of-interest" -exec grep 'phrase-to-search-for-within-that-file' {} +`. For finding a phrase within a certain file type. https://stackoverflow.com/a/30801017

9. `gnuplot`. To visualize data. https://stackoverflow.com/a/45803145 https://gnuplot.sourceforge.net/docs_4.2/node294.html#set_xrange

10. `grep`. To search for a given parameter in a file or folder.

11. `htop`. To get a dynamic overview of the system

12. `inxi -a -p`. To show the raw size of a drive

13. `inxi -Fxz`. To get a static overview of the system

14. `lscpu`, or `numactl --harware`. To check the construction of the CPUs, to setup numa partitioning.

15. `mate-keybinding-properties`. For changing keyboard shortcuts, such as forcing a window into 14 of the viewing screen.

16. `ncdu`. To see the organization and use of data within a directory. https://unix.stackexchange.com/a/342796

17. `nullmailer`. To setup scripts notification.sh and status.sh. https://wiki.debian.org/nullmailer

18. `pkill`; `pkill -9 -t pts/1`. To kill a rouge SSH session (probably initiated from PuTTY, but suspended because connection dropped). https://fedingo.com/how-to-kill-user-session-in-linux/

19. `rclone`. To setup a mount with OneDrive. https://rclone.org/. If searching the internet for how to use rclone, do NOT use the ITSFOSS article per the following information: https://forum.rclone.org/t/rclone-onedrive-enter-the-response-uri/19374/13

20. `rsync`. Preferred over `cp` for moving large directories. It allows for maintaining file ownership which aids with transferring data in a backup form (to make the destination be like the source) https://serverfault.com/questions/141773/what-is-archive-mode-in-rsync.

21. `sudo sh -c 'echo 1 > /proc/sys/vm/drop_caches'`; `sudo sh -c 'echo 2 > /proc/sys/vm/drop_caches'`. To clear RAM cache. https://www.geeksforgeeks.org/how-to-clear-ram-memory-cache-buffer-and-swap-space-on-linux/

22. `sudo shutdown -r now`. To restart the computer now. https://www.cs.stonybrook.edu/about-us/csintranet/faqs/Ubuntushutdown

23. `sudo shutdown -r +1`. To restart the computer in 1 minute.

24. `tail`; `tail ---disable-inotify -f -s 1 log.*`. To view the end of a file; to dynamically view the end of a log.* file.

25. `tar zxvf fileNameHere.tgz`. To unpack a .tgz file. https://www.cyberciti.biz/faq/unpack-tgz-linux-command-line/

26. `tree`. Software for visualizing folder and file structure.

27. `unzip file.zip` or `unzip file.zip -d destination_folder`. To unzip a .zip file. https://askubuntu.com/questions/86849/how-to-unzip-a-zip-file-from-the-terminal

28. `vim`. Preferred text editor that has many easy to use shortcuts.

    (a) Vim Cheat Sheet. https://vim.rtorr.com/

    (b) How to search in vim. https://linuxize.com/post/vim-search/

(c) Editing .vimrc to convert a tab to 4 spaces.

https://stackoverflow.com/questions/234564/

(d) Vim does not work as expected over SSH. Configure KiTTY/PuTTY to fix this.

https://vim.fandom.com/wiki/PuTTY_numeric_keypad_mappings

(e) Changing color schemes for better visualization, especially in OpenFOAM.

i. Configure SSH terminal to use 256 colors, not 8:

https://stackoverflow.com/questions/12064047/

https://superuser.com/questions/335471/

ii. Choose a vim colorsheme for .vimrc:

https://stackoverflow.com/questions/2975994/,

https://www.dunebook.com/best-vim-themes/

iii. Vundule may need to be setup: https://github.com/VundleVim/Vundle.vim

iv. For specific OpenFOAM visualization, setup Tobias Holzmann extensions:

https://bitbucket.org/shor-ty/vimextensionopenfoam/src/master/ which requires the use of pathogen, https://github.com/tpope/vim-pathogen

(f) Set vim to accept more lines into the memory.

https://stackoverflow.com/questions/3676855/vim-limited-line-memory

(g) Disable automatically creating *~ backup files. https://unix.stackexchange.com/a/3255

(h) Install vim-gnome to allow for xterm_clipboard to be installed to allow easier data copying between terminals.

https://unix.stackexchange.com/a/12565

https://superuser.com/a/805244

(i) Paste from vim if buffer was overwritten. https://stackoverflow.com/a/3651797

C.3 Workflow

In order to conduct simulations, a specific workflow was carried out to facilitate the geometry development, simulation setup & meshing, running a simulation, and post-processing the results. This section is included to detail the workflow that was used in this thesis.

C.3.1 Geometry Development

First, the geometry was developed using Salome, with cfmesh scripts used to generate a .fms file for input into cfmesh. The following details the steps required to carry out this work, sourced from this link:
https://curiosityfluids.com/2019/02/19/a-cfmesh-workflow-to-speed-up-and-improve-your-meshing/

1. Import the geometry into Salome. A .step file is preferred over a .stl file because the .stl file carries innate triangulation which propagates into the meshing step. The .step file does not have triangulation, and the meshing step is completed using pure geometry. A .step file can be saved from SolidWorks.

2. Open Salome 9.3. Download the relevant Python 3 scripts. The scripts available in OpenFOAM v2112 were developed for Salome 8.5.0. Use these updated scripts to run in Salome 9.3.0.

    - Updated **extractFeatureEdges** script (comment #25): https://www.cfd-online.com/Forums/openfoam-community-contributions/198872-general-workflow-create-flawless-mesh-cfmesh-2.html#post741088
    - Updated **salomeTriSurf** script (comment #19) https://www.cfd-online.com/Forums/openfoam-community-contributions/198872-general-workflow-create-flawless-mesh-cfmesh.html#post728586

The following workflow varies depending on if the geometry of interest is 3D or 2D:

3. 3D Workflow:

   (a) Create a geometry of the fluid domain (use boolean subtraction to remove solids).

4. 2D Workflow. 2D meshes require a RIBBION geometry. In Salome, this translates to a shell. For simple geometries:

   (a) Create a 3D representation of the geometry. Draw the geometry onto the XY plane. The Z-axis will have elongated cells because of the 2D geometry.

   (b) Build all faces of the geometry - neglect faces in the z direction.

   (c) Combine these faces into a Shell.

   The remaining workflow is the same for 3D and 2D geometries.

5. Create patches: **New Entity -> Group -> Create Group**

6. Extract Edges: Select **geometry in the tree**, hit **Ctrl+t**, then select **ExtractFeatureEdges3.py** script.

7. Switch to Meshing Window.

8. Generate Surface Mesh. Check that the mesh has faces and is not only a mesh of nodes. Mesh should show all components with adequate resolution, but does not need to be refined as the meshing software will implement the meshing resolution. Meshing recommendations:

   - Preference: **Netgen 1D-2D w/default settings**
   - Alternative: 2D settings with Triangle: Mefisto algorithm (provides an uniform triangle surface compared to NETGEN) and default hypothesis; 1D settings with Wire Discretization and adaptive hypothesis (play with the deflection value [not too small at first otherwise the display will freeze]. 0.001 seems like a good value for higher resolution on curves.)

9. Export geometry mesh to fms. With **Mesh1** Selected:

   (a) In mesh, go to **Mesh-> Create Groups from Geometry** and select **patches** and **featureEdges**.

   (b) **Ctrl+t** and load the **salomeTriSurf3.py** script.

   (c) Execute `triSurf(allEdges=True).writeFms('FileName.fms')`

Note: When manipulating the geometry, keep in mind that 1 mm in Salome becomes 1000 mm (1 m) in OpenFOAM. Manipulate and design the geometry accordingly. https://www.cfd-online.com/Forums/openfoam-meshing/63436-dimensions-salome.html#post212581

C.3.2 Simulation Setup & Meshing

In order to execute a simulation, a simulation folder must first be setup and initialized.

First execute:

```
cd ~/OpenFOAM/myWork
```

Then create the working directory for the simulation. For the purpose of this example, a generic folder will be created using the typical naming format; this naming format is important because different inputs in various scripts are based on the folder name. The folder naming format is as follows, where (parenthesis indicate an individual input) and [square brackets indicate a specific set of allowed inputs for the parenthetical set, with the optional values after the dash "-" as a more descriptive version of the shorthand allowed input. For example, "A - air"]:

1. YYYY-MMDD(alpha1)(alpha2)_

2. (geometryType[case2 - semi confined rectangular cylinder geometry, unconfined - rectangular cylinder in free flow, flowDev - flow development geometry, 53gcm - 3D high sided vehicle geometry])_

3. (fluid type[A - air, W - water])(arbitrary velocity number based on naming convention [0_001-0_100, and 0-8])_

4. (Flow regime[DevelopedFlow,UniformFlow])_

5. (Geometry Dimensions[XXm-height_XXm-upstream_XXm-downstream])

6. (Other miscellaenous attributes)

Using this naming convention, the following folder name has been selected:

`2023-0930aa_case2_W1_UniformFlow_12m-height_10m-upstream_7m-downstream_DEMO-PREP`

`cd 2023-0930aa_case2_W1_UniformFlow_12m-height_10m-upstream_7m-downstream_DEMO-PREP`

Inside of the simulation folder, execute:

`cpallNewCaseSetup`

This command refers to a quick reference command that is stored in the `.bashrc` file (Code Listing C.58). With this command executed, this imports the parent file tree from `~/OpenFOAM/myWork/global-base-case/` into the present folder. The resulting file structure is given in Appendix C.4.

Execute `ns` which also refers to a command that is stored in the `.bashrc` file (Code Listing C.58). Using vim, page through the three files that are preloaded, and edit as needed to make the system inputs. The three files are as follows:

1. `cfmesh/system/meshDict`.

   (a) Edit the value for the `surfaceFile "../...fms";` row, to direct the meshing to the appropriate `.fms` file.

   (b) Edit other values as necessary, including the `maxCellSize`, `patchBoundaryLayers`, and other meshing settings.

   (c) To write the data to the file, execute `:w`

   (d) To proceed to the 2nd file, execute `:next`

2. `analysis/globalMaster`

(a) Edit the values as necessary for this simulation. Typically, once a set of simulations has been defined, the processor definition is kept the same, and as a result, only the `numaNode` value is changed with each successive simulation.

(b) `numaNode` isolates the simulation into one of two numa's on Feynman's computer structure.

(c) `mpiProcessors(XYZ)` subdivides the simple decomposition method in a systematic manner.

(d) `mpiSleep` edits the amount of time between successive executions of the `helperRunningMPI.sh` script.

(e) To write the data to the file, execute `:w`

(f) To proceed to the 3rd file, execute `:next`

3. `analysis/system/controlDict`

(a) Edit the values as necessary for this simulation.

(b) `endTime` is changed depending on the simulation type and flow velocity.

(c) `writeInterval` and `purgeWrite` is changed depending on the visualization requirements for animations.

(d) To write the data to the file and exit the vim editor, execute `:wq`

Now that the preliminary data has been inputted and setup, a series of scripts must be executed to implement the final simulation setup. Execute `nsbashSetup`, which calls a script in `.bashrc` (Code Listing C.58). Depending on the configuration of the simulation, one of the following results will happen:

- Fluid Type. Fluid properties changed depending on if air or water is selected.
- Geometry Type. Depending on the selected geometry type, different values will be edited in `analysis/system/controlDict`, and the boundary conditions will be defined in the `0* folders`.

- If Uniform Flow is selected:
  - If the correct mesh exists, copy from mesh database in `~/OpenFOAM/myWork/global/geometry/` and remove `developMap` folder
  - If the correct mesh does not exist, mesh the geometry.
- If Developed Flow is selected:
  - If the correct mesh AND an associated developed flow profile exist in the `~/OpenFOAM/myWork/global/geometry/` database, copy to local folder.
  - If the correct mesh exists in the database, but the associated developed flow profile does not exist, copy the existing mesh to the local folder, and prepare the `developMap` folder for finalizing the developed flow profile for this specific geometry.
  - If neither the mesh or the associated developed flow profile exist, mesh the geometry, and prepare the `developMap` folder for finalizing the developed flow profile for this specific geometry.

In the case of this example simulation, the required mesh already exists, and was simply copied into the local folder.

NOTE: This workflow section did not significantly explain the mechanics of the developed flow profile. In essence, the output of a flow development simulation requires remapping to the input of any other simulation mesh. This remapping is achieved through a mix of Linux scripting and Matlab coding. This code is laid out in the script repository in Appendix C.5. There is a significant amount of comments throughout the code as it is executed. This should be helpful in determining how this code works.

C.3.3 Running a Simulation

Now that the simulation has been setup, it is run by `cd analysis` and executing `nbashallMaster` which calls a script in `.bashrc` (Code Listing C.58). Following this, the simulation can be textually checked by executing `tailallRun` or `tailT` or `taillog`. Each of these codes provide a different type of feedback from the various log files. Additionally, the convergence of the simulation can be evaluated by executing `gg`.

`bashstatus` can be executed by the user to calculate the expected completion of the simulation.

Once the simulation has completed, backup the data to the internal storage harddrive and the file server by `cd ../` and executing `bash allBackupStorage.sh`

C.3.4 Post-Processing Results

Once the simulation has completed, the results can be analyzed in Paraview or Matlab.

To analyze the results in Paraview, `cd analysis` and execute `parafoamfoam` which calls a script in `.bashrc` (Code Listing C.58). Various state files are saved in `~/OpenFOAM/myWork/global/paraview` to expedite the analysis of different types of data For a new analysis, use the `Save State` feature to retain this information for future use.

To analyze the results in Matlab, execute `bash allMatlabSetup.sh` to copy the most recent Matlab datafiles and start Matlab.

C.4 Folder Structure

C.4.1 Parent Simulation Folder

```
allBackupStorage.sh
allCellSetup.sh
allMatlabSetup.sh
allNewCaseSetup.sh
allSetupSettings.sh
analysis
cfmesh
developMap
global_values.m
```

analysis

```
0-developedFlow
   include
       atmospherePatch
       frontBackPatches
       initialConditions
   k
   nut
   omega
   p
   U
0-flowuniform
   include
       atmospherePatch
       fixedInlet
       frontBackPatches
       initialConditions
   k
   nut
   omega
   p
   U
0-unconfined
   include
       atmospherePatch
       fixedInlet
       frontBackPatches
       initialConditions
   k
   nut
   omega
   p
   U
allClean.sh
allFilesClear-Processor_ddt-n-_0.sh
allItemsToRevisit
allLatestMasterRunNotifyClean.sh
allMasterCleanRunNotifyClean.sh
allProcessorsClean.sh
allRun.sh
allUpdateScriptFiles.sh
constant
   momentumTransport
   transportProperties
foam.foam
globalMaster
gnu-courant
gnu-deltaT
gnu-forceCoeffs
```

```
gnu-residuals
helperMaster.sh
helperRunningMPI.sh
helperTouchLogs.sh
system
   controlDict
   decomposeParDict
   fvSchemes
   fvSolution
z_global-version_case2-2D-analysis_2023-0606
zItemsToRevisit
```

cfmesh

```
all2D-Mesh.sh
allCopy.sh
allMesh.sh
constant
foam.foam
system
   controlDict
   fvSchemes
   fvSolution
   meshDict
   meshDict_gcm
z_global-version_2023-0510
```

developMap

```
allClean.sh
allFinalInletData.sh
allMatlabSetup.sh
allProcessC.sh
allRun.sh
allTestMatlabIndexing.sh
startup.m
z_global-version_2023-0622
```

matlab

```
ALL_clear.m
ALL_import.m
fft_analysis.m
import_n_functions
   clearImportedVars.m
   clearUnusedVars.m
   function_uPlus_yPlus_calc.m
   importData_forceCoefficients.m
   importData_forceIncompressible_GCM.m
   importData_forceIncompressible.m
   importData_profileYaxis.m
   importData_yPlus.m
   transientDataCrop_forceCoefficients.m
   transientDataCrop_yPlus.m
NMF_Figure07_DevelopedFlowProfile.m
plot_forceCoeff.m
plot_forceGCM.m
plot_uPlus_yPlus.m
plot_yPlus.m
potentially_for_future
   ALL_transientDataCrop.m
   importData_probe.m
   importData_wallShearStress.m
z_global-version_2023-0803
```

C.4.2 /home/usr/OpenFOAM/myWork/global Folder

Output of `tree -L 2`:

```
archive
   080ms_Ti_5percent
   developed
   fluidProperties_incorrect-pre-2023_0228
   fvSchemes
   fvSchemes-kOmega-globalSettings
   fvSchemes-omega
   kOmegaSST-ICBC
   momentumTransport-kEpsilon
   mpi_numa
developedFlow
   100m_results
   200m_results
fluidProperties
functions
   archive
   case2
```

```
   flowDevelopment
   func_fieldAverage
   func_wallShearStress
   func_yPlus
   gcm
   global
   validation
fvSchemes-kOmegaSST
fvSchemes-kOmegaSST-53GCM
fvSolution
fvSolution-53GCM
fvSolution-SIMPLE
geometry
   53-GCM_12m-height_10m-upstream_7m-downstream_12m-
      boundingTruck_C-05_Layers_GCM-0_g-0
   53-GCM_12m-height_10m-upstream_7m-downstream_12m-
      boundingTruck_C-05_Layers_GCM-6_g-10
   53-GCM_12m-height_10m-upstream_7m-downstream_12m-
      boundingTruck_C-05_Layers_GCM-6_g-10
      _refinementBelowBottomOfTrailer_FrontToBack
   case2-semiConfined_12m-height_10m-upstream_7m-downstream_C
      -010122_Layers_c2-0_g-0
   case2-semiConfined_12m-height_10m-upstream_7m-downstream_C
      -010122_Layers_c2-20_g-24
   case2-semiConfined_12m-height_10m-upstream_7m-downstream_C
      -010122_Layers_c2-6_g-10
   case2-semiConfined_2m-height_5m-upstream_5m-downstream_C-010122
      _Layers_c2-6_g-10
   flowDevelopment_100m-length_12m-height_C-010122_Layers_c2-6_g-0
   flowDevelopment_100m-length_12m-height_C-010122_Layers_c2-6_g
      -10
   flowDevelopment_100m-length_12m-height_C-010122_Layers_c2-6_g
      -24
   flowDevelopment_200m-length_12m-height_C-010122_Layers_c2-6_g
      -24
   unconfined-case2_12m-height_10m-upstream_7m-downstream_C-010122
      _Layers_c2-0_g-10
   unconfined-case2_12m-height_10m-upstream_7m-downstream_C-010122
      _Layers_c2-20_g-10
   unconfined-case2_12m-height_10m-upstream_7m-downstream_C-010122
      _Layers_c2-6_g-10
momentumTransport-kOmegaSST
paraview
   case2-2D
   flowDevelopment
   GCM
velocity
   A0
   A1
   A7
```

```
archive
W0_001
W0_002
W0_005
W0_007
W0_010
W0_025
W0_050
W0_075
W0_100
W1
W2
W3
W4
W5
W6
W7
W8
W_base
```

/home/usr/OpenFOAM/myWork/global/developedFlow Folder

```
100m_results
  allImport100mResults.sh
  allImport100mResults.sh_200m_W8
  allTrim.sh
  allTrim.sh_200m_W8
  W0_001_1550-00sec
      original
        C
        k
        nut
        omega
        U
      trimmed
       trimmed_dF_W0_001_C
       trimmed_dF_W0_001_k
       trimmed_dF_W0_001_nut
       trimmed_dF_W0_001_omega
       trimmed_dF_W0_001_U
       wHeader_C
       wHeader_k
       wHeader_nut
       wHeader_omega
       wHeader_U
  W0_002_725-00sec
      original
        C
```

```
        k
        nut
        omega
        U
    trimmed
      trimmed_dF_W0_002_C
      trimmed_dF_W0_002_k
      trimmed_dF_W0_002_nut
      trimmed_dF_W0_002_omega
      trimmed_dF_W0_002_U
      wHeader_C
      wHeader_k
      wHeader_nut
      wHeader_omega
      wHeader_U
W0_005_387-50sec
    original
      C
      k
      nut
      omega
      U
    trimmed
      trimmed_dF_W0_005_C
      trimmed_dF_W0_005_k
      trimmed_dF_W0_005_nut
      trimmed_dF_W0_005_omega
      trimmed_dF_W0_005_U
      wHeader_C
      wHeader_k
      wHeader_nut
      wHeader_omega
      wHeader_U
W0_007_266-50sec
    original
      C
      k
      nut
      omega
      U
    trimmed
      trimmed_dF_W0_007_C
      trimmed_dF_W0_007_k
      trimmed_dF_W0_007_nut
      trimmed_dF_W0_007_omega
      trimmed_dF_W0_007_U
      wHeader_C
      wHeader_k
      wHeader_nut
      wHeader_omega
```

```
      wHeader_U
W0_010_195-00sec
    trimmed
      trimmed_dF_W0_C
      trimmed_dF_W0_k
      trimmed_dF_W0_nut
      trimmed_dF_W0_omega
      trimmed_dF_W0_U
      wHeader_C
      wHeader_k
      wHeader_nut
      wHeader_omega
      wHeader_U
W0_025_82-50sec
    trimmed
      trimmed_dF_W0-025_C
      trimmed_dF_W0-025_k
      trimmed_dF_W0-025_nut
      trimmed_dF_W0-025_omega
      trimmed_dF_W0-025_U
      wHeader_C
      wHeader_k
      wHeader_nut
      wHeader_omega
      wHeader_U
W0_050_43-00sec
    trimmed
      trimmed_dF_W0-050_C
      trimmed_dF_W0-050_k
      trimmed_dF_W0-050_nut
      trimmed_dF_W0-050_omega
      trimmed_dF_W0-050_U
      wHeader_C
      wHeader_k
      wHeader_nut
      wHeader_omega
      wHeader_U
W0_075_30-00sec
    trimmed
      trimmed_dF_W0-075_C
      trimmed_dF_W0-075_k
      trimmed_dF_W0-075_nut
      trimmed_dF_W0-075_omega
      trimmed_dF_W0-075_U
      wHeader_C
      wHeader_k
      wHeader_nut
      wHeader_omega
      wHeader_U
W0_100_23-50sec
```

```
    trimmed
     trimmed_dF_W0-100_C
     trimmed_dF_W0-100_k
     trimmed_dF_W0-100_nut
     trimmed_dF_W0-100_omega
     trimmed_dF_W0-100_U
     wHeader_C
     wHeader_k
     wHeader_nut
     wHeader_omega
     wHeader_U
W1_12-40sec
    trimmed
     trimmed_dF_W1_C
     trimmed_dF_W1_k
     trimmed_dF_W1_nut
     trimmed_dF_W1_omega
     trimmed_dF_W1_U
     wHeader_C
     wHeader_k
     wHeader_nut
     wHeader_omega
     wHeader_U
W2_07-20sec
    trimmed
     trimmed_dF_W2_C
     trimmed_dF_W2_k
     trimmed_dF_W2_nut
     trimmed_dF_W2_omega
     trimmed_dF_W2_U
     wHeader_C
     wHeader_k
     wHeader_nut
     wHeader_omega
     wHeader_U
W3_05-10sec
    trimmed
     trimmed_dF_W3_C
     trimmed_dF_W3_k
     trimmed_dF_W3_nut
     trimmed_dF_W3_omega
     trimmed_dF_W3_U
     wHeader_C
     wHeader_k
     wHeader_nut
     wHeader_omega
     wHeader_U
W4_04-00sec
    trimmed
     trimmed_dF_W4_C
```

```
      trimmed_dF_W4_k
      trimmed_dF_W4_nut
      trimmed_dF_W4_omega
      trimmed_dF_W4_U
      wHeader_C
      wHeader_k
      wHeader_nut
      wHeader_omega
      wHeader_U
W5_03-25sec
    trimmed
      trimmed_dF_W5_C
      trimmed_dF_W5_k
      trimmed_dF_W5_nut
      trimmed_dF_W5_omega
      trimmed_dF_W5_U
      wHeader_C
      wHeader_k
      wHeader_nut
      wHeader_omega
      wHeader_U
W6_02-75sec
    trimmed
      trimmed_dF_W6_C
      trimmed_dF_W6_k
      trimmed_dF_W6_nut
      trimmed_dF_W6_omega
      trimmed_dF_W6_U
      wHeader_C
      wHeader_k
      wHeader_nut
      wHeader_omega
      wHeader_U
W7_02-40sec
    trimmed
      trimmed_dF_W7_C
      trimmed_dF_W7_k
      trimmed_dF_W7_nut
      trimmed_dF_W7_omega
      trimmed_dF_W7_U
      wHeader_C
      wHeader_k
      wHeader_nut
      wHeader_omega
      wHeader_U
W8_00-54sec
   original
      C
      k
      nut
```

```
        omega
        U
    trimmed
        trimmed_dF_W8_C
        trimmed_dF_W8_k
        trimmed_dF_W8_nut
        trimmed_dF_W8_omega
        trimmed_dF_W8_U
        wHeader_C
        wHeader_k
        wHeader_nut
        wHeader_omega
        wHeader_U
200m_results
  080ms_0-84sec
    original
    trimmed
        trimmed_dF_080ms_C
        trimmed_dF_080ms_k
        trimmed_dF_080ms_nut
        trimmed_dF_080ms_omega
        trimmed_dF_080ms_U
        U_sorted
        wHeader_C
        wHeader_k
        wHeader_nut
        wHeader_omega
        wHeader_U
  144ms_0-48sec
    original
    trimmed
        trimmed_dF_144ms_C
        trimmed_dF_144ms_k
        trimmed_dF_144ms_nut
        trimmed_dF_144ms_omega
        trimmed_dF_144ms_U
        U_sorted
        wHeader_C
        wHeader_k
        wHeader_nut
        wHeader_omega
        wHeader_U
  208ms_0-34sec
    original
    trimmed
        trimmed_dF_208ms_C
        trimmed_dF_208ms_k
        trimmed_dF_208ms_nut
        trimmed_dF_208ms_omega
        trimmed_dF_208ms_U
```

```
        U_sorted
        wHeader_C
        wHeader_k
        wHeader_nut
        wHeader_omega
        wHeader_U
272ms_0-27sec
    17:56:44
    original
    trimmed
        trimmed_dF_272ms_C
        trimmed_dF_272ms_k
        trimmed_dF_272ms_nut
        trimmed_dF_272ms_omega
        trimmed_dF_272ms_U
        U_sorted
        wHeader_C
        wHeader_k
        wHeader_nut
        wHeader_omega
        wHeader_U
336ms_0-22sec
    original
    trimmed
        trimmed_dF_336ms_C
        trimmed_dF_336ms_k
        trimmed_dF_336ms_nut
        trimmed_dF_336ms_omega
        trimmed_dF_336ms_U
        U_sorted
        wHeader_C
        wHeader_k
        wHeader_nut
        wHeader_omega
        wHeader_U
400ms_0-19sec
    original
    trimmed
        trimmed_dF_400ms_C
        trimmed_dF_400ms_k
        trimmed_dF_400ms_nut
        trimmed_dF_400ms_omega
        trimmed_dF_400ms_U
        U_sorted
        wHeader_C
        wHeader_k
        wHeader_nut
        wHeader_omega
        wHeader_U
464ms_0-16sec
```

```
    original
    trimmed
        trimmed_dF_464ms_C
        trimmed_dF_464ms_k
        trimmed_dF_464ms_nut
        trimmed_dF_464ms_omega
        trimmed_dF_464ms_U
        U_sorted
        wHeader_C
        wHeader_k
        wHeader_nut
        wHeader_omega
        wHeader_U
  allImport200mResults.sh
  allTrim.sh
  archive
        allTrim_edited-2023_0211.sh
```

C.4.3 /home/usr/OpenFOAM/myWork/global/paraview Folder

```
case2-2D
  archive
      para_anim_U-streamlines-w-Cd-calc-vWORKING.pvsm
  cameraConfig_VnV_DomainHeightCompressed.pvcc
  para_anim_Q-Criterion_U-streamlines.pvsm
  para_anim_U-streamlines.pvsm
  para_anim_U-streamlines_unconfinedBody.pvsm
  para_anim_U-streamlines-w-Cd-dat-file.pvsm
  para_JFM_URANS_streamlines-w-Glyphs_qCriterion.pvsm
  para_JFM_URANS_streamlines-w-Glyphs_qCriterion_unconfinedBody.
      pvsm
  para_paper_Downstream-PressureVel-Plots.pvsm
  para_Thesis_U-vel-development.pvsm
  para_U-streamlines.pvsm
  para_U-vel-development.pvsm
  para_VnV_Downstream-AvgU-streamlines.pvsm
  para_VnV_HeightCompressed_U-streamlines_12mHeight.pvsm
  para_VnV_HeightCompressed_U-streamlines_12mHeight_wGlyphs.pvsm
  para_VnV_HeightCompressed_U-streamlines.pvsm
  para_VnV_Plot_AvgPressure-Upstream-of-Cylinder.pvsm
  para_vorticity-streamlines.pvsm
flowDevelopment
  para_anim_U-profile-at-X-equals-0_probePlot.pvsm
  para_anim_U-profile-at-X-equals-0.pvsm
  para_cellCenterSize.pvsm
  para_pre_inflationCheck.pvsm
  para_U-profile-at-X-equals-0-and-yPlus.pvsm
```

```
   para_U-profile-at-X-equals-0_DECOMPOSED.pvsm
   para_U-profile-at-X-equals-0.pvsm
GCM
   para_anim_U-streamlines_w3D-GCM-Body_n-GroundPlane_n-
       TrailerSlice.pvsm
   para_mesh-GCM-preview_slice-half-n-half.pvsm
   para_U-slices-decomposed.cgns
```

C.4.4 /home/usr/OpenFOAM/myWork/global/geometry Folder, select folders

```
case2-semiConfined_12m-height_10m-upstream_7m-downstream_C-010122
    _Layers_c2-6_g-10
   dateGeoCreated_2023-0406_0141PM
   dateLastCopied_2023-0930_0436PM
   log.cartesian2DMesh
   log.checkMesh
   log.renumberMesh
   meshDict
   polyMesh
       boundary
       cellZones
       faces
       faceZones
       neighbour
       owner
       points
       pointZones
   W0_010
       k.txt
       nut.txt
       omega.txt
       U.txt
   W0_025
       k.txt
       nut.txt
       omega.txt
       U.txt
   W0_050
       k.txt
       nut.txt
       omega.txt
       U.txt
   W0_075
       k.txt
       nut.txt
```

```
        omega.txt
        U.txt
    W0_100
        k.txt
        nut.txt
        omega.txt
        U.txt
    W1
        k.txt
        nut.txt
        omega.txt
        U.txt
    W2
        k.txt
        nut.txt
        omega.txt
        U.txt
    W3
        k.txt
        nut.txt
        omega.txt
        U.txt
    W4
        k.txt
        nut.txt
        omega.txt
        U.txt
    W5
        k.txt
        nut.txt
        omega.txt
        U.txt
    W6
        k.txt
        nut.txt
        omega.txt
        U.txt
    W7
        k.txt
        nut.txt
        omega.txt
        U.txt

unconfined-case2_12m-height_10m-upstream_7m-downstream_C-010122
    _Layers_c2-20_g-10
    dateGeoCreated_2023-0606_0247PM
    dateLastCopied_2023-0620_1030AM
    log.cartesian2DMesh
    log.checkMesh
    log.renumberMesh
```

```
   meshDict
   polyMesh
      boundary
      cellZones
      faces
      faceZones
      neighbour
      owner
      points
      pointZones
```

C.4.5 /home/usr/OpenFOAM/myWork/global-base-case Folder

Output of `tree -L 2`:

```
0-figures
allBackupStorage.sh
allCellSetup.sh
allCopyMatlabFiles.sh
allMatlabSetup.sh
allNewCaseSetup.sh
allRun_3DMesh-n-Run.sh
allRun_analysis-BackupStorage.sh
allSetupSettings.sh
allStatsStorage.sh
analysis
   archive
   case2-2D-analysis_2023-0606
   GCM*
   gcm-base-case_2023-0510
   kasun
archive
   allBackupStorageRsyncPartial.sh
   allBackupStorageRsync.sh
   allCellSetup_2023-0320.sh
   allCellSetup.sh_2023-0510
   allCellSetup.sh_2023-0620
   allCellSetup.sh_2023-0704
   allCellSetup.sh_2023-0705
   allCfmesh2D-developMap.sh
   allSetupMeshAndInletData.sh
   allSetupMeshAndInletData.sh_2023-0217
   allSetupSettings.sh_2023-0508
   allSetupSettings.sh_2023-0510
   allSetupSettings.sh_2023-0606
   allSetupSettings.sh_2023-0608
   allSetupSettings.sh_2023-0705
   allStats.sh
```

```
    allStatsStorage.sh_2023-0301
cfmesh
    2023-0510
    archive
    geometry
developMap
    2023-0622
    archive
    matlab
global_values.m
matlab
    2023-0807
    archive
startup.m
```

C.5 Script Repository

Many custom scripts were created throughout the course of this thesis. This section is included to briefly explain the purpose of these scripts and provide the scripts in their entirety.

Each simulation was setup inside a single parent folder, typically titled in the format of "YYYY-MMDDab_ simulationType_ FreeFlowVelocity_ UniformOrDevelopedFlow_ DetailsOnDomainDimensionsAndInflationLayers". Within this folder, 4 subfolders were included: 1) analysis, which contains the traditional OpenFOAM simulation; 2) cfmesh, which contained the meshing of the geometry; 3) developMap, which was an optional folder if developed flow was to be used; and 4) matlab, which is the folder containing all files necessary to evaluate the OpenFOAM data using Matlab. A final folder that aids in the developed flow profile will also be included. This section will contain the scripts from 6 locations, the parent folder, the 4 simulation sub-folders and the folder supporting the developed flow profile.

These scripts were created to complete the respective task, but are likely not the most efficient. It is likely that some of these scripts could be improved for the sake of efficiency and coding clarity.

C.5.1 Parent Simulation Folder

As previously described, each simulation was contained within a "Parent" folder. This allowed the model setup, analysis and local post processing to be contained within a single folder.

Scripts to Setup and Initialize New Simulations

Code Listing C.1: allNewCaseSetup.sh, to setup a new simulation and initialize all the folders.

```
#!/bin/bash

# First make the parent directory for this simulation PRIOR to
   running this script
echo "--- RUNNING allNewCaseSetup.sh ---"
mkdir cfmesh
mkdir analysis
mkdir developMap
cp -r /home/dksan/OpenFOAM/myWork/global-base-case/cfmesh/202*/*
      cfmesh/
cp -r /home/dksan/OpenFOAM/myWork/global-base-case/analysis/case2
   -2D*/* analysis/
cp -r /home/dksan/OpenFOAM/myWork/global-base-case/developMap
   /202*/*     developMap/
cp     /home/dksan/OpenFOAM/myWork/global-base-case/
   allBackupStorage.sh   .
cp     /home/dksan/OpenFOAM/myWork/global-base-case/allMatlabSetup
   .sh     .
cp     /home/dksan/OpenFOAM/myWork/global-base-case/global_values.
   m       .
cp     /home/dksan/OpenFOAM/myWork/global-base-case/
   allSetupSettings.sh   .
cp     /home/dksan/OpenFOAM/myWork/global-base-case/allCellSetup.
   sh      .

echo "--- allNewCaseSetup.sh FINISHED ---"
```

Parent Simulation Folder: Scripts to Setup and Initialize New Simulations

Code Listing C.2: allSetupSettings.sh, to automate most of the simulation settings based on the parameters given in the folder name.

```
#!/bin/bash
#PURPOSE: To setup most of all model settings to make automation
   easier

echo; basename $(pwd); echo

echo "--- RUNNING allSetupSettings.sh ---"

#Get variable names from existing file setup
export vel=$(       basename $(pwd) | sed 's/.*_W/W/' | sed 's/.*
   _A/A/' | sed 's/_UniformFlow.*//' | sed 's/_DevelopedFlow
   .*//')
export fluid=$(     echo $vel | cut -c1 )
export geo=$(       basename $(pwd) | sed -E -e 's/202.-.....(_|
   _...._)//' | sed 's/_.[[:digit:]].*_.*//')
export flowType=$(  basename $(pwd) | sed -E -e 's/.*(_..._|_..._
   ..._)//' | sed 's/Flow_.*/Flow/')
export upstream=$(  basename $(pwd) | sed 's/.*Flow_//g'     |
   sed 's/-upstream.*//g')
export downstream=$(basename $(pwd) | sed 's/.*upstream_//g' |
   sed 's/-downstream.*//g')
export inputVarName=$1

echo
echo "The following values have been obtained from the simulation
    name:"
echo " * Velocity: $vel  (A|W[0-7])"
echo " * Fluid: $fluid  (A|W)"
echo " * Geometry Type: $geo   (unconfined|flowDev|case2|53gcm)"
echo " * Flow Type: $flowType   (UniformFlow|DevelopedFlow)"
echo

# Update analysis/transportProperties with correct fluid type
if [[ "$fluid" == "W"  ]]; then
    #Do nothing. transportProperties is correct by default
    echo "> W: transportProperties kept as water_nu"
elif [[ "$fluid" == "A" ]]; then
    sed -i 's/water_nu/air_nu/' analysis/constant/
       transportProperties
    echo "> A: transportProperties updated to air_nu"
else
    echo "> W|A: ERROR while updating transportProperties"
fi
```

```
# Based on geometry type:
#   1) Change controlDict postProcessing functions
# Flow Development, comment out case2, comment in flowDevelopment
if [[ "$geo" == "flowDev" ]]; then
    echo "> flowDev: controlDict updated for flowDevelopment
       settings"
    echo "> flowDev: gnu-forceCoeffs deleted"
    sed -i '/\/\/.*flowDevelopment/s/^..//'     analysis/system/
       controlDict
    sed -i '/^#.*case2/s/^/\/\//'               analysis/system/
       controlDict
    rm analysis/gnu-forceCoeffs
else
    echo "> flowDev: controlDict NOT edited for flowDevelopment"
fi
# gcm, remove existing 0* folders and replace them with the gcm
   0* suite
if [[ "$geo" == "53gcm" ]]; then
    echo "> gcm: 0* folders updated for gcm settings"
    echo "> gcm: controlDict updated for gcm settings"
    echo "> gcm: allRun.sh updated for gcm settings"
    echo "> gcm: meshDict updated for gcm settings"
    rm -r analysis/0-*
    cp -r /home/dksan/OpenFOAM/myWork/global-base-case/analysis/
       gcm-base-case*/0-* analysis/
    sed -i '/\/\/.*gcm/s/^..//'   analysis/system/controlDict
    sed -i '/^#.*case2/s/^/\/\//'            analysis/system/
       controlDict
    sed -i 's/^#mpirun -np/mpirun -np/'         analysis/allRun.
       sh
    sed -i 's/^mpirun --bind/#mpirun --bind/'   analysis/allRun.
       sh
else
    echo "> gcm: 0* NOT deleted and replaced for gcm"
    echo "> gcm: controlDict NOT edited for gcm"
    echo "> gcm: allRun.sh NOT edited for gcm"
    echo "> gcm: meshDict NOT edited for gcm"
fi
# Unconfined, do nothing
# Case2 semi-confined, do nothing

#   2) Change 0/ boundary conditions to the approriate settings
if [[ "$geo" = "unconfined" ]]; then
    echo "> unconfined: Keeping 0-unconfined as 0/"
    mv analysis/0-unconfined analysis/0
    rm -r analysis/0-*
elif [[ "$flowType" == "UniformFlow" ]]; then
    echo "> UniformFlow: Keeping 0-flowuniform as 0/"
    mv analysis/0-flowuniform analysis/0
    rm -r analysis/0-*
```

```
elif [[ "$flowType" == "DevelopedFlow" ]]; then
    echo "> DevelopedFlow: Keeping 0-developedFlow as 0/"
    mv analysis/0-developedFlow analysis/0
    rm -r analysis/0-*
else
    echo "> Unconfined|UniformFlow|DevelopedFlow: ERROR while
       setting up 0/ folder"
fi

echo
bash allCellSetup.sh
echo

echo "--- allSetupSettings.sh FINISHED ---"
```

Parent Simulation Folder: Scripts to Setup and Initialize New Simulations

Code Listing C.3: allCellSetup.sh, called from allSetupSettings.sh - to copy over a previously generated mesh OR generate the mesh since it does not exist in the repository folder.

```
#!/bin/bash
#PURPOSE: To prepare the system mesh setup

echo "--- RUNNING allCellSetup.sh ---"

# Get information from pre-edited meshDict
fmsName=$(          cat cfmesh/system/meshDict | sed -n '/FMS/p'
   | sed 's/.*files_FMS-HDF\///' | sed 's/.fms";//')
fmsCellSize=$(      cat cfmesh/system/meshDict | sed -n '/
   maxCellSize/p' | cut -c15- | sed 's/.$//')
fmsLayersGround=$(  cat cfmesh/system/meshDict | sed -n '/
   ground_surface.*nLayers/p' | sed 's/.*nLayers //' | sed 's/;.*
   thickness.*//')
fmsC2BodyType=$(    cat cfmesh/system/meshDict | sed -n '/case2_
   .*nLayers/p' )

if [[ -z "$fmsC2BodyType"  ]]; then #Checks if the requested sed
   line from above has a value. If no value, then assumes geo
   type is GCM type
    echo "> Checking body type requested in meshDict: NOT C2 -->
       GCM"
    fmsLayersGCM=$(   cat cfmesh/system/meshDict | sed -n '/GCM_
       .*nLayers/p'   | sed 's/.*nLayers //' | sed 's/;.*
       thickness.*//')
    fmsNameComposite=$(echo ""$fmsName"_C-"$fmsCellSize"
       _Layers_GCM-"$fmsLayersGCM"_g-"$fmsLayersGround""
       $inputVarName"")
else
    echo "> Checking body type requested in meshDict: YES C2"
    fmsLayersCase2=$( cat cfmesh/system/meshDict | sed -n '/
       case2_.*nLayers/p' | sed 's/.*nLayers //' | sed 's/;.*
       thickness.*//')
    fmsNameComposite=$(echo ""$fmsName"_C-"$fmsCellSize"
       _Layers_c2-"$fmsLayersCase2"_g-"$fmsLayersGround"")

fi

echo "FMS composite name: $fmsNameComposite"
echo "geo from inside allCellSetup: $geo"
echo "InputVarName = $inputVarName"

# Function Definitions for code that repeats btwn Uniform &
   Developed Flow
copyMeshedGeo () {
```

```
    rsync -av  ~/OpenFOAM/myWork/global/geometry/
       $fmsNameComposite/polyMesh analysis/constant/
    rsync -a  ~/OpenFOAM/myWork/global/geometry/$fmsNameComposite
       /log.*     cfmesh/
    rm -f      ~/OpenFOAM/myWork/global/geometry/$fmsNameComposite
       /dateLastCopied_*
    touch      ~/OpenFOAM/myWork/global/geometry/$fmsNameComposite
       /dateLastCopied_$(date +%Y-%m%d_%I%M%p)
}
meshGeo () {
    # Check if the requested FMS file exists
    if [ -e /home/dksan/OpenFOAM/myWork/global-base-case/cfmesh/
       geometry/files_FMS-HDF/$fmsName.fms ]; then
        echo "> fms file exists in basecasecg. Proceeding with
           cfmesh-ing"
    else
        echo "> fms file does NOT exists in basecasecg"
        echo "> Exiting. Please generate the requested FMS file
           and then rerun this script"
        exit 1
    fi

    echo
    cd cfmesh
    if [[ "$geo" == "53gcm" ]]; then
        echo "> Mesh is 3D. Running allMesh.sh"; echo
        bash allMesh.sh
    else
        echo "> Mesh is 2D. Running all2D-Mesh.sh"; echo
        bash all2D-Mesh.sh
    fi
    echo
    echo "> cfmesh has been run."
    cd ../
    mkdir                              ~/OpenFOAM/myWork/global/
       geometry/$fmsNameComposite
    cp -r cfmesh/1/polyMesh            ~/OpenFOAM/myWork/global/
       geometry/$fmsNameComposite/
    cp -r cfmesh/log.*                 ~/OpenFOAM/myWork/global/
       geometry/$fmsNameComposite/
    cp -r cfmesh/system/meshDict       ~/OpenFOAM/myWork/global/
       geometry/$fmsNameComposite/
    touch      ~/OpenFOAM/myWork/global/geometry/$fmsNameComposite
       /dateGeoCreated_$(date +%Y-%m%d_%I%M%p)
    echo "> polyMesh has been backed up to global/geometry"
}

echo

# If analysis is with UniformFlow, then run the following
```

```
if [[ "$flowType" == "UniformFlow" ]]; then

    if   [ -d ~/OpenFOAM/myWork/global/geometry/$fmsNameComposite
        ]; then
        #Copy existing polyMesh files to constant/
        echo "> UniformFlow: FMS exists. Copying existing data"
        copyMeshedGeo
        rm -rf developMap/
        echo "> UniformFlow: fmsName/polyMesh has been copied to
           analysis/constant/"
    else
        if   [ -d /storage/dksan/zBackup/global-geometry-
           notUsedRecently/$fmsNameComposite ]; then
            echo "> UniformFlow: MeshedGeo EXISTS, but is in /
               storage/dksan/zBackup/global-geometry-
               notUsedRecently. UPDATE THIS SCRIPT OR mv DESIRED
               FOLDER TO global/geometry"
            echo "> UniformFlow: Exiting allCellSetup.sh"
        else
            echo "> UniformFlow: MeshedGeo does not exist.
               Meshing required."
            meshGeo
        fi
    fi

# If analysis is with DevelopedFlow, then run the following
elif [[ "$flowType" == "DevelopedFlow" ]]; then
    if   [ -d ~/OpenFOAM/myWork/global/geometry/$fmsNameComposite
        ]; then
    #Check if MeshedGeo exists
        #echo "MeshedGeo exists"

        if [ -d ~/OpenFOAM/myWork/global/geometry/
           $fmsNameComposite/$vel ]; then
        #If FMS exists, check if the mapped velocity field exists
            echo "> DevelopedFlow: MeshedGeo and VEL exist"
            case=1
        else
            echo "> DevelopedFlow: MeshedGeo exists, VEL does not
               "
            case=2
        fi

    else
        if   [ -d /storage/dksan/zBackup/global-geometry-
           notUsedRecently/$fmsNameComposite ]; then
            echo "> DevelopedFlow: MeshedGeo EXISTS, but is in /
               storage/dksan/zBackup/global-geometry-
               notUsedRecently. UPDATE THIS SCRIPT OR mv DESIRED
               FOLDER TO global/geometry"
```

```
        echo "> DevelopedFlow: Exiting allCellSetup.sh"
    else
        echo "> DevelopedFlow: Neither exists"
        case=3
    fi
fi

echo "> DevelopedFlow: Case value = $case. Running required
   meshing steps"; echo

case $case in
    1) # FMS and VEL exist
        #Copy existing polyMesh files to constant/
        copyMeshedGeo
        rm -rf developMap/
        echo "> fmsName/polyMesh has been copied to analysis/
           constant/"
        echo "> Mesh and inlet profile are ready."
        ;;

    2) # FMS exists. VEL does not
        #Copy existing polyMesh files to constant/
        copyMeshedGeo
        echo "> fmsName/polyMesh has been copied to analysis/
           constant/"

        #Run developMap
        cd developMap
        bash allRun.sh; echo
        echo "> developMap has been run."
        echo "> Mesh is ready. Run matlab and
           allFinalInletData.sh when ready"
        ;;

    3) # Neither exists
        echo "> MeshedGeo does not exist. Meshing required."
        meshGeo
        sleep 2

        #Run developMap
        cd developMap
        bash allRun.sh; echo
        echo "> developMap has been run."
        echo "> Mesh is ready. Run matlab and
           allFinalInletData.sh when ready"
        echo
        ;;
    *)
        echo "ERROR: No match in case structure"
        ;;
```

```
    esac
else
    echo "ERROR: Flow type not recognized. Mesh has NOT been
       setup."
fi

echo "--- allCellSetup.sh FINISHED ---"
```

Scripts for Completed Simulations

Code Listing C.4: allBackupStorage.sh, for backing up the final simulation results to the 1) internal hard-drive and 2) external NAS file server.

```
#!/bin/bash

# Run this script from the original working directory.
# This script automatically creates a new folder in /storage/
   dksan, backs it up to /nas, and deletes the original ru
echo "--- RUNNING allBackupStorage.sh ---"
echo "> File size of entire working directory is $(du -sh .)"
echo "> Current storage capacity:"
df -h | grep 'sda1\|md0\|Filesystem'

#Save current working directory
folderName=$(basename $(pwd))
sudo install -d -m 0755 -o dksan -g dksan /storage/dksan/
   $folderName
sudo install -d -m 0755 -o dksan -g dksan /storage/dksan/
   $folderName/cfmesh
sudo install -d -m 0755 -o dksan -g dksan /storage/dksan/
   $folderName/0-figures

#Backup to /storage
echo; echo "> Backing up $folderName to /storage"
sudo rsync -a --delete --info=progress2 analysis
                                        /storage/dksan/$folderName/
sudo rsync -a cfmesh/system/meshDict cfmesh/log*
                                        /storage/dksan/$folderName/
   cfmesh
sudo rsync -a 0-figures/
                                                              /
   storage/dksan/$folderName/0-figures/
sudo rsync -a global_values.m  log.*
                                                    /storage/dksan/
   $folderName/
sudo rsync -a /home/dksan/OpenFOAM/myWork/global-base-case/
   allMatlabSetup.sh        /storage/dksan/$folderName/
sudo rsync -a /home/dksan/OpenFOAM/myWork/global-base-case/
   allCopyMatlabFiles.sh    /storage/dksan/$folderName/
sudo rsync -a /home/dksan/OpenFOAM/myWork/global-base-case/
   allStatsStorage.sh       /storage/dksan/$folderName/
echo; echo ">  Backup of $folderName to /storage COMPLETE"

#Backup from /storage to /nas
echo;echo "> Backing up $folderName from /storage to /nas"
sudo rsync -a --delete --info=progress2 /storage/dksan/
   $folderName                          /nas/dksan/storage_bckp/
```

```
echo; echo "> Backup of $folderName from /storage to /nas
   COMPLETE"; echo

#Prompt to confirm deleting original data
echo "> ! Data has been backed up to /storage and /nas"
echo "> Current storage capacity:"
df -h | grep 'sda1\|md0\|Filesystem'; echo
read -p "> !! Are you sure you want to DELETE the original data
   in ~/OpenFOAM/myWork/?" -n 1 -r
echo;echo
if [[ $REPLY =~ ^[Yy]$ ]]
then
    #Delete original data
    echo ">   Deleting $folderName from ~/OpenFOAM/myWork"
    rm -r ~/OpenFOAM/myWork/$folderName
    echo ">   Deleting $folderName from ~/OpenFOAM/myWork
       COMPLETE"; echo
    echo "> Revised storage capacity:"
    df -h | grep 'sda1\|md0\|Filesystem'
else
    #Do NOT delete original data
    echo ">  Files are NOT being deleted out of ~/OpenFOAM/myWork
       "
fi

echo
echo "--- allBackupStorage.sh FINISHED ---"
```

Parent Simulation Folder: Scripts for Completed Simulations

Code Listing C.5: allMatlabSetup.sh, to update the local copy of matlab files and open Matlab.

```
#!/bin/bash

source ~/.bash_aliases
shopt -s expand_aliases

# Run this script from x11vnc, in simulation parent directory
# PURPOSE: Delete existing folder, then COPY matlab files to
   current directory and OPEN Matlab
echo; basename $(pwd); echo

echo "--- RUNNING allMatlabSetup.sh ---"
echo "> global_values.m"
cat global_values.m
echo "> Copying files from global-base-case to here"
sudo rm -rf matlab
sudo rm -f startup.m
sudo install -d -m 0755 -o dksan -g dksan matlab
sudo rsync -a    /home/dksan/OpenFOAM/myWork/global-base-case/
   matlab/202*/*       matlab/
sudo rsync -a    /home/dksan/OpenFOAM/myWork/global-base-case/
   startup.m           .

echo "> Opening matlab"
matlab
echo "--- allMatlabSetup.sh FINISHED ---"
```

Parent Simulation Folder: Scripts for Completed Simulations

Code Listing C.6: allStats.sh, to pull out key data from simulation for the purposes of checking key parameters, adding the data to the simulation spreadsheet and comparing the data with similar simulations.

```
#!/bin/bash
echo; basename $(pwd); echo

echo "--- RUNNING allStats.sh ---"
echo ---------------------- MESH stats -------------------------
cat cfmesh/meshDict      |   grep 'surfaceFile\|maxCellSize'; echo
cat cfmesh/meshDict      |   grep -B 4 'thicknessRatio\|
   additionalRefinement\|refinementThickness'; echo; echo
#cat cfmesh/log.cart*     |   grep 'ExecutionTime\|ClockTime'
cat cfmesh/log.cart*     |   grep 'ClockTime'
cat cfmesh/log.check*    |   grep 'cells:  '
#cat analysis/constant/polyMesh/neighbour     |   grep 'nCells:' |
    sed 's/.*nCells:/analysis polyMesh cell #: /' | sed 's/nFaces
   .*//'

echo ------------------------------------------------------------
echo

echo ------------------ ANALYSIS stats -------------------------
#cat analysis/0/include/initialConditions       | grep -B 2 '
   flowVelocity'
#cat analysis/allMasterCleanRunNotifyClean.sh   | grep 'deltaTsec
   ='
cat analysis/system/controlDict                | grep 'endTime '
echo "Data size: $(du -sh -- *                  | grep analysis |
   grep -v all | sed 's/analysis//' )"
#echo "Number of saved time series: $(find analysis/ -maxdepth 1
   -type d \( -name "[0-9]" -o -name "[0-9][0-9]" -o -name
   "[0-9][0-9][0-9]" -o -name "[0-9][0-9][0-9][0-9]" -o -name
   "*.*" \) | wc -l)"
echo "Number of saved time series: $(find analysis/ -maxdepth 1 -
   type d \( -name "[[:digit:]]*" \) | wc -l)"
cat analysis/log.allRun                        | grep 'Total
   Simulation'
cat analysis/log.allRun                        | grep 'Total
   mpirun RunTime'
cat analysis/globalMaster                      | grep '
   mpiProcessors='
echo ------------------------------------------------------------
echo "--- allStats.sh FINISHED ---"
```

Parent Simulation Folder: Scripts for Completed Simulations

Code Listing C.7: global_values.m, Global values that are saved in a text file for matlab so they are not disturbed by the allMatlabSetup.sh script.

```
xmin=2;
xmax=3;
```

Code Listing C.8: startup.m, Startup matlab script saved in a text file for matlab so they are not disturbed by the allMatlabSetup.sh script.

```
addpath(genpath('analysis/postProcessing/'))
addpath(genpath('matlab/'))
```

C.5.2 1) analysis Folder

The scripts contained in this section are used to run a simulation. Prior to using these scripts, work must be completed in the 2) cfmesh folder and 3) developMap folder.

Code Listing C.9: allMasterCleanRunNotifyClean.sh, Master script to run a new simulation.

```
#!/bin/bash
#PURPOSE: Run ALL typical run files in sequence starting from 0\
   time

source globalMaster
export mandantoryRest=$1

rm -f log.allRun
bash allClean.sh              >> log.allRun

bash helperMaster.sh
```

Code Listing C.10: allLatestMasterRunNotifyClean.sh, Master script to restart a simulation from a time saved in the backup folder

```
#!/bin/bash
#PURPOSE: Run ALL typical run files in sequence starting from
   latestTime

## BEFORE STARTING THE RESTARTED SIMULATION, CHECK:
# system/controlDict endTime is revised

source globalMaster
export mandantoryRest=$1

latest_time=$(ls | grep '[0-9]\+$' | sort -n | tail -n1)
rm -r           $latest_time
mkdir           backup/"$latest_time"_logs
mv log.*        backup/"$latest_time"_logs/
mv nohup.out    backup/"$latest_time"_logs/
cp -r           backup/$latest_time .

bash helperMaster.sh
```

analysis Folder

Code Listing C.11: allClean.sh, called by allMasterCleanRunNotifyClean.sh - To clean the analysis folder from all extraneous files - effectively restarting the simulation from the beginning.

```
#!/bin/bash
#PURPOSE: If a previous simulation has been run, this cleans out
   all unnecessary files to restart the simulation from 0\

echo "--- RUNNING allClean.sh ---"

find . -type d -name "pro*"                          -exec rm -rf {} +
find . -type d -name "post*"                         -exec rm -rf {} +
find . -type d -name "0.*"                           -not -path "*
   backup*"   -exec rm -rf {} +
find . -type d -name "[1-9].*"                       -not -path "*
   backup*"   -exec rm -rf {} +
find . -type d -name "[1-9]"                         -not -path "*
   backup*"   -exec rm -rf {} +
find . -type d -name "[1-9][0-9]"                    -not -path "*
   backup*"   -exec rm -rf {} +
find . -type d -name "[1-9][0-9][0-9]"               -not -path "*
   backup*"   -exec rm -rf {} +
find . -type d -name "[1-9][0-9][0-9][0-9]"          -not -path "*
   backup*"   -exec rm -rf {} +
find . -type d -name "[0-9]e-[0-9][0-9]"             -not -path "*
   backup*"   -exec rm -rf {} +
# in the FUTURE, try the following, especially if the lines
   immediately above need to be updated
# find -type d -name "[[:digit:]]*" -not -path "*backup*" -not -
   path "*0" -not -path "*postPro*"
find . -type f -name "log.*" -not -name "log.allRun" -exec rm -rf
    {} +
find ./0 -type f -name "C*"                                -exec rm -rf
    {} +
echo "--- allClean.sh FINISHED ---"
```

analysis Folder

Code Listing C.12: helperMaster.sh, Master script containing all subscripts that are applicable to both the allMasterCleanRunNotifyClean.sh and allLatestMasterRunNotifyClean.sh scripts.

```
#!/bin/bash
#PURPOSE: Runs Master sequence that is identical for:
#   allMasterCleanRunNotifyClean.sh
# allLatestMasterRunNotifyClean.sh

rm -f nohup.out
bash helperTouchLogs.sh
bash helperRunningMPI.sh     >> log.allRun
date                         >> log.allRun
echo "mandantory sleep for $mandantoryRest seconds" >> log.allRun
sleep $mandantoryRest
date                         >> log.allRun
bash allRun.sh               >> log.allRun
bash allProcessorsClean.sh   >> log.allRun
cat nohup.out                >> log.allRun
bash ~/notification.sh       >> log.allRun
bash ~/status.sh 1 sU4_simFinished > ~/log.status
```

Code Listing C.13: helperTouchLogs.sh, called by helperMaster.sh - to create all log files so that the tail command referenced in Section B.2 works flawlessly.

```
#!/bin/bash
#PURPOSE: touch log files to create them so that taillog
   automatically knows what files to pulse through

touch log.allRun
touch log.decomposePar
touch log.pimpleFoam
touch log.postProcess
touch log.reconstructPar
touch log.reconstructParLatestTime
touch log.warnings
```

analysis Folder

Code Listing C.14: helperRunningMPI.sh, called by helperMaster.sh - to place the simulation on hold until a previous MPI process on the specified numa completes.

```
#!/bin/bash
#PURPOSE: Every XX minutes, this while loop checks if the desired
    numa node is free
            # OR if the numa is being used by an existing mpi
               process.
            # Once the numa is free from an mpi process, the
               simulation is allowed to proceed

echo "--- RUNNING helperRunningMPI.sh ---"

rm -f log.helperRunningMPI
a=1 #Definition for a helper counter to track number of while
   loop checks

echo "Attempting run on Numa $(cat globalMaster  | grep 'numaNode
   =' | cut -c17)" >> log.helperRunningMPI
echo >> log.helperRunningMPI


while : #Infinite Loop runs until the desired numa value is NOT
   found in active mpi processes
do
    numaDesired=0 #Define value to compare numa existence with

    #Get information on Simulation Stats
    simPID=$(pgrep -u `whoami` mpirun)
    if [ -z "$simPID" ]
    then #No MPI simulation is running.
        : #numaDesired stays = 0 and the while loop will break
    else #MPI simulation is running
        simFolders=$(pwdx $simPID | sed 's/^[0-9]\+: //g') #Get
           working directory and remove PID values
        #Save base folder name to an array to be run through as a
            loop in the section below
        mapfile -t simNameArray < <( basename -a $(dirname
           $simFolders) )

        #Get numa value from running mpi processes
        for i in "${simNameArray[@]}"; do
            numaRunning=$(cat ~/OpenFOAM/myWork/$(echo "$i")/
               analysis/log.allRun                          | grep Numa
               | cut -c6)
```

```
            if [[ "$numaNode" == "$numaRunning" ]] #Check if
               desiredNuma is being used in existing mpi
               processes
            then #Desired numa is being used
                numaDesired=1
            fi
        done
    fi

    if [[ "$numaDesired" == 0 ]]
    then  #Desired numa is now FREE. break!
        echo "Attempt #$a at $(date): BREAKING OUT"  >> log.
           helperRunningMPI
        break
    else #Doing nothing because desiredNuma is being used
        echo "Attempt #$a at $(date): retrying soon" >> log.
           helperRunningMPI
        a=$(( $a+1 ))
    fi

    sleep $mpiSleep #Wait XX minutes before checking the mpi
       simulation conditions again

done

echo "--- helperRunningMPI.sh FINISHED ---"
```

analysis Folder

Code Listing C.15: allRun.sh, called by helperMaster.sh - to run the OpenFOAM simulation.

```
#!/bin/bash
#PURPOSE: To run a simulation using a standard run format

echo "--- RUNNING allRun.sh ---"
startTime=$(date +%H:%M:%S)
varStart=$(date +%s)
echo; echo "STARTED:"; date; echo

#Check OF Version and change to OF9 to run simulation properly
echo Current OFversion - $OFVERSION
if [[ "$OFVERSION" != "9" ]]
then
    source ~/OpenFOAM/OpenFOAM-9/etc/bashrc
    OFVERSION=9
fi

#RUN SIMULATION
echo Loaded OFversion for this run - $OFVERSION; echo
echo " --- Starting Simulation --- ";
echo "Numa $numaNode for $todaySimulationName"

#decomposePar
echo ">>> decomposePar Warnings <<<" >> log.warnings
echo "... decomposePar-ing"
decomposePar 2>&1                          | tee log.decomposePar
                | grep -B 1 'Warning' >> log.warnings
echo "> decomposePar complete"

#mpirun
echo ">>> mpirun Warnings <<<" >> log.warnings
echo "... mpirun-ing on numa $numaNode for simulation"
echo "$todaySimulationName"
mpivarStart=$(date +%s)
coproc coprocMPI0 { sleep 450;  bash ~/status.sh 1
    sU0_mpiStarted0 > ~/log.status; } #7.5 min update to send
    delayed status email after mpi starts
coproc coprocMPI1 { sleep 900;  bash ~/status.sh 1
    sU1_mpiStarted1 > ~/log.status; } #15  min update to send
    delayed status email after mpi starts
coproc coprocMPI2 { sleep 1800; bash ~/status.sh 1
    sU2_mpiStarted2 > ~/log.status; } #30  min update to send
    delayed status email after mpi starts
```

```
mpirun --bind-to none -np $mpiProcessors numactl --cpunodebind=
   $numaNode pimpleFoam -parallel 2>&1 | tee log.pimpleFoam |
   grep -B 1 'Warning' >> log.warnings
#mpirun -np $mpiProcessors pimpleFoam -parallel 2>&1 | tee log.
   pimpleFoam    | grep -B 1 'Warning' >> log.warnings
mpivarComp=$(date +%s)
mpirunTime=$( echo "$(( mpivarComp-mpivarStart )) / 3600" | bc -l
   )
printf '> Total mpirun RunTime = %.*f hrs\n' 9 $mpirunTime
echo "> mpirun complete ON $(date)"
echo "> controlDict/endTime is $(cat system/controlDict | grep '
   endTime ' | cut -c17- | sed 's/;.*//') sec"
bash ~/status.sh 1 sU3_mpiDONE > ~/log.status

#Save a backup copy of the most recent timestep in case the
   simulation needs to be later restarted
echo "... Backing up latestTime results"
mkdir -p backup
echo ">>> reconstructPar -latestTime Warnings <<<" >> log.
   warnings
reconstructPar -latestTime 2>&1          | tee log.
   reconstructParLatestTime  | grep -B 1 'Warning' >> log.
   warnings
backup_latest_time=$(ls processor0 | grep '[0-9]\+$' | sort -n |
   tail -n1)
echo "... latestTime is $backup_latest_time seconds"

#Check if latestTime folder <> 0
if [[ "$backup_latest_time" == "" ]] || [[ "$backup_latest_time"
   == 0 ]];then
    echo "> NOT backing up. 0/ folder is the latest time"; echo
else
    mv $backup_latest_time backup/
    echo "> latestTime result backup complete"; echo
fi

#Save HDD space before reconstructPar by deleting ddt* and *_0
   files from the processor folders
find ./processor* -type f -name "ddt*"  -exec rm -rf {} +
find ./processor* -type f -name "*_0"   -exec rm -rf {} +
echo "> deleting ddt* and *_0 files from processor folders
   complete"

#reconstructPar
echo ">>> reconstructPar -All Warnings <<<" >> log.warnings
echo "... reconstructPar-ing ..."
reconstructPar 2>&1                      | tee log.reconstructPar
            | grep 'Time = [0-9].[0-9][5]$\|Time = [0-9].[0-9]$
   ,
echo "> reconstructPar complete";
```

```
#postProcessing Functions that should occur ONLY at the
   latestTime
echo ">>> postProcess -All Warnings <<<" >> log.warnings
echo "... postProcess-ing ..."
postProcess -func writeCellCentres -latestTime 2>&1 | tee log.
   postProcess   | grep -B 1 'Warning' >> log.warnings
echo "> postProcess -latestTime complete";
echo " --- Simulation Complete --- "; echo

##Process time to run data
echo "COMPLETED:";date; echo
completedTime=$(date +%H:%M:%S)
varComp=$(date +%s)
runTime=$( echo "$(( varComp-varStart )) / 3600" | bc -l)
printf 'Total Simulation RunTime = %.*f hrs\n' 9 $runTime
echo "for $todaySimulationName"
echo


echo "--- allRun.sh FINISHED ---"
```

analysis Folder

Code Listing C.16: allProcessorsClean.sh, called by helperMaster.sh - to remove the processor folders.

```
#!/bin/bash
#PURPOSE: Simulation has finished, remove all unnecessary files

echo "--- RUNNING allProcessorClean.sh ---"
find . -type d -name "proc*" -exec rm -rf {} +
echo "--- allProcessorClean.sh FINISHED ---"
```

Code Listing C.17: allFilesClear-Processor_ddt-n-_0.sh, to be run during a simulation to clear the ddt* and *_0 files from the time folders.

```
#!/bin/bash
#PURPOSE: Run this script DURING a running simulation to delete
   extra 'ddt' and '_0' files that take up considerable space.
#Take care NOT to run this script when the simulation is in the
   final write time iteration b/c the data will be lost and the
   simulation will not be easily restarted from latestTime due to
    the data loss

echo "--- RUNNING allFilesClear-Processor_ddt-n-_0.sh ---"
echo
echo "> Storage CURRENT"
df -h | grep "/sda1\|/md0\|karan\|Filesystem"

#Delete ddt* and *_0 files from the processor folders
echo
echo "> deleting ddt* and *_0 files from processor folders"
find ./processor* -type f -name "ddt*"  -exec rm -rf {} +
find ./processor* -type f -name "*_0"   -exec rm -rf {} +
echo "> deleting ddt* and *_0 files from processor folders
   complete"
echo

echo "> Storage CLEANED UP"
df -h | grep "/sda1\|/md0\|karan\|Filesystem"

echo "--- allFilesClear-Processor_ddt-n-_0.sh FINISHED ---"
```

C.5.3 2) cfmesh Folder

These scripts are used to mesh the geometry, using an "fms" file generated using the Salome software.

Code Listing C.18: allMesh.sh, called by allCellSetup.sh in parent folder - to generate the 3D mesh.

```
#!/bin/bash

echo "--- RUNNING allMesh.sh ---"
##Check OF Version and change to OF2012 to run simulation
   properly
echo Current OFversion - $OFVERSION
if [[ "$OFVERSION" != "2112" ]]
then
    source ~/OpenFOAM/OpenFOAM-v2112/etc/bashrc
    OFVERSION=2112
    #echo $OFVERSION
    #export $OFVERSION
fi

echo Loaded OFversion for this run - $OFVERSION; echo

##Run simulation

rm -rf constant/polyMesh
rm -f log.*
rm -rf 1/ #Remove renumbered folder if it exists. This messes
   with the paraview visualization if it is not removed
cartesianMesh | tee log.cartesianMesh | grep 'Number of bad\|This
    may impair\|Number of newly generated\|ClockTime'
echo "cartesianMesh complete"
checkMesh -allTopology -allGeometry | tee log.checkMesh | grep '
   Checking\|Failed\|*\|cells:\|<<Writing\|orthogon'
echo "checkMesh complete"
echo "--- allMesh.sh FINISHED ---"

echo
bash allCopy.sh
```

cfmesh Folder

Code Listing C.19: all2D-Mesh.sh, called by allCellSetup.sh in parent folder - to generate the 2D mesh.

```
#!/bin/bash

echo "--- RUNNING all2D-Mesh.sh ---"
##Check OF Version and change to OF2012 to run simulation
   properly
echo "> Current OFversion - $OFVERSION"
if [[ "$OFVERSION" != "2112" ]]
then
   source ~/OpenFOAM/OpenFOAM-v2112/etc/bashrc
   OFVERSION=2112
   #echo $OFVERSION
   #export $OFVERSION
fi

echo "> Loaded OFversion for this run - $OFVERSION"; echo

##Run simulation

rm -rf constant/polyMesh
rm -f log.*
rm -rf 1/ #Remove renumbered folder if it exists. This messes
   with the paraview visualization if it is not removed
echo "> Starting cartesian2DMesh"
cartesian2DMesh 2>&1 | tee log.cartesian2DMesh | grep 'ClockTime
   \|memory\|stack violation\|Iteration:\|Adding'
echo "> cartesian2DMesh complete"; echo
echo "> Starting checkMesh"
checkMesh -allTopology -allGeometry 2>&1 | tee log.checkMesh |
   grep 'Checking\|Failed\|cells:\|<<Writing\|orthogon'
echo "> checkMesh complete";
echo "--- all2D-Mesh.sh FINISHED ---"

echo
bash allCopy.sh
```

cfmesh Folder

Code Listing C.20: allCopy.sh, called by allMesh.sh and all2D-Mesh.sh - to renumber the mesh and copy the mesh to the analysis folder.

```
#!/bin/bash

echo "--- RUNNING allCopy.sh ---"
rm -rf 1/ #Delete old renumbered mesh if such a folder exists
echo "> Renumbering mesh..."

source ~/OpenFOAM/OpenFOAM-9/etc/bashrc
renumberMesh | tee log.renumberMesh |  grep 'Writing'

echo "> Mesh has been renumbered"

rm -rf ../analysis/constant/polyMesh
cp -r 1/polyMesh ../analysis/constant/
#cp -r constant/polyMesh ../analysis/constant/

echo "> Mesh has been copied to the analysis folder"
echo "--- allCopy.sh FINISHED ---"
```

C.5.4 3) developMap Folder

These scripts are for setting up the developed flow profile, using data from previously completed simulations. This workflow requires a mix of Linux bash scripting and Matlab to correlate the previously completed simulations with the current geometry.

Pre-Matlab

Code Listing C.21: allRun.sh, called by allCellSetup.sh in the parent folder - to setup the developed flow profile for the simulation.

```
#!/bin/bash
#PURPOSE:

echo "--- RUNNING allRun.sh ---"

#fmsName=$(find ../cfmesh/ -name "meshDict" -exec grep
   surfaceFile {} + | sed 's/.*HDF\///g' | sed 's/.fms.*//g') #
   Get name of FMS file
#vel=$(basename $(dirname $(pwd))| sed 's/.*case2_//g' | sed 's/
   ms_.*/ms/g') #Get simulation velocity from folder name

#export fmsName
#export vel

#Numdiff compare sorted C values based on a tolerance. If no
   error, matlab procedure will complete just fine
if [[ "$1" == "" ]]
then
    tol=2e-4
else
    tol=$1
fi

bash allClean.sh

echo $fmsName > fmsName
echo $vel     > vel

bash allProcessC.sh $tol
bash allMatlabSetup.sh

echo "--- allRun.sh FINISHED ---"
```

developMap Folder: Pre-Matlab

Code Listing C.22: allClean.sh, called by allRun.sh - to reset the developMap folder.

```
#!/bin/bash
#PURPOSE:

echo "--- RUNNING allClean.sh ---"

rm -f log.*
rm -f C
rm -f trimmed*
rm -f wHeader*
rm -rf matlab/
rm -f fmsName
rm -f vel
rm -rf final/

echo "--- allClean.sh FINISHED ---"
```

developMap Folder: Pre-Matlab

Code Listing C.23: allProcessC.sh, called by allRun.sh - to prepare the data for rearrangement in Matlab.

```
#!/bin/bash
#PURPOSE: To prepare data for rearrangement in Matlab. Also
   preforms an initial | sort check of the mesh to make sure it
   matches from a preliminary standpoint

# $1 is a tolerance parameter for numdiff to use. Default is 1e-4

echo "--- RUNNING allProcessC.sh ---"

echo Current OFversion - $OFVERSION
if [[ "$OFVERSION" != "9" ]]
then
    source ~/OpenFOAM/OpenFOAM-9/etc/bashrc
    OFVERSION=9
fi
echo Loaded OFversion for this run - $OFVERSION; echo

#Process C values
cd ../analysis
echo " - Processing this simulation's mesh -"
echo "> Generating C in analysis/0/"
postProcess -func writeCellCentres -latestTime 2>&1 | tee ../
   developMap/log.postProcess    | grep -B 1 'Warning' >> ../
   developMap/log.warnings
cd ../
echo "> Copying C to developMap/"
find ./analysis/* -name "C" | xargs cp -t developMap/
#fmsName=$(find ./cfmesh/ -name "meshDict" -exec grep surfaceFile
    {} + | sed 's/.*HDF\///g' | sed 's/.fms.*//g')
cd developMap

#Trim C value for this simulation (specific FMS file)
echo "> Trimming C values to bare XYZ coordinates"
sed -n '/cross_inlet/,/cross_outlet/p'    C > wHeader_"$fmsName"
   _C
cellCount=$(sed '5!d' wHeader_"$fmsName"_C); echo cellCount =
   $cellCount
endOfFile=$(echo $cellCount+6 | bc)
#sed '1202,$d' wHeader_"$fmsName"_C | sed 1,6d | sed 's/[()]//g'
   > trimmed_"$fmsName"_C
sed "${endOfFile}q;$d" wHeader_"$fmsName"_C | sed 1,6d | sed 's
   /[()]//g' > trimmed_"$fmsName"_C
```

```
#Preliminary test of trimmed C values, numdiff compare the sorted
    lists

#Get simulation velocity from folder name
#vel=$(basename $(dirname $(pwd))| sed 's/.*case2_//g' | sed 's/
   ms_.*/ms/g')

echo; echo " - Processing the flowDev_$vel mesh -"
echo "> Copying flowDev_$vel C result to developMap"
cp ~/OpenFOAM/myWork/global/developedFlow/100m_results/$vel*/
   trimmed/trimmed_dF_$vel*C .

echo; echo " - Processing the resulting trimmed_*C values"
echo "> Removing XZ coordinates -- *y"
sed -r 's/(\s+)?\S+//3' trimmed_"$fmsName"_C |  sed -r 's/(\s+)?\
   S+//1' | tr -d ' ' > trimmed_"$fmsName"_C_y
sed -r 's/(\s+)?\S+//3' trimmed_dF_$vel*C   |  sed -r 's/(\s+)?\S
   +//1' | tr -d ' ' > trimmed_dF_"$vel"_C_y

echo "> Sorting by Y coordinate -- *sorted"
sort trimmed_"$fmsName"_C_y > trimmed_"$fmsName"_C_y_sorted
sort trimmed_dF_"$vel"_C_y  > trimmed_dF_"$vel"_C_y_sorted

#Numdiff compare sorted C values based on a tolerance. If no
   error, matlab procedure will complete just fine
if [[ "$1" == "" ]]
then
    tol=1e-4
else
    tol=$1
fi

echo; echo "> Checking numdiff on sorted C values"
numdiff -a $tol trimmed_"$fmsName"_C_y_sorted  trimmed_dF_"$vel"
   _C_y_sorted | tee log.numdiffTol_$tol
numdiff         trimmed_"$fmsName"_C_y_sorted  trimmed_dF_"$vel"
   _C_y_sorted >     log.numdiff

echo; echo "$(cat log.numdiff | grep Absolute | wc -l) values
   were detected when analyzed without a prescribed tolerance
   limit of $tol"; echo

echo "--- allProcessC.sh FINISHED ---"
```

developMap Folder: Pre-Matlab

Code Listing C.24: allMatlabSetup.sh, called by allRun.sh - to update the local matlab folder.

```
#!/bin/bash

source ~/.bash_aliases
shopt -s expand_aliases

# Run this script from x11vnc, in simulation parent directory
# PURPOSE: Delete existing folder, then COPY matlab files to
   current directory
echo "--- RUNNING allMatlabSetup.sh ---"
echo "> Copying files from global-base-case/developMap to here"
#rm -rf matlab
rm -f startup.m
install -d -m 0755 -o dksan -g dksan matlab

rsync -a    /home/dksan/OpenFOAM/myWork/global-base-case/
   developMap/matlab/202*/*        matlab/
rsync -a    /home/dksan/OpenFOAM/myWork/global-base-case/
   developMap/202*/startup.m            .

rsync -a    ~/OpenFOAM/myWork/global/developedFlow/100m_results/
   $vel*/trimmed/trimmed_dF*     matlab/beforeProcess/
rsync -a    trimmed_"$fmsName"_C

   matlab/beforeProcess/trimmed_fms_C
rsync -a    trimmed_"$fmsName"_C_y

   matlab/beforeProcess/trimmed_fms_C_y
rsync -a    trimmed_dF_"$vel"_C_y

   matlab/beforeProcess/

echo; echo ">> Open matlab when you are ready. Then run
   allFinalInletData.sh"; echo
#matlab
echo "--- allMatlabSetup.sh FINISHED ---"
```

Matlab

Code Listing C.25: allSortData.m, to sort the original developed flow profile with respect to the current mesh construction.

```
% Purpose: To sort trimmed_dF flow data with the given dF_C_y
   values w.r.t.
%   the specific geometry's C_y values
clc
clear

if exist('before','var') == 1
    %do nothing
else
    %Var does not exist. Import file
    importData_trimmed
end

% Compare dF_C_y and geometry_C_y w/an allowable tolerance
% Generate an index of values that correlates the C_y* files
   together
tol=2e-4;
orgLength=length(before.C_y_dF{1,1});

% Get row numer of C_y_dF where C_y_FMS(1,2,3...) exists. Save
   row num as index
idx = arrayfun(@(x) find(  before.C_y_dF{1,1} < x+tol & before.
   C_y_dF{1,1} > x-tol ,1)  ...
    ,before.C_y_FMS{1,1}, 'UniformOutput',false);
idxExact = arrayfun(@(x) find(  before.C_y_dF{1,1}==x,1)  ...
    ,before.C_y_FMS{1,1}, 'UniformOutput',false);

% Convert resulting index from cell data type to double
idx=cell2mat(idx);
idx_count=nnz(idx); %Count number of values
% Get duplicate values
[uniqueidx i] = unique(idx, 'first');
indexToDupes = find(not(ismember(1:numel(idx),i)));
clear i

% Process results
if orgLength ~= idx_count
    fprintf('ERROR. Vector length mismatch\n')
    fprintf('Original Length <> Length of index vector\n')
    fprintf('             %.0f <> %.0f\n',orgLength, idx_count)
else
    fprintf('Vector properly sized at %.0f\n',idx_count)
end
fprintf('\n')
```

```
if isempty(indexToDupes)
    fprintf('No duplicate values\n')
else
    fprintf('\nERROR. Duplicate values!! See the following rows:\
       n')
    fprintf(' > %.0f \n',indexToDupes)
end
fprintf('\n')


% Sort dF_flow values w.r.t. index
after.C_test=before.C_y_dF{1,1}(idx);
after.k=before.k_dF{1,1}(idx);
after.nut=before.nut_dF{1,1}(idx);
after.omega=before.omega_dF{1,1}(idx);
after.U=before.U_dF{1,1}(idx,:);
% after.vorticity=before.vorticity_dF{1,1}(idx,:);
% after.wallShearStress=before.wallShearStress_dF{1,1}(idx,:);

% > Test this logic by sorting C_y_dF. Result of max/min(C_y_FMS-
   C_y_dF)
% should be very small
diff=after.C_test-before.C_y_FMS{1,1};
fprintf('Max of diff: %e\n',max(abs(diff)))
fprintf('Min of diff: %e\n',min(abs(diff)))

fprintf('\nMax of U_x vector: %f\n',max(after.U(:,1)))
fprintf('Min of U_x vector: %f\n',min(after.U(:,1)))

% Export data to postProcessed
writematrix(after.C_test,        './matlab/postProcessed/
   C_y_dF_idxed')
writematrix(after.k,             './matlab/postProcessed/k')
writematrix(after.nut,           './matlab/postProcessed/nut')
writematrix(after.omega,         './matlab/postProcessed/omega')
writematrix(after.U,             './matlab/postProcessed/U','
   Delimiter','space')
% writematrix(after.vorticity,     './matlab/postProcessed/
   vorticity','Delimiter','space')
% writematrix(after.wallShearStress,'./matlab/postProcessed/
   wallShearStress','Delimiter','space')
```

Post-Matlab

Code Listing C.26: allFinalInletData.sh, to format the Matlab results with OpenFOAM headers and footers, and copying the data to the repository.

```
#!/bin/bash
# PURPOSE: To add the header/footer data to the raw .txt files

echo "--- RUNNING allFinalInletData.sh ---"

fmsName=$(    cat ../cfmesh/system/meshDict | sed -n '/FMS/p' |
   sed 's/.*files_FMS-HDF\///' | sed 's/.fms";//')
fmsCellSize=$(       cat ../cfmesh/system/meshDict | sed -n '/
   maxCellSize/p' | cut -c15- | sed 's/.$//')
fmsLayersCase2=$(    cat ../cfmesh/system/meshDict | sed -n '/
   case2_.*nLayers/p'          | sed 's/.*nLayers //' | sed 's/;.*
   thickness.*//')
fmsLayersGround=$(   cat ../cfmesh/system/meshDict | sed -n '/
   ground_surface.*nLayers/p' | sed 's/.*nLayers //' | sed 's/;.*
   thickness.*//')
fmsNameComposite=$(echo ""$fmsName"_C-"$fmsCellSize"_Layers_c2-"
   $fmsLayersCase2"_g-"$fmsLayersGround"")
vel=$(       basename $(dirname $(pwd)) | sed 's/.*_W/W/' | sed 's
   /.*_A/A/' | sed 's/_UniformFlow.*//' | sed 's/_DevelopedFlow
   .*//')

rm -rf   final
mkdir    final
cp -r    matlab/postProcessed/* final/
rm       final/C_y_dF_idxed.txt

echo " -- Processing scalars"
##########################################
echo " - Processing k.txt -"
sed -i '1i\
    cross_inlet\
    {\
        type fixedValue;\
        value nonuniform List<scalar>\
1186\
(
' final/k.txt
sed -i '$a\
)\
;\
}
' final/k.txt

##########################################
```

```
echo " - Processing nut.txt -"
sed -i '1i\
    cross_inlet\
    {\
        type fixedValue;\
        value nonuniform List<scalar>\
1186\
(
' final/nut.txt
sed -i '$a\
)\
;\
}
' final/nut.txt

##############################################
echo " - Processing omega.txt -"
sed -i '1i\
    cross_inlet\
    {\
        type fixedValue;\
        value nonuniform List<scalar>\
1186\
(
' final/omega.txt
sed -i '$a\
)\
;\
}
' final/omega.txt


echo " -- Processing vectors"
##############################################
echo " - Processing U.txt -"
sed -i  's/$/)/'    final/U.txt
sed -i  's/^/(/'    final/U.txt
sed -i  's/\r//g'   final/U.txt
sed -i '1i\
    cross_inlet\
    {\
        type fixedValue;\
        value nonuniform List<vector>\
1186\
(
' final/U.txt
sed -i '$a\
)\
;\
}
```

```
' final/U.txt

##############################################
#echo " - Processing vorticity.txt -"
#sed -i 's/$)/' final/vorticity.txt
#sed -i 's/^(/' final/vorticity.txt
#sed -i 's/\r//g' final/vorticity.txt
#sed -i '1i\
#    cross_inlet\
#    {\
#        type fixedValue;\
#        value nonuniform List<vector>\
#995\
#(
#' final/vorticity.txt
#sed -i '$a\
#)\
#;\
#}
#' final/vorticity.txt

##############################################
#echo " - Processing wallShearStress.txt -"
#sed -i 's/$)/' final/wallShearStress.txt
#sed -i 's/^(/' final/wallShearStress.txt
#sed -i 's/\r//g' final/wallShearStress.txt
#sed -i '1i\
#    cross_inlet\
#    {\
#        type fixedValue;\
#        value nonuniform List<vector>\
#995\
#(
#' final/wallShearStress.txt
#sed -i '$a\
#)\
#;\
#}
#' final/wallShearStress.txt

echo "> Copying mapped velocity data"
mkdir                          ~/OpenFOAM/myWork/global/geometry/
   $fmsNameComposite/$vel/
cp final/*                     ~/OpenFOAM/myWork/global/geometry/
   $fmsNameComposite/$vel/
find ../analysis/0 -type f -name "C*"                              -
   exec rm -rf {} +

echo "--- allFinalInletData.sh FINISHED ---"
```

C.5.5 4) matlab Folder

Each simulation was setup such that the data could be analyzed locally, looking at a single simulation. The following contains the matlab codes required for importing the data, pre-processing the data, and generating results.

Import Data

Code Listing C.27: clearImportedVars.m, clear unused variables from importing.

```
%Clear unused variables from importing

clear d_*
```

Code Listing C.28: clearUnusedVars.m, clear unused variables from importing.

```
%Clear unused variables from importing

clear sts*
clear fileID*
clear helper*
clear numProbes
clear columnID
clear ii*
clear headerLines*
```

matlab Folder: Import Data

Code Listing C.29: importData_forceCoefficients.m, to import OpenFOAM force coefficient data for 2D simulations.

```
headerLines_fC = 9;

%Dynamically save the folder location irrespective of time folder
helper_fC_dir = dir(fullfile('./analysis/postProcessing/
   forceCoeffsIncompressible','**/forceCoeffs.dat'));
helper_fC_dir1 = strcat(helper_fC_dir.folder,'/',helper_fC_dir.
   name);
%Open folder
fileID1fC = fopen(helper_fC_dir1,'r'); %Open file for 'r' reading
fileID2fC = fopen(helper_fC_dir1,'r'); %Open file for 'r' reading
%Save data
helper_fC_header = textscan(fileID1fC, '%s %s %s %s %s %s %s', '
   CollectOutput', true); %textscan(open fileID,%s for text/cell
   array..do this 13 times,and 'CollectOutput' true to
   concatenate the results into one variable'
helper_fC_data = textscan(fileID2fC,'%f %f %f %f %f %f', '
   HeaderLines', headerLines_fC, 'CollectOutput', true);
%Close folder for Matlab's sanity
sts1fC = fclose(fileID1fC);
sts2fC = fclose(fileID2fC);

%Variable to grep data based on specific patch name
helper_columns = {'Cm' 'C_d' 'Cl' 'Cl_f' 'Cl_r'};

for ii=1:size(helper_columns,2)
    d_forceCoefficients.data.(char(helper_columns{ii})) =
       helper_fC_data{1,1}(:,ii+1);     %Using pulled indicies,
       pull out data for each patch and store in it's own
       structure
end
d_forceCoefficients.data.time = helper_fC_data{1,1}(:,1);
d_forceCoefficients.header = helper_fC_header{1}(1:headerLines_fC
   ,:);
    %LATER: Change the 5 to a var. Either dynamically set or user
        set the number of header lines.
    %Could dynamically calculate this based on the number of '#'s
        in a file (since this is a Linux comment)
```

matlab Folder: Import Data

Code Listing C.30: importData_forceIncompressible_GCM.m, to import OpenFOAM force coefficient data for 3D simulations.

```
headerLines_f = 3;

% 1: Case2

%Dynamically save the folder location irrespective of time folder
helper_f1_dir = dir(fullfile('./analysis/postProcessing/
    forceIncompressible-GCM','**/forces.dat'));
helper_f1_dir1 = strcat(helper_f1_dir.folder,'/',helper_f1_dir.
    name);
%Open folder
fileID1_f1 = fopen(helper_f1_dir1,'r'); %Open file for 'r'
    reading
fileID2_f1 = fopen(helper_f1_dir1,'r'); %Open file for 'r'
    reading
%Save data
helper_f1_header = textscan(fileID1_f1, '%s %s %s %s %s %s %s', '
    CollectOutput', true); %textscan(open fileID,%s for text/cell
    array..do this 13 times,and 'CollectOutput' true to
    concatenate the results into one variable'
helper_f1_data = textscan(fileID2_f1,'%f ((%f %f %f) (%f %f %f))
    ((%f %f %f) (%f %f %f))', 'HeaderLines', headerLines_f, '
    CollectOutput', true);
%Close folder for Matlab's sanity
sts_ID1_f1 = fclose(fileID1_f1);
sts_ID2_f1 = fclose(fileID2_f1);

%Variable to grep data based on specific patch name
helper_columns_f1 = {'F_p_x' 'F_p_y' 'F_p_z' 'F_visc_x' 'F_visc_y
    ' 'F_visc_z' 'M_p_x' 'M_p_y' 'M_p_z' 'M_visc_x' 'M_visc_y' '
    M_visc_z'};

for ii_f1=1:size(helper_columns_f1,2)
    d_forceIncomp_GCM.data.(char(helper_columns_f1{ii_f1})) =
        helper_f1_data{1,1}(:,ii_f1+1);     %Using pulled indicies
        , pull out data for each patch and store in it's own
        structure
end
d_forceIncomp_GCM.data.time = helper_f1_data{1,1}(:,1);
d_forceIncomp_GCM.header = helper_f1_header{1}(1:headerLines_f,:)
    ;
```

matlab Folder: Import Data

Code Listing C.31: importData_forceIncompressible.m, to import OpenFOAM force data.

```
headerLines_f = 3;

% 1: Case2

%Dynamically save the folder location irrespective of time folder
helper_f1_dir = dir(fullfile('./analysis/postProcessing/
    forceIncompressible-case2','**/forces.dat'));
helper_f1_dir1 = strcat(helper_f1_dir.folder,'/',helper_f1_dir.
    name);
%Open folder
fileID1_f1 = fopen(helper_f1_dir1,'r'); %Open file for 'r'
    reading
fileID2_f1 = fopen(helper_f1_dir1,'r'); %Open file for 'r'
    reading
%Save data
helper_f1_header = textscan(fileID1_f1, '%s %s %s %s %s %s %s', '
    CollectOutput', true); %textscan(open fileID,%s for text/cell
    array..do this 13 times,and 'CollectOutput' true to
    concatenate the results into one variable'
helper_f1_data = textscan(fileID2_f1,'%f ((%f %f %f) (%f %f %f))
    ((%f %f %f) (%f %f %f))', 'HeaderLines', headerLines_f, '
    CollectOutput', true);
%Close folder for Matlab's sanity
sts_ID1_f1 = fclose(fileID1_f1);
sts_ID2_f1 = fclose(fileID2_f1);

%Variable to grep data based on specific patch name
helper_columns_f1 = {'F_p_x' 'F_p_y' 'F_p_z' 'F_visc_x' 'F_visc_y
    ' 'F_visc_z' 'M_p_x' 'M_p_y' 'M_p_z' 'M_visc_x' 'M_visc_y' '
    M_visc_z'};

for ii_f1=1:size(helper_columns_f1,2)
    d_forceIncomp_case2.data.(char(helper_columns_f1{ii_f1})) =
        helper_f1_data{1,1}(:,ii_f1+1);     %Using pulled indicies
        , pull out data for each patch and store in it's own
        structure
end
d_forceIncomp_case2.data.time = helper_f1_data{1,1}(:,1);
d_forceIncomp_case2.header = helper_f1_header{1}(1:headerLines_f
    ,:);


%%%% ------------------------------------------------
    %%%%%%%%%%%%%%%%%%%%
```

```
% 2: Bottom_case2

%Dynamically save the folder location irrespective of time folder
helper_f2_dir = dir(fullfile('./analysis/postProcessing/
    forceIncompressible-case2_bottom','**/forces.dat'));
helper_f2_dir1 = strcat(helper_f2_dir.folder,'/',helper_f2_dir.
    name);
%Open folder
fileID1_f2 = fopen(helper_f2_dir1,'r'); %Open file for 'r'
    reading
fileID2_f2 = fopen(helper_f2_dir1,'r'); %Open file for 'r'
    reading
%Save data
helper_f2_header = textscan(fileID1_f2, '%s %s %s %s %s %s %s', '
    CollectOutput', true); %textscan(open fileID,%s for text/cell
    array..do this 13 times,and 'CollectOutput' true to
    concatenate the results into one variable'
helper_f2_data = textscan(fileID2_f2,'%f ((%f %f %f) (%f %f %f))
    ((%f %f %f) (%f %f %f))', 'HeaderLines', headerLines_f, '
    CollectOutput', true);
%Close folder for Matlab's sanity
sts_ID1_f2 = fclose(fileID1_f2);
sts_ID2_f2 = fclose(fileID2_f2);

%Variable to grep data based on specific patch name
helper_columns_f2 = {'F_p_x' 'F_p_y' 'F_p_z' 'F_visc_x' 'F_visc_y
    ' 'F_visc_z' 'M_p_x' 'M_p_y' 'M_p_z' 'M_visc_x' 'M_visc_y' '
    M_visc_z'};

for ii_f2=1:size(helper_columns_f2,2)
    d_forceIncomp_bottom_c2.data.(char(helper_columns_f2{ii_f2}))
        = helper_f2_data{1,1}(:,ii_f2+1);     %Using pulled
       indicies, pull out data for each patch and store in it's
       own structure
end
d_forceIncomp_bottom_c2.data.time = helper_f2_data{1,1}(:,1);
d_forceIncomp_bottom_c2.header = helper_f2_header{1}(1:
    headerLines_f,:);

%%%% ------------------------------------------------
    %%%%%%%%%%%%%%%%%%
% 3: Inlet_case2

%Dynamically save the folder location irrespective of time folder
helper_f3_dir = dir(fullfile('./analysis/postProcessing/
    forceIncompressible-case2_inlet','**/forces.dat'));
helper_f3_dir1 = strcat(helper_f3_dir.folder,'/',helper_f3_dir.
    name);
%Open folder
```

```
fileID1_f3 = fopen(helper_f3_dir1,'r'); %Open file for 'r'
    reading
fileID2_f3 = fopen(helper_f3_dir1,'r'); %Open file for 'r'
    reading
%Save data
helper_f3_header = textscan(fileID1_f3, '%s %s %s %s %s %s %s', '
    CollectOutput', true); %textscan(open fileID,%s for text/cell
    array..do this 13 times,and 'CollectOutput' true to
    concatenate the results into one variable'
helper_f3_data = textscan(fileID2_f3,'%f ((%f %f %f) (%f %f %f))
    ((%f %f %f) (%f %f %f))', 'HeaderLines', headerLines_f, '
    CollectOutput', true);
%Close folder for Matlab's sanity
sts_ID1_f3 = fclose(fileID1_f3);
sts_ID2_f3 = fclose(fileID2_f3);

%Variable to grep data based on specific patch name
helper_columns_f3 = {'F_p_x' 'F_p_y' 'F_p_z' 'F_visc_x' 'F_visc_y
    ' 'F_visc_z' 'M_p_x' 'M_p_y' 'M_p_z' 'M_visc_x' 'M_visc_y' '
    M_visc_z'};

for ii_f3=1:size(helper_columns_f3,2)
    d_forceIncomp_inlet_c2.data.(char(helper_columns_f3{ii_f3}))
      = helper_f3_data{1,1}(:,ii_f3+1);     %Using pulled
      indicies, pull out data for each patch and store in it's
      own structure
end
d_forceIncomp_inlet_c2.data.time = helper_f3_data{1,1}(:,1);
d_forceIncomp_inlet_c2.header = helper_f3_header{1}(1:
    headerLines_f,:);

%%%% ------------------------------------------------
    %%%%%%%%%%%%%%%%%%%
% 4: Top_case2

%Dynamically save the folder location irrespective of time folder
helper_f4_dir = dir(fullfile('./analysis/postProcessing/
    forceIncompressible-case2_top','**/forces.dat'));
helper_f4_dir1 = strcat(helper_f4_dir.folder,'/',helper_f4_dir.
    name);
%Open folder
fileID1_f4 = fopen(helper_f4_dir1,'r'); %Open file for 'r'
    reading
fileID2_f4 = fopen(helper_f4_dir1,'r'); %Open file for 'r'
    reading
%Save data
helper_f4_header = textscan(fileID1_f4, '%s %s %s %s %s %s %s', '
    CollectOutput', true); %textscan(open fileID,%s for text/cell
    array..do this 13 times,and 'CollectOutput' true to
    concatenate the results into one variable'
```

```
helper_f4_data = textscan(fileID2_f4,'%f ((%f %f %f) (%f %f %f))
   ((%f %f %f) (%f %f %f))', 'HeaderLines', headerLines_f, '
   CollectOutput', true);
%Close folder for Matlab's sanity
sts_ID1_f4 = fclose(fileID1_f4);
sts_ID2_f4 = fclose(fileID2_f4);

%Variable to grep data based on specific patch name
helper_columns_f4 = {'F_p_x' 'F_p_y' 'F_p_z' 'F_visc_x' 'F_visc_y
   ' 'F_visc_z' 'M_p_x' 'M_p_y' 'M_p_z' 'M_visc_x' 'M_visc_y' '
   M_visc_z'};

for ii_f4=1:size(helper_columns_f4,2)
    d_forceIncomp_top_c2.data.(char(helper_columns_f4{ii_f4})) =
       helper_f4_data{1,1}(:,ii_f4+1);     %Using pulled indicies
       , pull out data for each patch and store in it's own
       structure
end
d_forceIncomp_top_c2.data.time = helper_f4_data{1,1}(:,1);
d_forceIncomp_top_c2.header = helper_f4_header{1}(1:headerLines_f
   ,:);

%%%% ------------------------------------------------
   %%%%%%%%%%%%%%%%%%
% 5: Outlet_case2

%Dynamically save the folder location irrespective of time folder
helper_f5_dir = dir(fullfile('./analysis/postProcessing/
   forceIncompressible-case2_outlet','**/forces.dat'));
helper_f5_dir1 = strcat(helper_f5_dir.folder,'/',helper_f5_dir.
   name);
%Open folder
fileID1_f5 = fopen(helper_f5_dir1,'r'); %Open file for 'r'
   reading
fileID2_f5 = fopen(helper_f5_dir1,'r'); %Open file for 'r'
   reading
%Save data
helper_f5_header = textscan(fileID1_f5, '%s %s %s %s %s %s %s', '
   CollectOutput', true); %textscan(open fileID,%s for text/cell
   array..do this 13 times,and 'CollectOutput' true to
   concatenate the results into one variable'
helper_f5_data = textscan(fileID2_f5,'%f ((%f %f %f) (%f %f %f))
   ((%f %f %f) (%f %f %f))', 'HeaderLines', headerLines_f, '
   CollectOutput', true);
%Close folder for Matlab's sanity
sts_ID1_f5 = fclose(fileID1_f5);
sts_ID2_f5 = fclose(fileID2_f5);

%Variable to grep data based on specific patch name
```

```
helper_columns_f5 = {'F_p_x' 'F_p_y' 'F_p_z' 'F_visc_x' 'F_visc_y
   ' 'F_visc_z' 'M_p_x' 'M_p_y' 'M_p_z' 'M_visc_x' 'M_visc_y' '
   M_visc_z'};

for ii_f5=1:size(helper_columns_f5,2)
    d_forceIncomp_outlet_c2.data.(char(helper_columns_f5{ii_f5}))
        = helper_f5_data{1,1}(:,ii_f5+1);     %Using pulled
       indicies, pull out data for each patch and store in it's
       own structure
end
d_forceIncomp_outlet_c2.data.time = helper_f5_data{1,1}(:,1);
d_forceIncomp_outlet_c2.header = helper_f5_header{1}(1:
   headerLines_f,:);
```

matlab Folder: Import Data

Code Listing C.32: importData_profileYaxis.m, to import OpenFOAM data collected along a line.

```
function import_yProfile = importData_profileYaxis(time)
    if exist(strcat('./analysis/postProcessing/
       profile_yAxis_ustar_ystar/',num2str(time)), 'dir')
        %fileID2fC = fopen('./analysis/postProcessing/
           profile_yAxis_ustar_ystar/time/line_wallShearStress_U.
           xy','r'); %Open file for 'r' reading
        fileID2fC = fopen(strcat('./analysis/postProcessing/
           profile_yAxis_ustar_ystar/',num2str(time),'/
           line_wallShearStress_U.xy'),'r'); %Open file for 'r'
           reading
        helper_fC_data = textscan(fileID2fC,'%f %f %f %f %f %f %f
           ', 'CollectOutput', true);
        %Close folder for Matlab's sanity
        sts2fC = fclose(fileID2fC);

        %Variable to grep data based on specific patch name
        helper_columns = {'CellCenter' 'WSSx' 'WSSy' 'WSSz' 'Ux'
           'Uy' 'Uz'};
        sort = 1;

    elseif exist(strcat('./analysis/postProcessing/
       profile_yAxisUniform_ustar_ystar_ddt/',num2str(time)), '
       dir')
        fileID2fC = fopen(strcat('./analysis/postProcessing/
           profile_yAxisUniform_ustar_ystar_ddt/',num2str(time)
           ,'/line_wallShearStress_U_ddt0(U).xy'),'r'); %Open
           file for 'r' reading
        helper_fC_data = textscan(fileID2fC,'%f %f %f %f %f %f %f
            %f %f %f', 'CollectOutput', true);
        %Close folder for Matlab's sanity
        sts2fC = fclose(fileID2fC);

        %Variable to grep data based on specific patch name
        helper_columns = {'CellCenter' 'WSSx' 'WSSy' 'WSSz' 'Ux'
           'Uy' 'Uz' 'ddt0Ux' 'ddt0Uy' 'ddt0Uz'};
        sort = 1;

    elseif  exist(strcat('./analysis/postProcessing/
       profile_yAxisUniform_ustar_ystar_one/',num2str(time)), '
       dir')
        fileID1fC = fopen(strcat('./analysis/postProcessing/
           profile_yAxisUniform_ustar_ystar_one/',num2str(time)
           ,'/line_wallShearStress_U_ddt0(U).xy'),'r'); %Open
           file for 'r' reading
```

```
    helper1_fC_data = textscan(fileID1fC,'%f %f %f %f %f %f %
       f %f %f %f', 'CollectOutput', true);
    %Close folder for Matlab's sanity
    sts1fC = fclose(fileID1fC);

    fileID2fC = fopen(strcat('./analysis/postProcessing/
       profile_yAxisUniform_ustar_ystar_two/',num2str(time)
       ,'/line_wallShearStress_U_ddt0(U).xy'),'r'); %Open
       file for 'r' reading
    helper2_fC_data = textscan(fileID2fC,'%f %f %f %f %f %f %
       f %f %f %f', 'CollectOutput', true);
    %Close folder for Matlab's sanity
    sts2fC = fclose(fileID2fC);

    helper_fC_data = cat(1,helper1_fC_data,helper2_fC_data);

    %Variable to grep data based on specific patch name
    helper_columns = {'CellCenter' 'WSSx' 'WSSy' 'WSSz' 'Ux'
       'Uy' 'Uz' 'ddt0Ux' 'ddt0Uy' 'ddt0Uz'};
    sort = 2;

elseif  exist(strcat('./analysis/postProcessing/
   xyzAxisUniform_ustar_ystar_a/',num2str(time)), 'dir')
    fileID1fC = fopen(strcat('./analysis/postProcessing/
       xyzAxisUniform_ustar_ystar_a/',num2str(time),'/
       line_wallShearStress_U_ddt0(U).xy'),'r'); %Open file
       for 'r' reading
    helper1_fC_data = textscan(fileID1fC,'%f %f %f %f %f %f %
       f %f %f %f %f %f', 'CollectOutput', true);
    %Close folder for Matlab's sanity
    sts1fC = fclose(fileID1fC);

    fileID2fC = fopen(strcat('./analysis/postProcessing/
       xyzAxisUniform_ustar_ystar_b/',num2str(time),'/
       line_wallShearStress_U_ddt0(U).xy'),'r'); %Open file
       for 'r' reading
    helper2_fC_data = textscan(fileID2fC,'%f %f %f %f %f %f %
       f %f %f %f %f %f', 'CollectOutput', true);
    %Close folder for Matlab's sanity
    sts2fC = fclose(fileID2fC);

    fileID3fC = fopen(strcat('./analysis/postProcessing/
       xyzAxisUniform_ustar_ystar_c/',num2str(time),'/
       line_wallShearStress_U_ddt0(U).xy'),'r'); %Open file
       for 'r' reading
    helper3_fC_data = textscan(fileID3fC,'%f %f %f %f %f %f %
       f %f %f %f %f %f', 'CollectOutput', true);
    %Close folder for Matlab's sanity
    sts3fC = fclose(fileID3fC);
```

```
    helper_fC_data = cat(1,helper1_fC_data,helper2_fC_data,
       helper3_fC_data);

    %Variable to grep data based on specific patch name
    helper_columns = {'xDist' 'yDist' 'zDist' 'WSSx' 'WSSy' '
       WSSz' 'Ux' 'Uy' 'Uz' 'ddt0Ux' 'ddt0Uy' 'ddt0Uz'};
    sort = 2;

else
    fprintf('ERROR - cannot open requested yProfile folder')
end


if sort == 1
    for ii=1:size(helper_columns,2)
        d_yProfile.(char(helper_columns{ii})) =
           helper_fC_data{1,1}(:,ii);     %Using pulled
           indicies, pull out data for each patch and store
           in it's own structure
    end
    import_yProfile = d_yProfile;

elseif sort == 2 %Import data that is broken into 2 lines for
    data refinement
    for ii=1:size(helper_columns,2)
        helper1.(char(helper_columns{ii})) = helper_fC_data
           {1,1}(:,ii);     %Using pulled indicies, pull out
           data for each patch and store in it's own
           structure
        helper2.(char(helper_columns{ii})) = helper_fC_data
           {2,1}(:,ii);     %Using pulled indicies, pull out
           data for each patch and store in it's own
           structure
        helper3.(char(helper_columns{ii})) = helper_fC_data
           {3,1}(:,ii);     %Using pulled indicies, pull out
           data for each patch and store in it's own
           structure
       %helper2.CellCenter(:) = helper2.CellCenter(:) + 1;
        helper1.(char(helper_columns{ii}))(end) = [];
        helper2.(char(helper_columns{ii}))(end) = [];
        d_yProfile.(char(helper_columns{ii})) = [helper1.(
           char(helper_columns{ii})); helper2.(char(
           helper_columns{ii})); helper3.(char(helper_columns
           {ii}))];
    end

    %helper_fC_data{1,1}(:,ii);     %Using pulled indicies,
       pull out data for each patch and store in it's own
       structure
```

```
        import_yProfile = d_yProfile;
    else
        fprintf('ERROR - cannot save data into proper format')
        import_yProfile = NaN;
        return
end
```

matlab Folder: Import Data

Code Listing C.33: importData_yPlus.m, to import OpenFOAM yPlus data.

```
headerLines_yPlus = 2;

helper_yPlus_dir = dir(fullfile('./analysis/postProcessing/yPlus
   ','**/yPlus.dat'));
helper_yPlus_dir1 = strcat(helper_yPlus_dir.folder,'/',
   helper_yPlus_dir.name);
fileID1yPlus = fopen(helper_yPlus_dir1,'r'); %Open file for 'r'
   reading
fileID2yPlus = fopen(helper_yPlus_dir1,'r'); %Open file for 'r'
   reading
helper_yPlus_header = textscan(fileID1yPlus, '%s %s %s %s %s %s',
    'CollectOutput', true); %textscan(open fileID,%s for text/
   cell array..do this 13 times,and 'CollectOutput' true to
   concatenate the results into one variable'
helper_yPlus_data = textscan(fileID2yPlus,'%f %s %f %f %f', '
   HeaderLines', headerLines_yPlus, 'CollectOutput', true);
sts1yPlus = fclose(fileID1yPlus);
sts2yPlus = fclose(fileID2yPlus);

%Variable to grep data based on specific patch name
a = sum(any(strcmp(helper_yPlus_header{1,1},'case2_top')));
if a == 1 %Checks if this data includes the case2_body
    %If yes, define
    helper_patches2 = {'case2_bottom' 'case2_inlet' 'case2_outlet
       ' 'case2_top' 'cross_inlet' 'cross_outlet' 'atmosphere' '
       ground_surface'};
else
    %Otherwise, define the general boundary patches
    helper_patches2 = {'cross_inlet' 'cross_outlet' 'atmosphere'
       'ground_surface'};
end


for ii=1:size(helper_patches2,2)
    helper_patch_rows(:,ii) = find(strcmp(helper_yPlus_data
       {1,2}(:,1),helper_patches2(1,ii)));          %Pull
       indicies that match helper_patches
    d_yPlus.data.(char(helper_patches2{ii})) = helper_yPlus_data
       {1,3}(helper_patch_rows(:,ii),1:3);     %Using pulled
       indicies, pull out data for each patch and store in it's
       own structure
end
d_yPlus.data.time = helper_yPlus_data{1,1}(helper_patch_rows(:,1)
   ,1);
```

```
d_yPlus.header = helper_yPlus_header{1}(1:headerLines_yPlus,:);
    %LATER: Change the 5 to a var. Either dynamically set or user
        set the number of header lines.
    %Could dynamically calculate this based on the number of '#'s
        in a file (since this is a Linux comment)
```

Functions

Code Listing C.34: function_uPlus_yPlus_calc.m, to calculate yPlus and uPlus based on Pope equations.

```
function results = function_uPlus_yPlus_calc(yProfile)
    %Water
    rho_water = 998.2;  % kg/m^3
    nu_water = 1.004e-06;    % m^2/s

    % tau_wall (7.24)
    %tau_wall = rho_water*nu_water*d_yProfile.ddt0Uy(1,1);

    % mu_tau (7.25) - friction velocity
    %mu_tau = sqrt(abs(tau_wall)/rho_water);
    mu_tau = sqrt(abs(yProfile.WSSx(1,1)));

    % y+ (7.28)
    results.yPlus = (mu_tau.*yProfile.yDist)/nu_water;

    % u+ (7.35)
    results.uPlus = yProfile.Ux/mu_tau;

    % u+ from eqn (7.43)
    k = 0.41;
    B = 5.2;
    results.uPlusfromEqn = (1/k)*log(results.yPlus) + B;
end
```

matlab Folder: Functions

Code Listing C.35: transientDataCrop_forceCoefficients.m, to trim the imported dataset to the specified limit times.

```
%PURPOSE: Define time for simulation to settle to a steady state.
    This
%script then crops all imported data based on this specified time


%% Crop data files that pull data at each timeStep

function crop = transientDataCrop_forceCoefficients(
   d_forceCoefficients,transientTmin,transientTmax)
    %transientRowMin = find(d_forceCoefficients.data.time ==
       transientTmin);
    %transientRowMax = find(d_forceCoefficients.data.time ==
       transientTmax);
    [~,transientRowMin] = (min(abs(d_forceCoefficients.data.time
       - transientTmin)));
    [~,transientRowMax] = (min(abs(d_forceCoefficients.data.time
       - transientTmax)));
    transientLength = length(d_forceCoefficients.data.time);
    crop = structfun(@(x) (removerows(x,'ind', [1:transientRowMin
       -1,transientRowMax+1:transientLength])),
       d_forceCoefficients.data, 'UniformOutput', false);
    clear transient*
end
```

matlab Folder: Functions

Code Listing C.36: transientDataCrop_yPlus.m, to trim the imported yPlus dataset to the specified limit times.

```
%PURPOSE: Define time for simulation to settle to a steady state.
    This
%script then crops all imported data based on this specified time


%% Crop data files that pull data at each timeStep

function crop = transientDataCrop_yPlus(d_yPlus,transientTmin,
   transientTmax)
    transientRow2Min = find(d_yPlus.data.time == transientTmin);
    transientRow2Max = find(d_yPlus.data.time == transientTmax);
    transientLength2 = length(d_yPlus.data.time);
    crop = structfun(@(x) (removerows(x,'ind', [1:
       transientRow2Min-1,transientRow2Max+1:transientLength2])),
        d_yPlus.data, 'UniformOutput', false);
    clear transient*
end
```

Matlab Files applicable to each simulation

Code Listing C.37: fft_analysis.m, to generate a FFT analysis to determine the Strouhal number.

```
% Future items:
    % With a longer signal, test to see if the resolution of the
       FFT
    % analysis increases

if exist("global_values.m",'file') == 2 %If this folder contains
   a global_values.m file, then run that file
    global_values
else %Otherwise, define relevant parameters
    xmin = 0.5;
    xmax = 1;
end

plot_forceCoeff

%% Values that the user can change
% xmin = 0.635364;
% xmax = 0.8335;
L = 5000;                                % Length of signal
per=.05;        % Percent of total vector length to show in graph
    .
% n = 2^nextpow2(L); % Increase resolution of FFT analysis and
   make it easier for the FFT analysis to compute
n = L;


% Given OpenFOAM signal is sampled at an irregular rate.
% Interpolate the signal to sample at a regular sampling rate
t = linspace (xmin, xmax, L+1);     % Time vector given set xmax/
   xmin values and L

dt = (xmax - xmin) /L;                % Delta T
df = 2*pi / dt;

% OpenFOAM data
y_OF = d_forceCoefficients.data.C_d;
t_OF = d_forceCoefficients.data.time;

y_sampled = interp1(t_OF, y_OF,t,'linear','extrap');   %
   Interpolate OF data onto t vector
Fs = 1/dt;                            % Sampling Frequency

Y = fft(y_sampled,n);
```

```
P2 = abs(Y/L);                    % 2 sided spectrum
P1 = P2(1:n/2+1);                 % 1 sided spectrum
P1(1:end-2) = P1(2:end-1);        % Removing extraneous point in
   column 1

f = Fs*(0:(n/2))/n;               % Frequency domain
% Find maximum amplitude and the correlated frequency (y)
[maxYValue_amplitude, indexAtMaxY_amplitude] = max(P1);
maxFrequency_AtMaxYValue = f(indexAtMaxY_amplitude(1))

if ~exist("currentDirectory","var")
    figure(3)
    plot(t,y_sampled)
    title("OF Data - Interpolated/Sampled")
    xlabel("t (sec)")
    ylabel("C_d")

    figure(4)
    plot(f(1:L*per/2),P1(1:L*per/2))
    title("Single-Sided Amplitude Spectrum of OF Data")
    xlabel("f (Hz)")
    ylabel("|P1(f)|")
    hold on;
    plot(maxFrequency_AtMaxYValue, maxYValue_amplitude, 'r+', '
       MarkerSize', 30);

    % figure(5)
    % autocorr(y_sampled,NumLags=L);
    % title("Autocorrelation of Interpolated OF Data")
    % xlabel("Sample Number (L)")
    % ylabel("Magnitude of Autocorrelation")
    % %Elementwise. .matrix


    % [acf,lags] = xcorr(t,y_sampled);
    % figure(6)
    % plot(acf,lags)
end

clear y_sampled y_OF t_OF df dt f Fs indexAtMaxY_amplitude
   maxYValue_amplitude L n P1 P2 per t Y
```

matlab Folder: Matlab Files applicable to each simulation

Code Listing C.38: NMF_Figure07_DevelopedFlowProfile.m, to generate developed profile figure.

```
clear; clc; clf
% Plot u_star y_star

startTime = 12.4;
endTime = 12.4;
writeInterval = startTime;
timeseries = linspace(startTime,endTime,(endTime-startTime)/
   writeInterval+1);

d_yProfileEnd = importData_profileYaxis(endTime);
resultsEnd = function_uPlus_yPlus_calc(d_yProfileEnd);

for jj=1:size(timeseries,2)
    clf
    fprintf('Time = %2.2f\n',timeseries(1,jj))
    d_yProfile = importData_profileYaxis(timeseries(1,jj));

    results = function_uPlus_yPlus_calc(d_yProfile);

%     export = figure(1);

    figure(1)
    ax = axes();
    hold (ax);
    ylabel(ax,'y/\delta');
    xlabel(ax,'U/u_\tau');
    ylim([0 2])
    grid on
    %ylabel = [0 1];
    plot(results.uPlus,d_yProfile.yDist/0.365)
    %title(ax,'Developed Flow Profile')
    %title(strcat({'Time = '},sprintf('%05.2f',timeseries(1,jj))
       ,' sec'))
    hold off;

%     exportgraphics(export,strcat('./0-figures/m_uPlus_yPlus_',
   sprintf('%04d',jj),'.png'),'Resolution',600)
%     im = imread(strcat('./0-figures/m_uPlus_yPlus_',sprintf
   ('%04d',jj),'.png'));
%     im2 = padarray(im, [100 100], 255);
%     imwrite(im2,strcat('./0-figures/m_uPlus_yPlus_',sprintf
   ('%04d',jj),'.png'));

%     delete(a)
```

```
end
```

matlab Folder: Matlab Files applicable to each simulation

Code Listing C.39: plot_forceCoeff.m, to process and plot the force coefficient data.

```
%Load forceCoefficients data if not already loaded
clear d_* C_* coefficient
clear rho_water
clear RollingMoment MomentArm
clear xmin xmax

if ~exist("currentDirectory","var"); clear;clc;clf; end

if exist("currentDirectory","var") %If this file is being run
   from a parent file in /storage/dksan, such that the var "
   currentDirectory" is created, then:
    global_values
else %Otherwise, define relevant parameters
    if exist("global_values_JFM")
        global_values_JFM
    else
        global_values
    end
    if exist("JFM_points")
        JFM_points
    end
end
coefficient = 'C_d';

if exist('d_forceCoefficients','var') == 1
    %Do nothing
else
    %Var does not exist. Import file
    incompressibleData = isfolder('./analysis/postProcessing/
       forceIncompressible-case2/');
    clear d_forceCoefficients
    clear d_forceIncompressible*

    importData_forceCoefficients
    if incompressibleData
        importData_forceIncompressible
    end

    fprintf('Original datasets:\n')
    fprintf('forceCoeffs                    - Min time: %f, Max time:
       %f\n',d_forceCoefficients.data.time(1,1),
       d_forceCoefficients.data.time(end,1))
    if incompressibleData
```

```
fprintf('forceIncompressible_case2  - Min time: %f, Max time:
    %f\n',d_forceIncomp_case2.data.time(1,1),
   d_forceIncomp_case2.data.time(end,1))
fprintf('forceIncompressible_bottom - Min time: %f, Max time:
    %f\n',d_forceIncomp_bottom_c2.data.time(1,1),
   d_forceIncomp_bottom_c2.data.time(end,1))
fprintf('forceIncompressible_inlet  - Min time: %f, Max time:
    %f\n',d_forceIncomp_inlet_c2.data.time(1,1),
   d_forceIncomp_inlet_c2.data.time(end,1))
fprintf('forceIncompressible_outlet - Min time: %f, Max time:
    %f\n',d_forceIncomp_outlet_c2.data.time(1,1),
   d_forceIncomp_outlet_c2.data.time(end,1))
fprintf('forceIncompressible_top    - Min time: %f, Max time:
    %f\n',d_forceIncomp_top_c2.data.time(1,1),
   d_forceIncomp_top_c2.data.time(end,1))
end

d_forceCoefficients.data =
   transientDataCrop_forceCoefficients(d_forceCoefficients,
   xmin,xmax);
if incompressibleData
d_forceIncomp_case2.data =
   transientDataCrop_forceCoefficients(d_forceIncomp_case2,
   xmin,xmax);
d_forceIncomp_bottom_c2.data =
   transientDataCrop_forceCoefficients(
   d_forceIncomp_bottom_c2,xmin,xmax);
d_forceIncomp_inlet_c2.data =
   transientDataCrop_forceCoefficients(d_forceIncomp_inlet_c2
   ,xmin,xmax);
d_forceIncomp_outlet_c2.data =
   transientDataCrop_forceCoefficients(
   d_forceIncomp_outlet_c2,xmin,xmax);
d_forceIncomp_top_c2.data =
   transientDataCrop_forceCoefficients(d_forceIncomp_top_c2,
   xmin,xmax);
end

fprintf('\nTrimmed datasets:\n')
fprintf('forceCoeffs                - Min time: %f, Max time:
    %f\n',d_forceCoefficients.data.time(1,1),
   d_forceCoefficients.data.time(end,1))
if incompressibleData
fprintf('forceIncompressible_case2  - Min time: %f, Max time:
    %f\n',d_forceIncomp_case2.data.time(1,1),
   d_forceIncomp_case2.data.time(end,1))
fprintf('forceIncompressible_bottom - Min time: %f, Max time:
    %f\n',d_forceIncomp_bottom_c2.data.time(1,1),
   d_forceIncomp_bottom_c2.data.time(end,1))
```

```
    fprintf('forceIncompressible_inlet  - Min time: %f, Max time:
        %f\n',d_forceIncomp_inlet_c2.data.time(1,1),
       d_forceIncomp_inlet_c2.data.time(end,1))
    fprintf('forceIncompressible_outlet - Min time: %f, Max time:
        %f\n',d_forceIncomp_outlet_c2.data.time(1,1),
       d_forceIncomp_outlet_c2.data.time(end,1))
    fprintf('forceIncompressible_top    - Min time: %f, Max time:
        %f\n',d_forceIncomp_top_c2.data.time(1,1),
       d_forceIncomp_top_c2.data.time(end,1))
    end
    fprintf('\n')
    clearUnusedVars
end

d_mean.Cd = mean(d_forceCoefficients.data.C_d);
d_mean.Cm = mean(d_forceCoefficients.data.Cm);
d_mean.Cl = mean(d_forceCoefficients.data.Cl);

fprintf('Mean of Cd = %2.4f\n',d_mean.Cd)
fprintf('Mean of Cm = %2.4f\n',d_mean.Cm)
fprintf('Mean of Cl = %2.4f\n',d_mean.Cl)

if ~exist("currentDirectory","var")
    figure(1)
    boxplot([d_forceCoefficients.data.C_d d_forceCoefficients.
       data.Cl d_forceCoefficients.data.Cl_f d_forceCoefficients.
       data.Cl_r d_forceCoefficients.data.Cm], ...
       {"Cd", "Cl", "Cl_f", "Cl_r", "Cm"})
    hold off
    xlabel('Force Coefficient Type')
    ylabel('Coefficient Value')

    if exist("JFM_points")
        [d,ix] = min( abs( d_forceCoefficients.data.time-
           x_JFM_points));
        JFM_alpha = {'a) ' 'b) ' 'c) ' 'd) '};
        JFM_labels = cellfun(@num2str,num2cell(x_JFM_points),'
           UniformOutput',false);
        JFM_omega = {' s' ' s' ' s' ' s'};
        JFM_labels2 = cellfun( @(x,y,z) [x,y,z], JFM_alpha,
           JFM_labels, JFM_omega,'UniformOutput', false );


        figure(2)
        plot(d_forceCoefficients.data.time,d_forceCoefficients.
           data.(coefficient),"Color",'b')
        hold on
        plot(x_JFM_points,d_forceCoefficients.data.(coefficient)(
           ix,1),'k*')
```

```
        %text(x_JFM_points*1.002,d_forceCoefficients.data.(
            coefficient)(ix,1), JFM_labels2)
        labelpoints(x_JFM_points,d_forceCoefficients.data.(
            coefficient)(ix,1), JFM_labels2,'E',.2,'
            backgroundColor', [1 1 1 .6])
        xlabel('sec')
        ylabel('C_{drag}')
        ylim([0 2.5])
        grid on
        hold off
    else
        figure(2)
        plot(d_forceCoefficients.data.time,d_forceCoefficients.
            data.(coefficient),"Color",'b')
        xlabel('sec')
        ylabel('C_{drag}')
        ylim([0 2.5])
        grid on
    end

    rho_water = 998.2; %kg/m^3

    MomentArm = d_forceIncomp_inlet_c2.data.M_p_z(:,1)./
        d_forceIncomp_inlet_c2.data.F_p_x(:,1);
%     figure
%     plot(d_forceIncomp_inlet_c2.data.time,MomentArm)
    mean(MomentArm)

    C_side = mean(d_forceIncomp_case2.data.F_p_x)/(0.5*rho_water
        *0.364922*1*5.33333333333^2)
    C_lift = mean(d_forceIncomp_case2.data.F_p_y)/(0.5*rho_water
        *0.364922*1*5.33333333333^2)
    RollingMoment = -(d_forceIncomp_case2.data.F_p_x * (0.148507
        + (0.364922/2))) + (d_forceIncomp_case2.data.F_p_y *
        (0.3239/2));
    C_rollover = mean(RollingMoment)/(0.5*rho_water
        *0.364922*1*0.3239*5.33333333333^2)
end
```

matlab Folder: Matlab Files applicable to each simulation

Code Listing C.40: plot_forceGCM.m, to process and plot the force coefficient data.

```
%Load forceCoefficients data if not already loaded
clear d_* C_* coefficient
clear rho_water
clear RollingMoment MomentArm
clear xmin xmax

if ~exist("currentDirectory","var"); clear;clc;clf; end

if exist("global_values.m",'file') == 2 %If this folder contains
   a global_values.m file, then run that file
    global_values
else %Otherwise, define relevant parameters
    xmin = 0.264525;
    xmax = 0.343005;
end
coefficient = 'Cd';

if exist('d_forceCoefficients','var') == 1
    %Do nothing
else
    %Var does not exist. Import file
    incompressibleData = isfolder('./analysis/postProcessing/
       forceIncompressible-GCM/');
    clear d_forceCoefficients
    clear d_forceIncompressible*

    importData_forceCoefficients
    if incompressibleData
        importData_forceIncompressible_GCM
    end

    fprintf('Original datasets:\n')
    fprintf('forceCoeffs                    - Min time: %f, Max time:
       %f\n',d_forceCoefficients.data.time(1,1),
       d_forceCoefficients.data.time(end,1))
    if incompressibleData
    fprintf('forceIncompressible_GCM  - Min time: %f, Max time: %
       f\n',d_forceIncomp_GCM.data.time(1,1),d_forceIncomp_GCM.
       data.time(end,1))
    %fprintf('forceIncompressible_bottom - Min time: %f, Max time
       : %f\n',d_forceIncomp_bottom_c2.data.time(1,1),
       d_forceIncomp_bottom_c2.data.time(end,1))
```

```
%fprintf('forceIncompressible_inlet  - Min time: %f, Max time
    : %f\n',d_forceIncomp_inlet_c2.data.time(1,1),
    d_forceIncomp_inlet_c2.data.time(end,1))
%fprintf('forceIncompressible_outlet - Min time: %f, Max time
    : %f\n',d_forceIncomp_outlet_c2.data.time(1,1),
    d_forceIncomp_outlet_c2.data.time(end,1))
%fprintf('forceIncompressible_top    - Min time: %f, Max time
    : %f\n',d_forceIncomp_top_c2.data.time(1,1),
    d_forceIncomp_top_c2.data.time(end,1))
end

d_forceCoefficients.data =
    transientDataCrop_forceCoefficients(d_forceCoefficients,
    xmin,xmax);
if incompressibleData
d_forceIncomp_GCM.data = transientDataCrop_forceCoefficients(
    d_forceIncomp_GCM,xmin,xmax);
%d_forceIncomp_bottom_c2.data =
    transientDataCrop_forceCoefficients(
    d_forceIncomp_bottom_c2,xmin,xmax);
%d_forceIncomp_inlet_c2.data =
    transientDataCrop_forceCoefficients(d_forceIncomp_inlet_c2
    ,xmin,xmax);
%d_forceIncomp_outlet_c2.data =
    transientDataCrop_forceCoefficients(
    d_forceIncomp_outlet_c2,xmin,xmax);
%d_forceIncomp_top_c2.data =
    transientDataCrop_forceCoefficients(d_forceIncomp_top_c2,
    xmin,xmax);
end

fprintf('\nTrimmed datasets:\n')
fprintf('forceCoeffs                - Min time: %f, Max time:
     %f\n',d_forceCoefficients.data.time(1,1),
    d_forceCoefficients.data.time(end,1))
if incompressibleData
fprintf('forceIncompressible_GCM  - Min time: %f, Max time: %
    f\n',d_forceIncomp_GCM.data.time(1,1),d_forceIncomp_GCM.
    data.time(end,1))
%fprintf('forceIncompressible_bottom - Min time: %f, Max time
    : %f\n',d_forceIncomp_bottom_c2.data.time(1,1),
    d_forceIncomp_bottom_c2.data.time(end,1))
%fprintf('forceIncompressible_inlet  - Min time: %f, Max time
    : %f\n',d_forceIncomp_inlet_c2.data.time(1,1),
    d_forceIncomp_inlet_c2.data.time(end,1))
%fprintf('forceIncompressible_outlet - Min time: %f, Max time
    : %f\n',d_forceIncomp_outlet_c2.data.time(1,1),
    d_forceIncomp_outlet_c2.data.time(end,1))
```

```
    %fprintf('forceIncompressible_top     - Min time: %f, Max time
        : %f\n',d_forceIncomp_top_c2.data.time(1,1),
        d_forceIncomp_top_c2.data.time(end,1))
    end
    fprintf('\n')
    clearUnusedVars
end

d_mean.Cd = mean(d_forceCoefficients.data.Cd);
d_mean.Cm = mean(d_forceCoefficients.data.Cm);
d_mean.Cl = mean(d_forceCoefficients.data.Cl);

fprintf('Mean of Cd = %2.4f\n',d_mean.Cd)
fprintf('Mean of Cm = %2.4f\n',d_mean.Cm)
fprintf('Mean of Cl = %2.4f\n',d_mean.Cl)

if ~exist("currentDirectory","var")
    figure(1)
    boxplot([d_forceCoefficients.data.Cd d_forceCoefficients.data
        .Cl d_forceCoefficients.data.Cl_f d_forceCoefficients.data
        .Cl_r d_forceCoefficients.data.Cm], ...
        {"Cd", "Cl", "Cl_f", "Cl_r", "Cm"})
    hold off
    xlabel('Force Coefficient Type')
    ylabel('Coefficient Value')

    figure(2)
    plot(d_forceCoefficients.data.time,d_forceCoefficients.data.(
        coefficient))
    xlabel('sec')
    ylabel(coefficient)

    rho_water = 998.2; %kg/m^3

    %MomentArm = d_forceIncomp_inlet_c2.data.M_p_z(:,1)./
        d_forceIncomp_inlet_c2.data.F_p_x(:,1);
%     figure
%     plot(d_forceIncomp_inlet_c2.data.time,MomentArm)
    %mean(MomentArm)

    %C_side calc requires forces on inlet face.
%     C_side = mean(d_forceIncomp_GCM.data.F_p_x +
   d_forceIncomp_GCM.data.F_visc_x)...
%         /(0.5*rho_water*0.364922*1*5.33333333333^2)

    C_lift = mean(d_forceIncomp_GCM.data.F_p_y +
        d_forceIncomp_GCM.data.F_visc_y)...
         /(0.5*rho_water*0.364922*1*5.33333333333^2)
```

```
    % C_rollover. Per Baker/Soper 2022. CoR is leeward wheels.
       This is
    % calculated based on the total forces, with an assumed
       center of force
    % at the centroid of the rectangular cylinder.
    RollingMoment = ...
        ((d_forceIncomp_GCM.data.F_p_x + d_forceIncomp_GCM.data.
           F_visc_x)...
            * (0.148507 + (0.364922/2)))...
        + ((d_forceIncomp_GCM.data.F_p_y + d_forceIncomp_GCM.data
           .F_visc_y)...
            * (0.3239/2));
    C_rollover = mean(RollingMoment)...
        /(0.5*rho_water*0.364922^2*1*5.33333333333^2)

    % C_drag. GOOD. THIS IS ALL FORCES, PRESSURE AND VISCOUS IN
       THE X
    % DIRECTION. THIS MATCHES OF'S CALC OF CD. DOES IT MAKE SENSE
        TO
    % INCLUDE ALL FORCES IN THE X DIRECTION? 7/24/23. It makes
       sense to me
    C_drag = mean(d_forceIncomp_GCM.data.F_p_x +
       d_forceIncomp_GCM.data.F_visc_x) ...
        /(0.5*rho_water*0.364922*1*5.33333333333^2)

%     % St Number
%     paraRes.Strouhal(paraRes_i,1) ...
%         = (data_coeffs_unconfined.(char(paraSet.vel(paraRes_i
   ,2))).maxFrequency_AtMaxYValue * paraSet.characteristicLength)
    / paraSet.flow(paraRes_i,1);
end
```

matlab Folder: Matlab Files applicable to each simulation

Code Listing C.41: plot_uPlus_yPlus.m, to plot yPlus and uPlus to generate a nondimensional developed flow profile plot.

```
clear; clc; clf
% Plot u_star y_star

startTime = 12.4;
endTime = 12.4;
writeInterval = startTime;
timeseries = linspace(startTime,endTime,(endTime-startTime)/
   writeInterval+1);

d_yProfileEnd = importData_profileYaxis(endTime);
resultsEnd = function_uPlus_yPlus_calc(d_yProfileEnd);

for jj=1:size(timeseries,2)
    clf
    fprintf('Time = %2.2f\n',timeseries(1,jj))
    d_yProfile = importData_profileYaxis(timeseries(1,jj));

    results = function_uPlus_yPlus_calc(d_yProfile);

    % figure(1)
    % semilogx(results.yPlus,results.uPlus,results.yPlus,results.
       uPlusfromEqn)
    % legend('u^+','u^+ from Eqn','Location','southeast')
    % grid on
    % xlabel('y^+')
    % ylabel('u^+')

    % Setup a figure with y+ and cell height on the same plot:
    % https://www.mathworks.com/matlabcentral/answers/509899-plot
       -with-two-related-x-axes#answer_419278

    export = figure(1);

    % setup bottom axis
    ax = axes();
    hold(ax);
    ax.XAxis.Scale = 'log';
    xlabel(ax,'y^+');
    ylabel(ax,'u^+');

    % setup top axis
    ax_top = axes();
    hold(ax_top);
    ax_top.XAxis.Scale = 'log';
```

```
    ax_top.XAxisLocation = 'top';
    ax_top.YAxisLocation = "right";
    ax_top.YTick = [];
    ax_top.Color = 'none';
    xlabel(ax_top,'Distance from ground surface, y = 0m');

    % linking axis
    linkprop([ax, ax_top],{'Units', 'Position', '
       ActivePositionProperty'});
    ax.Position(4) = ax.Position(4) * .92;

    % configure limits of bottom axis
    ax.XLim = [resultsEnd.yPlus(2) resultsEnd.yPlus(end)];
    ax.YLim = [floor(resultsEnd.uPlus(2)/5)*5 ceil(resultsEnd.
       uPlus(end)/5)*5];
    ax.XAxis.TickLength = [0.02, 0.00];


    % configure limits and labels of top axis
    ax_top.XLim=[d_yProfileEnd.yDist(2) d_yProfileEnd.yDist(end)
       ];
    if d_yProfile.yDist(end) <= 2
        ax_top.XTick = [0 0.001 0.01 0.1 0.5 1 2];
    elseif d_yProfile.yDist(end) <= 12
        ax_top.XTick = [0 0.001 0.01 0.1 0.5 1 2 5 12];
    else
        fprintf('yProfile limits out of bounds. Setup appropriate
            ax_top.XTick settings')
    end
    ax_top.XAxis.MinorTick = 'on';
    %ax_top.XAxis.TickLabels = ax_top.XAxis.TickValues
    ax_top.XAxis.TickLength = [0.02, 0.00];

    semilogx(ax,results.yPlus,results.uPlus,results.yPlus,results
       .uPlusfromEqn)
    %plot(ax_top,d_yProfile.CellCenter,results.uPlus)
    ax.XGrid = 'on';
    ax.YGrid = 'on';
    legend(ax,'u^+ from Developed Flow Profile','u^+ from Pope,
       2000','Location','southeast','Color',[1 1 1])
    title(strcat({'Time = '},sprintf('%05.2f',timeseries(1,jj)),'
        sec'))
    hold off;

%     a = annotation('rectangle',[0 0 1 1],'Color','w');
    exportgraphics(export,strcat('./0-figures/m_uPlus_yPlus_',
       sprintf('%04d',jj),'.png'),'Resolution',600)
    im = imread(strcat('./0-figures/m_uPlus_yPlus_',sprintf('%04d
       ',jj),'.png'));
    im2 = padarray(im, [100 100], 255);
```

```
        imwrite(im2,strcat('./0-figures/m_uPlus_yPlus_',sprintf('%04d
            ',jj),'.png'));

%       delete(a)

end
```

matlab Folder: Matlab Files applicable to each simulation

Code Listing C.42: plot_yPlus.m, to graphically display the raw yPlus results.

```
%% Global Definitions
if ~exist("currentDirectory","var"); clear;clc;clf; end

if exist("global_values.m",'file') == 2 %If this folder contains
   a global_values.m file, then run that file
    global_values
else %Otherwise, define relevant parameters
    xmin = 0.1;
    xmax = 0.4;
end

if contains(pwd,"flowDev")
    patch = 'ground_surface';
else
    % Change patch name to graph here, if file has the
       rectangular cylinder
    patch = 'case2_bottom'; % ground_surface   case2_inlet
       case2_top   case2_outlet   case2_bottom
end
patchNames = ["case2_bottom"   "case2_inlet"   "case2_outlet"   "
   case2_top"   "ground_surface"];
columnTypes = ["min" "max" "average"]; %For selecting the correct
    column in order from data
columnOfInterest = "average";

% Process Data
%Load forceCoefficients data if not already loaded
if exist('d_yPlus','var') == 1
    %Do nothing
else
    %Var does not exist. Import file
    clear r_yPlus
    importData_yPlus
    fprintf('Original yPlus dataset\nMin time: %f, Max time: %f\n
       \n',d_yPlus.data.time(1,1),d_yPlus.data.time(end,1))
    d_yPlus.data = transientDataCrop_yPlus(d_yPlus,xmin,xmax);
    fprintf('Trimmed yPlus dataset\nMin time: %f, Max time: %f\n\
       n',d_yPlus.data.time(1,1),d_yPlus.data.time(end,1))
    clearUnusedVars
end

r_yPlus.excelPoints = [];
```

```
if a == 1 %Check identifier from importData_yPlus file checking
    case2*
    r_yPlus.case2_bottom.mean    = mean(d_yPlus.data.case2_bottom
        (:,:),1);
    r_yPlus.case2_inlet.mean     = mean(d_yPlus.data.case2_inlet
        (:,:),1);
    r_yPlus.case2_outlet.mean    = mean(d_yPlus.data.case2_outlet
        (:,:),1);
    r_yPlus.case2_top.mean       = mean(d_yPlus.data.case2_top
        (:,:),1);

    r_yPlus.case2_bottom.max_100Prtile     = max(d_yPlus.data.
        case2_bottom(:,2));
    r_yPlus.case2_inlet.max_100Prtile      = max(d_yPlus.data.
        case2_inlet(:,2));
    r_yPlus.case2_outlet.max_100Prtile     = max(d_yPlus.data.
        case2_outlet(:,2));
    r_yPlus.case2_top.max_100Prtile        = max(d_yPlus.data.
        case2_top(:,2));

    r_yPlus.case2_bottom.min_000Prtile  = min(d_yPlus.data.
        case2_bottom(:,1));
    r_yPlus.case2_inlet.min_000Prtile   = min(d_yPlus.data.
        case2_inlet(:,1));
    r_yPlus.case2_outlet.min_000Prtile  = min(d_yPlus.data.
        case2_outlet(:,1));
    r_yPlus.case2_top.min_000Prtile     = min(d_yPlus.data.
        case2_top(:,1));

    r_yPlus.case2_bottom.max_005Prtile  = prctile(d_yPlus.data.
        case2_bottom(:,2),5);
    r_yPlus.case2_inlet.max_005Prtile   = prctile(d_yPlus.data.
        case2_inlet(:,2),5);
    r_yPlus.case2_outlet.max_005Prtile  = prctile(d_yPlus.data.
        case2_outlet(:,2),5);
    r_yPlus.case2_top.max_005Prtile     = prctile(d_yPlus.data.
        case2_top(:,2),5);

    r_yPlus.case2_bottom.max_050Prtile  = prctile(d_yPlus.data.
        case2_bottom(:,2),50);
    r_yPlus.case2_inlet.max_050Prtile   = prctile(d_yPlus.data.
        case2_inlet(:,2),50);
    r_yPlus.case2_outlet.max_050Prtile  = prctile(d_yPlus.data.
        case2_outlet(:,2),50);
    r_yPlus.case2_top.max_050Prtile     = prctile(d_yPlus.data.
        case2_top(:,2),50);

    r_yPlus.case2_bottom.max_095Prtile  = prctile(d_yPlus.data.
        case2_bottom(:,2),95);
```

```
    r_yPlus.case2_inlet.max_095Prtile    = prctile(d_yPlus.data.
       case2_inlet(:,2),95);
    r_yPlus.case2_outlet.max_095Prtile   = prctile(d_yPlus.data.
       case2_outlet(:,2),95);
    r_yPlus.case2_top.max_095Prtile      = prctile(d_yPlus.data.
       case2_top(:,2),95);

    r_yPlus.case2_bottom.min_005Prtile   = prctile(d_yPlus.data.
       case2_bottom(:,1),5);
    r_yPlus.case2_inlet.min_005Prtile    = prctile(d_yPlus.data.
       case2_inlet(:,1),5);
    r_yPlus.case2_outlet.min_005Prtile   = prctile(d_yPlus.data.
       case2_outlet(:,1),5);
    r_yPlus.case2_top.min_005Prtile      = prctile(d_yPlus.data.
       case2_top(:,1),5);

    r_yPlus.case2_bottom.min_050Prtile   = prctile(d_yPlus.data.
       case2_bottom(:,1),50);
    r_yPlus.case2_inlet.min_050Prtile    = prctile(d_yPlus.data.
       case2_inlet(:,1),50);
    r_yPlus.case2_outlet.min_050Prtile   = prctile(d_yPlus.data.
       case2_outlet(:,1),50);
    r_yPlus.case2_top.min_050Prtile      = prctile(d_yPlus.data.
       case2_top(:,1),50);

    r_yPlus.case2_bottom.min_095Prtile   = prctile(d_yPlus.data.
       case2_bottom(:,1),95);
    r_yPlus.case2_inlet.min_095Prtile    = prctile(d_yPlus.data.
       case2_inlet(:,1),95);
    r_yPlus.case2_outlet.min_095Prtile   = prctile(d_yPlus.data.
       case2_outlet(:,1),95);
    r_yPlus.case2_top.min_095Prtile      = prctile(d_yPlus.data.
       case2_top(:,1),95);

    % Saving results into a tbl for easy copy/paste to excel
       datastructure
    % For excel max max, using max_095
    % For excel min min, using min_005
    % Excel tbl order: max max, min min, avg min, avg max, avg
    for i_excel=1:4
        r_yPlus.excelPoints = horzcat( ...
            r_yPlus.excelPoints, ...
            r_yPlus.(char(patchNames(1,i_excel))).max_095Prtile,
               ...
            r_yPlus.(char(patchNames(1,i_excel))).min_005Prtile,
               ...
            r_yPlus.(char(patchNames(1,i_excel))).mean);
    end
end
```

```
r_yPlus.ground_surface.mean = mean(d_yPlus.data.ground_surface
    (:,:),1);
r_yPlus.ground_surface.max_005Prtile = prctile(d_yPlus.data.
    ground_surface(:,2),5);
r_yPlus.ground_surface.max_050Prtile = prctile(d_yPlus.data.
    ground_surface(:,2),50);
r_yPlus.ground_surface.max_095Prtile = prctile(d_yPlus.data.
    ground_surface(:,2),95);
r_yPlus.ground_surface.max_100Prtile = max(d_yPlus.data.
    ground_surface(:,2));

r_yPlus.ground_surface.min_000Prtile = min(d_yPlus.data.
    ground_surface(:,1));
r_yPlus.ground_surface.min_005Prtile = prctile(d_yPlus.data.
    ground_surface(:,1),5);
r_yPlus.ground_surface.min_050Prtile = prctile(d_yPlus.data.
    ground_surface(:,1),50);
r_yPlus.ground_surface.min_095Prtile = prctile(d_yPlus.data.
    ground_surface(:,1),95);

r_yPlus.excelPoints = horzcat(r_yPlus.excelPoints,r_yPlus.(char(
    patchNames(1,5))).max_095Prtile,r_yPlus.(char(patchNames(1,5))
    ).min_005Prtile,r_yPlus.(char(patchNames(1,5))).mean);

clear i_excel

if ~exist("currentDirectory","var") || simFigures
    % Print figures because this script is NOT being run from a
        parent file
    fprintf('              mean of >    %s    %s    %s\n',
        columnTypes)
    printstruct(r_yPlus)

    % Plot Data
    column = strmatch(columnOfInterest,columnTypes,"exact");

    %Figure 1 - plot average, min OR max of one patch
    figure(1)
    plot(d_yPlus.data.time,d_yPlus.data.(patch)(:,column))
    hold on
    plot([xmin xmax],[r_yPlus.(patch).mean(1,column) r_yPlus.(
        patch).mean(1,column)])
    hold off
    % ylim([4e3 1.8e4])
    xlim([xmin xmax])
    legend('yPlus actual','yPlus mean')
    xlabel('time (sec)')
    ylabel('yPlus')
    title(strcat(patch," ",columnOfInterest," yPlus values"),'
        Interpreter','none')
```

```
    %Figure 2 - plot average, min and max of one patch on the
       same figure
    figure(2)
    %Plot average
    subplot(2,2,1:2)
    plot(d_yPlus.data.time,d_yPlus.data.(patch)(:,3))
    hold on
    plot([xmin xmax],[r_yPlus.(patch).mean(1,3) r_yPlus.(patch).
       mean(1,3)])
    hold off
    xlim([xmin xmax])
    legend('yPlus actual','yPlus mean')
    xlabel('time (sec)')
    ylabel('yPlus')
    title('Average')

    %
    %Plot min
    subplot(2,2,3)
    plot(d_yPlus.data.time,d_yPlus.data.(patch)(:,1))
    hold on
    plot([xmin xmax],[r_yPlus.(patch).mean(1,1) r_yPlus.(patch).
       mean(1,1)])
    hold off
    xlim([xmin xmax])
    legend('yPlus actual','yPlus mean')
    xlabel('time (sec)')
    ylabel('yPlus')
    title('Min')

    %Plot max
    subplot(2,2,4)
    plot(d_yPlus.data.time,d_yPlus.data.(patch)(:,2))
    hold on
    plot([xmin xmax],[r_yPlus.(patch).mean(1,2) r_yPlus.(patch).
       mean(1,2)])
    hold off
    xlim([xmin xmax])
    legend('yPlus actual','yPlus mean')
    xlabel('time (sec)')
    ylabel('yPlus')
    title('Max')

    sgtitle(strcat("yPlus values on ",patch),'Interpreter','none
       ')
end
```

C.5.6 Supporting the Developed Flow Profile

The scripts provided here are intended to find the results of the developed profile simulation, save the appropriate time step, and trim the data so that the processes in the developMap folder can be completed.

Code Listing C.43: allImport100mResults.sh, to copy the results from the completed developed flow profile simulation.

```
#!/bin/bash

#PURPOSE: To copy data from /storage to here for local/easy
   manipulation
# $1 is the speed of interest from global/developedFlow/100
   m_results/
# Folder name is given as velocity_time in the form of W#_###-##
   sec

echo "--- RUNNING allImport100mResults.sh ---"

echo "> mkdir /original and /trimmed"
mkdir $1
mkdir $1/original/
mkdir $1/trimmed/

vel=$( echo $1 | sed 's/\(.*\)_.*/\1 /' | tr -d '\n' )
                #Parse out of the folder name the velocity
#vel=$( echo $1 | sed 's/_.*sec//' )              #Parse out of
   the folder name the velocity
#vel=$( echo $1 | head -c 2)              #Parse out of the
   folder name the velocity
time=$(echo $1 | sed 's/^W.*_//' | sed 's/sec//' | sed 's/-/./' |
    sed '/\./ s/\.\{0,1\}0\{1,\}$//' | sed 's/^0*//') #Parse out
   of the folder name the time
#time=$(echo $1 | head -c 8 | tail -c 5 | sed 's/-/./' | sed
   '/\./ s/\.\{0,1\}0\{1,\}$//' | sed 's/^0*//') #Parse out of
   the folder name the time

echo Vel = $vel
echo Time = $time

echo "> Copying data from /storage to here"
#Copy 2022-1223*/analysis/#-##/ values to /original
rsync -a /storage/dksan/2023-0316a*_$(echo $vel)*/analysis/$time/
    $1/original/

echo "> Copying data from /storage to here ..."
```

```
#Copy 0/C to /original
find /storage/dksan/2023-0316a*_$(echo $vel)*/analysis/* -name "C
    " | xargs cp -t $1/original/

echo "--- allImport100mResults.sh FINISHED ---"
```

Supporting the Developed Flow Profile

Code Listing C.44: allTrim.sh, to trim the copied data so only the necessary data is retained for the procedures in developMap.

```
#!/bin/bash

#PURPOSE: To trim data in /original
# $1 is the speed of interest from global/developedFlow/100
   m_results/
echo "--- RUNNING allTrim.sh ---"

varExtract=("C" "k" "nut" "omega" "U")
varDelete=("p" "phi" "uniform" "wallShearStress" "yPlus")

echo "> Extracting relevant patches from raw data -- wHeader*"
echo "> Removing top/bottom of datasets -- trim*"

vel=$( echo $1 | sed 's/\(.*\)_.*/\1/' | tr -d '\n')
                #Parse out of the folder name the velocity
#vel=$( echo $1 | sed 's/_.*sec//' )              #Parse out of
   the folder name the velocity
#vel=$(echo $1 | head -c 2)

flowDevCellNum=1186 #1195

for value in ${varExtract[@]}; do
    echo "   * Processing $value"
    sed -n '/cross_outlet/,/ground_surface/p' $1/original/$value
       > $1/trimmed/wHeader_$value
    #lineNumber to remove from wHeader, from the top until the
       first line that says 995
    lineNum=$(grep -m 1 -n $flowDevCellNum $1/trimmed/
       wHeader_$value)
    lineNum=${lineNum%:*}
    sed "1,$((lineNum-1))d" $1/trimmed/wHeader_$value | sed "1,2d
       " | sed "$(($flowDevCellNum+1)),\$d" | sed 's/[()]//g' >
       $1/trimmed/trimmed_dF_"$vel"_$value
done

for value2 in ${varDelete[@]}; do
    rm -rf $1/original/$value2
done
echo "$1/original cleaned out of unused files"

echo "--- allTrim.sh FINISHED ---"
```

C.5.7 Global Simulation Scripts - In Storage Folder

These scripts are contained within the final data storage folder (for ex: \storage\username\). The purpose of these scripts is prepare the processing of the collective data set, to calculate results and plot them. This requires making sure the local Matlab folders are up to date, that the global_values.m files are properly configured, importing results, and plotting the data.

Code Listing C.45: allCheckMatlabPrep_GlobalValues_Matlab.sh, to check that the local Matlab folders have been updated to match the global Matlab folder for simulations that meet the required criteria (ls | grep).

```
#!/bin/bash

# $1 is the folder name(s) that should be updated. It can be a
   partial name such that all completions of that name will be
   updated
# $2 is an optional check to suppress/alter matlab folder check

# Run this script from the original working directory.

echo "--- RUNNING allCheckMatlabPrep_GlobalValues_Matlab.sh ---"
echo

#simMatches=$(ls | grep $1)
mapfile -t simMatches < <( ls | grep $1 )
echo Checking values for:

for i in "${simMatches[@]}"; do
    echo "$i:   $(cat /storage/dksan/$i/global_values.m | grep x
       | tr -d '\n')  $(cat /storage/dksan/$i/analysis/system/
       controlDict | grep 'endTime '| sed 's/;.*\/\/.*/;/')"
    colordiff -r --exclude=*~  /storage/dksan/$i/matlab/ /home/
       dksan/OpenFOAM/myWork/global-base-case/matlab/202*/
done

echo
echo "--- allCheckMatlabPrep_GlobalValues_Matlab.sh FINISHED ---"
```

Global Simulation Scripts - In Storage Folder

Code Listing C.46: allGlobal_ValuesUpdate.sh, to update the global_values.m file for simulations that meet the required criteria (ls | grep).

```
#!/bin/bash

# $1 is the folder name(s) that should be updated. It can be a
   partial name such that all completions of that name will be
   updated
# $2 is the xmin value that should be used
# $3 is the xmax value that should be used

# Run this script from the original working directory.

echo "--- RUNNING allGlobal_ValuesUpdate.sh ---"
echo

#simMatches=$(ls | grep $1)
mapfile -t simMatches < <( ls | grep $1 )
echo Updating global values for:

for i in "${simMatches[@]}"; do
    echo "$i"
    sed -i "s/xmin=.*;/xmin=$2;/" /storage/dksan/$i/global_values
       .m
    sed -i "s/xmax=.*;/xmax=$3;/" /storage/dksan/$i/global_values
       .m

    #cat /storage/dksan/$i/global_values.m
    #sleep 1
done

echo
echo "--- allGlobal_ValuesUpdate.sh FINISHED ---"
```

Global Simulation Scripts - In Storage Folder

Code Listing C.47: allMatlabVariablesLegal.sh, to update folder names to make them "Matlab" legal, by removing dashes and replacing with underscores.

```
#!/bin/bash

# $1 is the folder name(s) that should be updated. It can be a
   partial name such that all completions of that name will be
   updated

# Run this script from the original working directory.

echo "--- RUNNING allMatlabVariablesLegal.sh ---"
echo

mapfile -t simMatches < <( ls | grep $1 | grep 0- )
echo Updating folder names for:

for i in "${simMatches[@]}"; do
    echo "$i"
        mv                              $i                                  $(
           echo "$i" | sed "s/0-/0_/g")
    sudo mv /nas/dksan/storage_bckp/$i /nas/dksan/storage_bckp/$(
       echo "$i" | sed "s/0-/0_/g")
    #sleep 5
done

echo
echo Revised names:
ls | grep $1

echo
echo "--- allMatlabVariablesLegal.sh FINISHED ---"
```

Global Simulation Scripts - In Storage Folder

Code Listing C.48: allUpdateMatlabFiles.sh, update the local Matlab folder for simulations that meet the required criteria (ls | grep). This is required if allCheckMatlabPrep_GlobalValues_Matlab.sh shows that the local Matlab folder is outdated.

```
#!/bin/bash

# $1 is the folder name(s) that should be updated. It can be a
   partial name such that all completions of that name will be
   updated
# Run this script from the original working directory.

echo "--- RUNNING allUpdateMatlabFiles.sh ---"
echo

#simMatches=$(ls | grep $1)
mapfile -t simMatches < <( ls | grep $1 )
sudo echo Updating matlab folder for:

for i in "${simMatches[@]}"; do
    echo "$i"

    sudo rm -rf                                          /storage/dksan/$i
       /matlab
    sudo rm -f                                           /storage/dksan/$i
       /startup.m
    sudo install -d -m 0755 -o dksan -g dksan    /storage/dksan/$i
       /matlab
    sudo rsync -a    /home/dksan/OpenFOAM/myWork/global-base-case/
       matlab/202*/*   /storage/dksan/$i/matlab/
    sudo rsync -a    /home/dksan/OpenFOAM/myWork/global-base-case/
       startup.m       /storage/dksan/$i/
#    sleep 10
done

echo
echo "--- allUpdateMatlabFiles.sh FINISHED ---"
```

Global Simulation Scripts - In Storage Folder

Code Listing C.49: func_calcJFMCoefficients.m, Matlab function to calculate the aerodynamic coefficients from the OpenFOAM force data.

```
function [paraRes] = func_calcJFMCoefficients(paraSet,
    data_coeffs_unconfined)
% Calculate coefficients. Using CIVE 300 definition, where "drag
    is the net
% force in the direction of flow due to the pressure AND shear
    forces on
% the surface of the object."
for paraRes_i=1:length(paraSet.flow(:,1))

    % C_side. IS THIS SUPPOSED TO BE JUST THE FORCE ON THE INLET
        SIDE? WHAT
    % ABOUT VISCOUS FORCES? 7/24/23. Per CIVE 300 book, yes,
        include viscous
    paraRes.C_side(paraRes_i,1) ...
        = mean(...
            data_coeffs_unconfined.(char(paraSet.vel(paraRes_i,2)
                )).d_forceIncomp_inlet_c2.data.F_p_x + ...
            data_coeffs_unconfined.(char(paraSet.vel(paraRes_i,2)
                )).d_forceIncomp_inlet_c2.data.F_visc_x) / ...
        (0.5 * paraSet.rho_water * paraSet.characteristicLength *
            1 * paraSet.flow(paraRes_i,1)^2);

    % C_lift. GOOD. THIS INCLUDES ALL FORCES IN THE Y DIRECTION
        AND MATCHES
    % OF'S RESULTS
    paraRes.C_lift(paraRes_i,1) ...
        = mean(...
            data_coeffs_unconfined.(char(paraSet.vel(paraRes_i,2)
                )).d_forceIncomp_case2.data.F_p_y + ...
            data_coeffs_unconfined.(char(paraSet.vel(paraRes_i,2)
                )).d_forceIncomp_case2.data.F_visc_y) / ...
        (0.5 * paraSet.rho_water * paraSet.characteristicLength *
            1 * paraSet.flow(paraRes_i,1)^2);

    % C_rollover. Per Baker/Soper 2022. CoR is leeward wheels.
        This is
    % calculated based on the total forces, with an assumed
        center of force
    % at the centroid of the rectangular cylinder.
    paraRes.RollingMoment.(strcat('paraRes_i',sprintf('%d',
        paraRes_i))) ...
        = ((data_coeffs_unconfined.(char(paraSet.vel(paraRes_i,2)
            )).d_forceIncomp_case2.data.F_p_x +
```

```
            data_coeffs_unconfined.(char(paraSet.vel(paraRes_i,2))
            ).d_forceIncomp_case2.data.F_visc_x) ...
                  * (0.148507 + (0.364922/2)))...
        + ((data_coeffs_unconfined.(char(paraSet.vel(paraRes_i,2)
            )).d_forceIncomp_case2.data.F_p_y +
            data_coeffs_unconfined.(char(paraSet.vel(paraRes_i,2))
            ).d_forceIncomp_case2.data.F_visc_y) ...
                  * (0.3239/2));
    paraRes.C_rollover(paraRes_i,1) ...
        = mean(paraRes.RollingMoment.(strcat('paraRes_i',sprintf
            ('%d',paraRes_i))))/ ...
        (0.5 * paraSet.rho_water * paraSet.characteristicLength^2
            * 1 * paraSet.flow(paraRes_i,1)^2);

    % C_drag. GOOD. THIS IS ALL FORCES, PRESSURE AND VISCOUS IN
        THE X
    % DIRECTION. THIS MATCHES OF'S CALC OF CD. DOES IT MAKE SENSE
        TO
    % INCLUDE ALL FORCES IN THE X DIRECTION? 7/24/23. It makes
        sense to me
    paraRes.C_drag(paraRes_i,1) ...
        = mean(...
            data_coeffs_unconfined.(char(paraSet.vel(paraRes_i,2)
                )).d_forceIncomp_case2.data.F_p_x + ...
            data_coeffs_unconfined.(char(paraSet.vel(paraRes_i,2)
                )).d_forceIncomp_case2.data.F_visc_x) / ...
        (0.5 * paraSet.rho_water * paraSet.characteristicLength *
            1 * paraSet.flow(paraRes_i,1)^2);

    % St Number
    paraRes.Strouhal(paraRes_i,1) ...
        = (data_coeffs_unconfined.(char(paraSet.vel(paraRes_i,2))
            ).maxFrequency_AtMaxYValue * paraSet.
            characteristicLength) / paraSet.flow(paraRes_i,1);
end

clear paraRes_i* para_i
end
```

Global Simulation Scripts - In Storage Folder

Code Listing C.50: func_importSims.m, function to import the simulations that meet the required pattern. This supports the remaining Matlab scripts which analyze the simulations in a conglomerated data set.

```
function [data] = func_importSims(simName)
% This function goes through the tedious process of importing the
% simulation folders that meet the name completion of "simName"

% simName should be a partial completion of a folder name in this
    directory
% (eg '2023-032' will import all folders for '2023-0321*' and
   '2023-0322*')

% The output of this function is a single structure that contains
    a
% sub-structure for each folder of data

% Get folders that meet the partial name given
z_folders = dir(simName);
z_numOfFolders = length(z_folders);
global simFigures

% Loop through all folders
for z_i_numFolders=1:z_numOfFolders
    % Get the full directory name and cd to that directory
    dirName = strcat(z_folders(z_i_numFolders).folder,'/',
       z_folders(z_i_numFolders).name);
    cd(dirName);

    fprintf('\n - - - - - - - - - - - - - - - - - - - - - - - - -
       - - - - - - - - - - - - - - - - - - - ')
    [~,currentDirectory,~]=fileparts(pwd)
    % Load local startup files and run matlab script
    startup
    if ~contains(currentDirectory,"flowDev")
        %plot_forceCoeff;
        fft_analysis;
    end
    plot_yPlus;
    %dbstop if simFigures

    clear dirName

    % Create a list of the local variables to be saved. These
       local
    % variables will be placed into a single temporary structure
       -
    % struct_myData
```

```
    struct_varGenerate=[who,who].';
    struct_myData=struct(struct_varGenerate{:});
    struct_variableRedef=char(structvars(struct_myData,0));
    struct_numOfVars=length(struct_variableRedef(:,1)) ;
    struct_eval=''; % Create a string to be evaluated to transfer
        existing vars into a structure
    for i_struct=1:struct_numOfVars
        struct_stringComp(i_struct,:) = struct_variableRedef(
            i_struct,:);
        struct_eval = strcat(struct_eval,struct_stringComp(
            i_struct,:));
    end
    eval(struct_eval) % Transfer variables to structure

    % Delete previously saved data 'a_' (if applicable). Purpose:
        so that
    % previously created structures are not appended as never
       ending
    % substructures
    struct_names = fieldnames( struct_myData );
    struct_oldData = struct_names( find( ~cellfun( @isempty,
       strfind( struct_names , 'a_' ) ) ) );
    hasField = isfield(struct_myData, struct_oldData); % Will be
       True or False.
    % Now remove it if it's there.
    if hasField
        % Field is there.  Remove it.
        struct_myData = rmfield( struct_myData, struct_oldData);
    else
        % Field is not there, warn user.
        fprintf('No old data to remove. Proceeding to next
            dataset')
        %warningMessage = sprintf('Warning: the structure "
            handles"\ndoes not have a field called "Data."');
        %uiwait(warndlg(warningMessage));
    end

    feval(@()assignin('caller',strcat('a_',char(extractBetween(
       currentDirectory,"2023-","Flow_"))),struct_myData));

    clear struct_* i_struct hasField warningMessage
       currentDirectory
    clear d_yPlus
    cd ../
end

clear d_* C_* coefficient
clear rho_water
clear RollingMoment MomentArm
clear xmin xmax
```

```
% Save the datastructures a_ to a parent structure 'data'.
   Follows same
% logic as method above
struct_who = [,who].';
struct_varGenerate      = struct_who(~cellfun('isempty',regexp(
   struct_who,'^a_')));
struct_varGenerate(2,:) = struct_who(~cellfun('isempty',regexp(
   struct_who,'^a_')));

data=struct(struct_varGenerate{:});
struct_variableRedef=char(structvars(data,0));
struct_numOfVars=length(struct_variableRedef(:,1)) ;
struct_eval=''; % Create a string to be evaluated to transfer
   existing vars into a structure
for i_struct=1:struct_numOfVars
    struct_stringComp(i_struct,:) = struct_variableRedef(i_struct
       ,:);
    struct_eval = strcat(struct_eval,struct_stringComp(i_struct
       ,:));
end
eval(struct_eval) % Transfer variables to structure
clear struct* a_* i_struct

end
```

Global Simulation Scripts - In Storage Folder

Code Listing C.51: plot_all_yPlus.m, a Matlab file to plot yPlus for all simulations. The results are in a tabular format which can be copy/pasted.

```
%Plot data

resampleData = true;
if resampleData

    clear;clc;clf
    close all
    cd /storage/dksan/
    global simFigures
    simFigures = false; %% Define as true/false if I would like
       to see figures in the following sims
    data_Validation = func_importSims('2023-0606c*');
    %data_H1     = func_importSims('2023-0305*');
    %data_H2     = func_importSims('2023-0306*');
    %mergestructs = @(data_H1,data_H2) cell2struct([struct2cell(
       data_H1);struct2cell(data_H2)],[fieldnames(data_H1);
       fieldnames(data_H2)]);
    %data_H = mergestructs(data_H1, data_H2);

    clear data_H1 data_H2 mergestructs

    %save("processedData_VnV_Validation","data_Validation");
else
    fprintf('\nData has not been resampled\n')
    if exist('data_Validation','var')
        fprintf('\nData has NOT been imported\n')
    else
        fprintf('\nData HAS been imported\n')
        %load("processedData_VnV_Validation","data_Validation");
    end
end

clear z_* paraSet paraRes* ax* resampleData


% CALCULATIONS
clear pmetric*
paraSet.flow = zeros(2,2);
paraSet.vel(1,1) = {'W0'};
%paraSet.vel(1,1) = {'W1'};
%paraSet.vel(3,1) = {'W2'};
%paraSet.vel(4,1) = {'W3'};
%paraSet.vel(5,1) = {'W4'};
%paraSet.vel(6,1) = {'W5'};
```

```
%paraSet.vel(7,1) = {'W6'};
%paraSet.vel(2,1) = {'W7'};

paraSet.vel(1,2) = {'a_0522_case2_W0'};
%paraSet.vel(1,2) = {'a_0417ba_case2_W1'};
%paraSet.vel(3,2) = {'a_0417bb_case2_W2'};
%paraSet.vel(4,2) = {'a_0417bc_case2_W3'};
%paraSet.vel(5,2) = {'a_0417bd_case2_W4'};
%paraSet.vel(6,2) = {'a_0417be_case2_W5'};
%paraSet.vel(7,2) = {'a_0417bf_case2_W6'};
%paraSet.vel(2,2) = {'a_0417bg_case2_W7'};

paraSet.flow(1,1) = 0.275127287475132;
%paraSet.flow(1,1) = 5.33333333333333;
%paraSet.flow(3,1) = 9.6;
%paraSet.flow(4,1) = 13.8666666666667;
%paraSet.flow(5,1) = 18.1333333333333;
%paraSet.flow(6,1) = 22.4;
%paraSet.flow(7,1) = 26.6666666666667;
%paraSet.flow(2,1) = 30.9333333333333;

paraSet.characteristicLength = 0.364922;
paraSet.nu_water = 0.000001004;
paraSet.rho_water = 998.2;

paraSet.numberOfSims = numel(fieldnames(data_Validation));
%paraSet.USlength = linspace(2,20,19);
%paraSet.DSlength = linspace(2,15,14);
%paraSet.H_W1length= linspace(2,16,8);
%paraSet.H_W7length= linspace(2,20,10);
paraSet.dataNames_Validation = fieldnames(data_Validation);
%paraSet.dataNames_H     = fieldnames(data_H);

% Calculate Reynolds Number
%for para_i=1:length(paraSet.flow(:,1))
%    paraSet.flow(para_i,2) = (paraSet.flow(para_i,1) * paraSet.
   characteristicLength)/paraSet.nu_water;
%end

% Merge data into a single dataset.
%for paraRes_i2=1:(length(paraSet.layers_case2)); paraRes.
   Cd_Layers_W0(paraRes_i2,1)      = data_Validation.(char(paraSet
   .dataNames_Validation(paraRes_i2,1))).d_mean.Cd; end
for paraRes_i3=1:(paraSet.numberOfSims); paraRes.yPlus_excelPts(
   paraRes_i3,:)    = data_Validation.(char(paraSet.
   dataNames_Validation(paraRes_i3,1))).r_yPlus.excelPoints; end
% for paraRes_i32=1:(length(paraSet.layers_case2)); paraRes.
   c2bottom_max_095percentile(paraRes_i32,1)    = data_Validation
   .(    char(paraSet.dataNames_Validation(       paraRes_i32,1))).
   r_yPlus.case2_bottom.max_095Prtile; end
```

```
% for paraRes_i33=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_max_095percentile(paraRes_i33,1)    = data_Validation
    .(    char(paraSet.dataNames_Validation(       paraRes_i33,1))).
    r_yPlus.case2_bottom.max_095Prtile; end
% for paraRes_i34=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_max_095percentile(paraRes_i34,1)    = data_Validation
    .(    char(paraSet.dataNames_Validation(       paraRes_i34,1))).
    r_yPlus.case2_bottom.max_095Prtile; end
% for paraRes_i35=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_max_095percentile(paraRes_i35,1)    = data_Validation
    .(    char(paraSet.dataNames_Validation(       paraRes_i35,1))).
    r_yPlus.case2_bottom.max_095Prtile; end
% for paraRes_i4=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_min_005percentile(paraRes_i4,1)    = data_Validation.(
       char(paraSet.dataNames_Validation(       paraRes_i4,1))).
    r_yPlus.case2_bottom.min_005Prtile; end
% for paraRes_i5=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_mean(paraRes_i5,1)                  = data_Validation.(
       char(paraSet.dataNames_Validation(       paraRes_i5,1))).
    r_yPlus.case2_bottom.mean(1,3); end
%for paraRes_i0=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_min_005percentile(paraRes_i0,1)    = data_Validation.(
       char(paraSet.dataNames_Validation(       paraRes_i0,1))).
    r_yPlus.case2_bottom.min_005Prtile; end
clear paraRes_i* para_i

% Calculate % Error based on prescribed value
% paraRes.H_W1(:,2)  = (paraRes.H_W1(:,1) /paraRes.H_W1(6,1)  -
    1) * 100;
% paraRes.H_W7(:,2)  = (paraRes.H_W7(:,1) /paraRes.H_W7(6,1)  -
    1) * 100;
% paraRes.US_W1(:,2) = (paraRes.US_W1(:,1)/paraRes.US_W1(9,1) -
    1) * 100;
% paraRes.US_W7(:,2) = (paraRes.US_W7(:,1)/paraRes.US_W7(9,1) -
    1) * 100;
% paraRes.DS_W1(:,2) = (paraRes.DS_W1(:,1)/paraRes.DS_W1(6,1) -
    1) * 100;
% paraRes.DS_W7(:,2) = (paraRes.DS_W7(:,1)/paraRes.DS_W7(9,1) -
    1) * 100;

clf

% Cd_interp = 2.2562065318068;
% for i=1:(length(paraRes.Cd_Layers_W0(:,1)))
%     paraRes.errorVal(i,1) = paraRes.Cd_Layers_W0(i,1) -
    Cd_interp;
%     paraRes.errorPer(i,1) = (100*paraRes.Cd_Layers_W0(i,1)/
    Cd_interp) - 100;
```

```
%     fprintf('Layer Count = %d. Cd = %.4f; Error = %.4f %%\n',
    paraSet.layers_case2(1,i), paraRes.Cd_Layers_W0(i,1), paraRes.
    errorPer(i,1))
% end
% fprintf('\nMax Error %.4f %%\n',max(abs(paraRes.errorPer)))
% fprintf(    'Min Error %.4f %%\n',min(abs(paraRes.errorPer)))
%
% vnv_tbl = horzcat(transpose(paraSet.layers_case2),paraRes.
    Cd_Layers_W0,paraRes.errorPer)
% vnv_tbl2 = vnv_tbl
% vnv_tbl2(:,2) = round(vnv_tbl(:,2),5);
% vnv_tbl2(:,3) = round(vnv_tbl(:,3),5);

% figure(1)
% %subplot(2,1,1)
% ax11.line1 = plot(paraSet.layers_case2,paraRes.Cd_Layers_W0
    (:,1),'b*-');
% hold on
% ax11.line2 = plot([0,8],[2.256206532, 2.256206532],'s-','Color
    ',[1 0.5 0]);
% grid on
% %grid minor
% ylim([0 2.5]);
% xlim([0 8]);
% xlabel('Number of Inflation Layers')
% ylabel('C_{drag}')
% title('Validation of Unconfined Rectangular Cylinder')
% hold off
% %
% subplot(2,1,2)
% ax12.line1 = plot(paraSet.H_W1length,paraRes.H_W1(:,2),'b*-');
% hold on
% ax12.line2 = plot([2.256206532, 2.256206532],paraRes.Layers_W0
    (:,1),'s-','Color',[1 0.5 0]);
% grid on
% grid minor
% ylim([-20    ceil(max(max(paraRes.H_W7(:,2),paraRes.H_W7(:,2)))
    /10)*10]);
% xlim([0 max(paraSet.H_W7length)+2]);
% xlabel('Domain Height (m)')
% ylabel('% Relative Error')
% %title('Upstream Verification')
% legend('W1','W7')
% hold off
%
% figure(2)
% subplot(2,1,1)
% ax21.line1 = plot(paraSet.USlength,paraRes.US_W1(:,1),'b*-');
% hold on
```

```
% ax21.line2 = plot(paraSet.USlength,paraRes.US_W7(:,1),'s-','
    Color',[1 0.5 0]);
% grid on
% grid minor
% ylim([0 ceil(max(max(paraRes.US_W1(:,1),paraRes.US_W7(:,1))))])
    ;
% xlim([0 max(paraSet.USlength)+2]);
% %xlabel('Length Upstream of Cylinder Centroid (m)')
% ylabel('C_{drag}')
% title('Upstream Verification')
% hold off
%
% subplot(2,1,2)
% ax22.line1 = plot(paraSet.USlength,paraRes.US_W1(:,2),'b*-');
% hold on
% ax22.line2 = plot(paraSet.USlength,paraRes.US_W7(:,2),'s-','
    Color',[1 0.5 0]);
% grid on
% grid minor
% ylim([-20    ceil(max(max(paraRes.US_W1(:,2),paraRes.US_W7(:,2))
    )/10)*10]);
% xlim([0 max(paraSet.USlength)+2]);
% xlabel('Length Upstream of Cylinder Centroid (m)')
% ylabel('% Relative Error')
% %title('Upstream Verification')
% legend('W1','W7')
% hold off
%
% figure(3)
% subplot(2,1,1)
% ax3.line1 = plot(paraSet.DSlength,paraRes.DS_W1(:,1),'b*-');
% hold on
% ax3.line2 = plot(paraSet.DSlength,paraRes.DS_W7(:,1),'s-','
    Color',[1 0.5 0]);
% grid on
% grid minor
% ylim([0 ceil(max(max(paraRes.DS_W1(:,1),paraRes.DS_W7(:,1))))])
    ;
% xlim([0 max(paraSet.DSlength)+1]);
% %xlabel('Length Downstream of Cylinder Centroid (m)')
% ylabel('C_{drag}')
% title('Downstream Verification')
% hold off
%
% subplot(2,1,2)
% ax3.line1 = plot(paraSet.DSlength,paraRes.DS_W1(:,2),'b*-');
% hold on
% ax3.line2 = plot(paraSet.DSlength,paraRes.DS_W7(:,2),'s-','
    Color',[1 0.5 0]);
% grid on
```

```
% grid minor
% ylim([-30    ceil(max(max(paraRes.DS_W1(:,2),paraRes.DS_W7(:,2))
    )/10)*10]);
% xlim([0 max(paraSet.DSlength)+1]);
% xlabel('Length Downstream of Cylinder Centroid (m)')
% ylabel('% Relative Error')
% %title('Downstream Verification')
% legend('W1','W7')
% hold off
```

Global Simulation Scripts - In Storage Folder

Code Listing C.52: plotJFM_semiconfined.m, Matlab file to conglomerate, calculate and plot the results of Chapter 5.

```
%Plot data

resampleData = false;
if resampleData

    clear;clc;clf
    close all
    cd /storage/dksan/
    global simFigures
    simFigures = false; %% Define as true/false if I would like
       to see figures in the following sims
    data_coeffs = func_importSims('2023-0417b*'); %% Change this
        value to change the sampled dataset

    save("processedData_JFM_Coeffs","data_coeffs");
else
    fprintf('\nData has not been resampled\n')
    if exist('data_coeffs','var')
        fprintf('\nData has NOT been imported\n')
    else
        fprintf('\nData HAS been imported\n')
        load("processedData_JFM_Coeffs","data_coeffs");
    end
end

clear z_* paraSet paraRes* ax* resampleData


% CALCULATIONS
clear pmetric*
paraSet.flow = zeros(16,2);
paraSet.vel(1,1) = {'W0_001'};
paraSet.vel(2,1) = {'W0_002'};
paraSet.vel(3,1) = {'W0_005'};
paraSet.vel(4,1) = {'W0_007'};
paraSet.vel(5,1) = {'W0_010'};
paraSet.vel(6,1) = {'W0_025'};
paraSet.vel(7,1) = {'W0_050'};
paraSet.vel(8,1) = {'W0_075'};
paraSet.vel(9,1) = {'W0_100'};
paraSet.vel(10,1) = {'W1'};
paraSet.vel(11,1) = {'W2'};
paraSet.vel(12,1) = {'W3'};
paraSet.vel(13,1) = {'W4'};
```

```
paraSet.vel(14,1) = {'W5'};
paraSet.vel(15,1) = {'W6'};
paraSet.vel(16,1) = {'W7'};
paraSet.vel(17,1) = {'W8'};

paraSet.vel(1,2) = {'a_0417b0_001_case2_W0_001_Developed'};
paraSet.vel(2,2) = {'a_0417b0_002_case2_W0_002_Developed'};
paraSet.vel(3,2) = {'a_0417b0_005_case2_W0_005_Developed'};
paraSet.vel(4,2) = {'a_0417b0_007_case2_W0_007_Developed'};
paraSet.vel(5,2) = {'a_0417b0_010_case2_W0_010_Developed'};
paraSet.vel(6,2) = {'a_0417b0_025_case2_W0_025_Developed'};
paraSet.vel(7,2) = {'a_0417b0_050_case2_W0_050_Developed'};
paraSet.vel(8,2) = {'a_0417b0_075_case2_W0_075_Developed'};
paraSet.vel(9,2) = {'a_0417b0_100_case2_W0_100_Developed'};
paraSet.vel(10,2) = {'a_0417ba_case2_W1_Developed'};
paraSet.vel(11,2) = {'a_0417bb_case2_W2_Developed'};
paraSet.vel(12,2) = {'a_0417bc_case2_W3_Developed'};
paraSet.vel(13,2) = {'a_0417bd_case2_W4_Developed'};
paraSet.vel(14,2) = {'a_0417be_case2_W5_Developed'};
paraSet.vel(15,2) = {'a_0417bf_case2_W6_Developed'};
paraSet.vel(16,2) = {'a_0417bg_case2_W7_Developed'};
paraSet.vel(17,2) = {'a_0417bh_case2_W8_Developed'};

paraSet.flow(1,1) = 0.0275127287475132;
paraSet.flow(2,1) = 0.0687818218687829;
paraSet.flow(3,1) = 0.137563643737566;
paraSet.flow(4,1) = 0.206345465606349;
paraSet.flow(5,1) = 0.275127287475132;
paraSet.flow(6,1) = 0.687818218687829;
paraSet.flow(7,1) = 1.37563643737566;
paraSet.flow(8,1) = 2.06345465606349;
paraSet.flow(9,1) = 2.75127287475132;
paraSet.flow(10,1) = 5.33333333333333;
paraSet.flow(11,1) = 9.6;
paraSet.flow(12,1) = 13.8666666666667;
paraSet.flow(13,1) = 18.1333333333333;
paraSet.flow(14,1) = 22.4;
paraSet.flow(15,1) = 26.6666666666667;
paraSet.flow(16,1) = 30.9333333333333;
paraSet.flow(17,1) = 275.127287475132;

paraSet.characteristicLength = 0.364922;
paraSet.nu_water = 0.000001004;
paraSet.rho_water = 998.2;

% Calculate Reynolds Number
for para_i=1:length(paraSet.flow(:,1))
    paraSet.flow(para_i,2) = (paraSet.flow(para_i,1) * paraSet.
       characteristicLength)/paraSet.nu_water;
end
```

```
paraRes = func_calcJFMCoefficients(paraSet,data_coeffs);
paraRes.Strouhal(1,1) = NaN;
clf

%figure(1)
fig1_minVal = 1e3;
fig1_maxVal = 1e8;

figure(1)
%subplot(2,2,1)
semilogx(paraSet.flow(:,2),paraRes.C_side(:,1),'b*-')
xticks([1e3 1e5 1e7 1e8])
grid on
xlim([fig1_minVal fig1_maxVal])
ylim([0 1])
xlabel('Re')
ylabel('C_{side}')

figure(2)
%subplot(2,2,2)
semilogx(paraSet.flow(:,2),paraRes.C_lift(:,1),'b*-')
xticks([1e3 1e5 1e7 1e8])
grid on
xlim([fig1_minVal fig1_maxVal])
ylim([-.2 0.2])
xlabel('Re')
ylabel('C_{lift}')

figure(3)
% subplot(2,2,3)
semilogx(paraSet.flow(:,2),paraRes.C_rollover(:,1),'b*-')
xticks([1e3 1e5 1e7 1e8])
grid on
xlim([fig1_minVal fig1_maxVal])
ylim([0 2.5])
xlabel('Re')
ylabel('C_{rollover}')

% subplot(2,2,4)
% semilogx(paraSet.flow(:,2),paraRes.C_drag(:,1),'b*-')
% xticks([1e3 1e5 1e7 1e8])
% grid on
% xlim([fig1_minVal fig1_maxVal])
% ylim([0 2.5])
% xlabel('Re')
% ylabel('C_{drag}')

figure(4)
fig2_minVal = 1e3;
```

```
fig2_maxVal = 1e8;

semilogx(paraSet.flow(:,2),paraRes.C_drag(:,1),'b*-')
grid on
xlim([fig2_minVal fig2_maxVal])
ylim([0 2.5])
xlabel('Re')
ylabel('C_{drag}')

figure(5)
semilogx(paraSet.flow(:,2),paraRes.Strouhal(:,1),'b*-')
grid on
xlim([fig2_minVal fig2_maxVal])
ylim([0 0.3])
xlabel('Re')
ylabel('St')

%Plot of my rollover coefficient, normalized by Baker/Soper's 0.5
    value
%from Table 2
figure(6)
% subplot(2,2,3)
semilogx(paraSet.flow(:,2),paraRes.C_rollover(:,1)/0.5,'b*-')
xticks([1e3 1e5 1e7 1e8])
grid on
xlim([fig1_minVal fig1_maxVal])
ylim([0 4])
xlabel('Re')
ylabel('C_{RL}(90)/C_{RL}(30)')
```

Global Simulation Scripts - In Storage Folder

Code Listing C.53: plotJFM_unconfined.m, Matlab file to conglomerate, calculate and plot the results of Chapter 5.

```
%Plot data

resampleData = false;
if resampleData

    clear;clc;clf
    close all
    cd /storage/dksan/
    global simFigures
    simFigures = false; %% Define as true/false if I would like
       to see figures in the following sims
    data_coeffs_unconfined = func_importSims('2023-0620*'); %%
       Change this value to change the sampled dataset

    save("processedData_JFM_Coeffs_unconfined","
       data_coeffs_unconfined");
else
    fprintf('\nData has not been resampled\n')
    if exist('data_coeffs_unconfined','var')
        fprintf('\nData has NOT been imported\n')
    else
        fprintf('\nData HAS been imported\n')
        load("processedData_JFM_Coeffs_unconfined","
           data_coeffs_unconfined");
    end

end

clear z_* paraSet paraRes* ax* resampleData

%data_coeffs_unconfined.b_NA_data = {};
data_coeffs_unconfined.b_NA_data.d_forceIncomp_inlet_c2.data.
   F_p_x = [0];
data_coeffs_unconfined.b_NA_data.d_forceIncomp_inlet_c2.data.
   F_visc_x = [0];
data_coeffs_unconfined.b_NA_data.d_forceIncomp_case2.data.F_p_y =
    [0];
data_coeffs_unconfined.b_NA_data.d_forceIncomp_case2.data.
   F_visc_y = [0];
data_coeffs_unconfined.b_NA_data.d_forceIncomp_case2.data.F_p_x =
    [0];
data_coeffs_unconfined.b_NA_data.d_forceIncomp_case2.data.
   F_visc_x = [0];
data_coeffs_unconfined.b_NA_data.maxFrequency_AtMaxYValue = [0];
```

```
% CALCULATIONS
clear pmetric*
paraSet.flow = zeros(16,2);
paraSet.vel(1,1) = {'W0_001'};
paraSet.vel(2,1) = {'W0_002'};
paraSet.vel(3,1) = {'W0_005'};
paraSet.vel(4,1) = {'W0_007'};
paraSet.vel(5,1) = {'W0_010'};
paraSet.vel(6,1) = {'W0_025'};
paraSet.vel(7,1) = {'W0_050'};
paraSet.vel(8,1) = {'W0_075'};
paraSet.vel(9,1) = {'W0_100'};
paraSet.vel(10,1) = {'W1'};
paraSet.vel(11,1) = {'W2'};
paraSet.vel(12,1) = {'W3'};
paraSet.vel(13,1) = {'W4'};
paraSet.vel(14,1) = {'W5'};
paraSet.vel(15,1) = {'W6'};
paraSet.vel(16,1) = {'W7'};
paraSet.vel(17,1) = {'W8'};

paraSet.vel(1,2) = {'a_0620ca_unconfined_W0_001_Uniform'};
paraSet.vel(2,2) = {'a_0620cb_unconfined_W0_002_Uniform'};
paraSet.vel(3,2) = {'a_0620cc_unconfined_W0_005_Uniform'};
paraSet.vel(4,2) = {'a_0620cd_unconfined_W0_007_Uniform'};
paraSet.vel(5,2) = {'a_0620ea_unconfined_W0_010_Uniform'};
paraSet.vel(6,2) = {'a_0620eb_unconfined_W0_025_Uniform'};
paraSet.vel(7,2) = {'a_0620ec_unconfined_W0_050_Uniform'};
paraSet.vel(8,2) = {'a_0620ed_unconfined_W0_075_Uniform'};
paraSet.vel(9,2) = {'a_0620ee_unconfined_W0_100_Uniform'};
paraSet.vel(10,2) = {'a_0620ga_unconfined_W1_Uniform'};
paraSet.vel(11,2) = {'a_0620gb_unconfined_W2_Uniform'};
paraSet.vel(12,2) = {'a_0620gc_unconfined_W3_Uniform'};
paraSet.vel(13,2) = {'a_0620gd_unconfined_W4_Uniform'};
paraSet.vel(14,2) = {'a_0620ge_unconfined_W5_Uniform'};
paraSet.vel(15,2) = {'a_0620gf_unconfined_W6_Uniform'};
paraSet.vel(16,2) = {'a_0620gg_unconfined_W7_Uniform'};
paraSet.vel(17,2) = {'a_0620gh_unconfined_W8_Uniform'};

paraSet.flow(1,1) = 0.0275127287475132;
paraSet.flow(2,1) = 0.0687818218687829;
paraSet.flow(3,1) = 0.137563643737566;
paraSet.flow(4,1) = 0.206345465606349;
paraSet.flow(5,1) = 0.275127287475132;
paraSet.flow(6,1) = 0.687818218687829;
paraSet.flow(7,1) = 1.37563643737566;
paraSet.flow(8,1) = 2.06345465606349;
paraSet.flow(9,1) = 2.75127287475132;
```

```
paraSet.flow(10,1) = 5.33333333333333;
paraSet.flow(11,1) = 9.6;
paraSet.flow(12,1) = 13.8666666666667;
paraSet.flow(13,1) = 18.1333333333333;
paraSet.flow(14,1) = 22.4;
paraSet.flow(15,1) = 26.6666666666667;
paraSet.flow(16,1) = 30.9333333333333;
paraSet.flow(17,1) = 275.127287475132;

paraSet.characteristicLength = 0.364922;
paraSet.nu_water = 0.000001004;
paraSet.rho_water = 998.2;

% Calculate Reynolds Number
for para_i=1:length(paraSet.flow(:,1))
    paraSet.flow(para_i,2) = (paraSet.flow(para_i,1) * paraSet.
       characteristicLength)/paraSet.nu_water;
end

paraRes = func_calcJFMCoefficients(paraSet,data_coeffs_unconfined
   );

clf

%figure(1)
fig1_minVal = 1e3;
fig1_maxVal = 1e8;

figure(1)
%subplot(2,2,1)
semilogx(paraSet.flow(:,2),paraRes.C_side(:,1),'b*-')
xticks([1e3 1e5 1e7 1e8])
grid on
xlim([fig1_minVal fig1_maxVal])
ylim([0 1])
xlabel('Re')
ylabel('C_{side}')

figure(2)
%subplot(2,2,2)
semilogx(paraSet.flow(:,2),paraRes.C_lift(:,1),'b*-')
xticks([1e3 1e5 1e7 1e8])
grid on
xlim([fig1_minVal fig1_maxVal])
ylim([-.2 0.2])
xlabel('Re')
ylabel('C_{lift}')

figure(3)
%subplot(2,2,3)
```

```
semilogx(paraSet.flow(:,2),paraRes.C_rollover(:,1),'b*-')
xticks([1e3 1e5 1e7 1e8])
grid on
xlim([fig1_minVal fig1_maxVal])
ylim([0 2.5])
xlabel('Re')
ylabel('C_{rollover}')

% subplot(2,2,4)
% semilogx(paraSet.flow(:,2),paraRes.C_drag(:,1),'b*-')
% xticks([1e3 1e5 1e7 1e8])
% grid on
% xlim([fig1_minVal fig1_maxVal])
% ylim([0 2.5])
% xlabel('Re')
% ylabel('C_{drag}')

figure(4)
fig2_minVal = 1e3;
fig2_maxVal = 1e8;
%Blevins data
paraRes.Blevins_C_drag = [1.1; 1.2; 0.4; 0.8];
paraRes.Blevins_Re = [1e4; 1e5; 1e6; 1e7];

semilogx(paraSet.flow(:,2),paraRes.C_drag(:,1),'b*-')
%hold on
grid on
%semilogx(paraRes.Blevins_Re(:,1),paraRes.Blevins_C_drag(:,1),'b
    *-')
xlim([fig2_minVal fig2_maxVal])
ylim([0 2.5])
xlabel('Re')
ylabel('C_{drag}')
%hold off

figure(5)
semilogx(paraSet.flow(:,2),paraRes.Strouhal(:,1),'b*-')
grid on
xlim([fig2_minVal fig2_maxVal])
ylim([0 0.3])
xlabel('Re')
ylabel('St')
```

Global Simulation Scripts - In Storage Folder

Code Listing C.54: plotVnV_Validation_DragCoefficientInterpolation.m, Matlab file to plot the validation results of Chapter 4.

```
clear
load("savedData_VnV_Validation_DragCoefficientInterp.mat")

intRatio = 0.3239/0.364922; % Cylinder Width/Height
intDragCoeff = interp1(dragCoefficient(6:7,1),dragCoefficient
    (6:7,2),intRatio);

figure(1)
plot(dragCoefficient(:,1),dragCoefficient(:,2),'-*')
hold on
grid on
plot(intRatio,intDragCoeff,'^')
ylabel('C_{drag}')
xlabel('Rectangular Cylinder Width to Height Ratio')
%title('Interpolated Drag Coefficient')
legend('Blevins, 1984','Interpolated Target Drag Coefficient')
ylim([0 3])
hold off
```

Global Simulation Scripts - In Storage Folder

Code Listing C.55: plotVnV_Validation_InflationLayers.m, Matlab file to conglomerate, calculate and plot the results of Chapter 4.

```
%Plot data

resampleData = false;
if resampleData

    clear;clc;clf
    close all
    cd /storage/dksan/
    data_Validation = func_importSims('2023-0522*');
    %data_H1     = func_importSims('2023-0305*');
    %data_H2     = func_importSims('2023-0306*');
    %mergestructs = @(data_H1,data_H2) cell2struct([struct2cell(
       data_H1);struct2cell(data_H2)],[fieldnames(data_H1);
       fieldnames(data_H2)]);
    %data_H = mergestructs(data_H1, data_H2);

    clear data_H1 data_H2 mergestructs

    save("processedData_VnV_Validation","data_Validation");
else
    fprintf('\nData has not been resampled\n')
    if exist('data_Validation','var')
        fprintf('\nData has NOT been imported\n')
    else
        fprintf('\nData HAS been imported\n')
        load("processedData_VnV_Validation","data_Validation");
    end
end

clear z_* paraSet paraRes* ax* resampleData


% CALCULATIONS
clear pmetric*
paraSet.flow = zeros(2,2);
paraSet.vel(1,1) = {'W0'};
%paraSet.vel(1,1) = {'W1'};
%paraSet.vel(3,1) = {'W2'};
%paraSet.vel(4,1) = {'W3'};
%paraSet.vel(5,1) = {'W4'};
%paraSet.vel(6,1) = {'W5'};
%paraSet.vel(7,1) = {'W6'};
%paraSet.vel(2,1) = {'W7'};
```

```
paraSet.vel(1,2) = {'a_0522_case2_W0'};
%paraSet.vel(1,2) = {'a_0417ba_case2_W1'};
%paraSet.vel(3,2) = {'a_0417bb_case2_W2'};
%paraSet.vel(4,2) = {'a_0417bc_case2_W3'};
%paraSet.vel(5,2) = {'a_0417bd_case2_W4'};
%paraSet.vel(6,2) = {'a_0417be_case2_W5'};
%paraSet.vel(7,2) = {'a_0417bf_case2_W6'};
%paraSet.vel(2,2) = {'a_0417bg_case2_W7'};

paraSet.flow(1,1) = 0.275127287475132;
%paraSet.flow(1,1) = 5.33333333333333;
%paraSet.flow(3,1) = 9.6;
%paraSet.flow(4,1) = 13.8666666666667;
%paraSet.flow(5,1) = 18.1333333333333;
%paraSet.flow(6,1) = 22.4;
%paraSet.flow(7,1) = 26.6666666666667;
%paraSet.flow(2,1) = 30.9333333333333;

paraSet.characteristicLength = 0.364922;
paraSet.nu_water = 0.000001004;
paraSet.rho_water = 998.2;

paraSet.layers_case2 = linspace(0,8,5);
%paraSet.USlength = linspace(2,20,19);
%paraSet.DSlength = linspace(2,15,14);
%paraSet.H_W1length= linspace(2,16,8);
%paraSet.H_W7length= linspace(2,20,10);
paraSet.dataNames_Validation = fieldnames(data_Validation);
%paraSet.dataNames_H     = fieldnames(data_H);

% Calculate Reynolds Number
%for para_i=1:length(paraSet.flow(:,1))
%     paraSet.flow(para_i,2) = (paraSet.flow(para_i,1) * paraSet.
   characteristicLength)/paraSet.nu_water;
%end

% Merge data into a single dataset.
for paraRes_i2=1:(length(paraSet.layers_case2)); paraRes.
   Cd_Layers_W0(paraRes_i2,1)      = data_Validation.(char(paraSet
   .dataNames_Validation(paraRes_i2,1))).d_mean.Cd; end
for paraRes_i3=1:(length(paraSet.layers_case2)); paraRes.
   yPlus_excelPts(paraRes_i3,:)    = data_Validation.(char(paraSet
   .dataNames_Validation(paraRes_i3,1))).r_yPlus.excelPoints; end
% for paraRes_i32=1:(length(paraSet.layers_case2)); paraRes.
   c2bottom_max_095percentile(paraRes_i32,1)    = data_Validation
   .(    char(paraSet.dataNames_Validation(       paraRes_i32,1))).
   r_yPlus.case2_bottom.max_095Prtile; end
% for paraRes_i33=1:(length(paraSet.layers_case2)); paraRes.
   c2bottom_max_095percentile(paraRes_i33,1)    = data_Validation
```

```
    .(    char(paraSet.dataNames_Validation(      paraRes_i33,1))).
    r_yPlus.case2_bottom.max_095Prtile; end
% for paraRes_i34=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_max_095percentile(paraRes_i34,1)   = data_Validation
    .(    char(paraSet.dataNames_Validation(      paraRes_i34,1))).
    r_yPlus.case2_bottom.max_095Prtile; end
% for paraRes_i35=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_max_095percentile(paraRes_i35,1)   = data_Validation
    .(    char(paraSet.dataNames_Validation(      paraRes_i35,1))).
    r_yPlus.case2_bottom.max_095Prtile; end
% for paraRes_i4=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_min_005percentile(paraRes_i4,1)    = data_Validation.(
        char(paraSet.dataNames_Validation(      paraRes_i4,1))).
    r_yPlus.case2_bottom.min_005Prtile; end
% for paraRes_i5=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_mean(paraRes_i5,1)                 = data_Validation.(
        char(paraSet.dataNames_Validation(      paraRes_i5,1))).
    r_yPlus.case2_bottom.mean(1,3); end
%for paraRes_i0=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_min_005percentile(paraRes_i0,1)    = data_Validation.(
        char(paraSet.dataNames_Validation(      paraRes_i0,1))).
    r_yPlus.case2_bottom.min_005Prtile; end
clear paraRes_i* para_i

% Calculate % Error based on prescribed value
% paraRes.H_W1(:,2)  = (paraRes.H_W1(:,1) /paraRes.H_W1(6,1)  -
    1) * 100;
% paraRes.H_W7(:,2)  = (paraRes.H_W7(:,1) /paraRes.H_W7(6,1)  -
    1) * 100;
% paraRes.US_W1(:,2) = (paraRes.US_W1(:,1)/paraRes.US_W1(9,1) -
    1) * 100;
% paraRes.US_W7(:,2) = (paraRes.US_W7(:,1)/paraRes.US_W7(9,1) -
    1) * 100;
% paraRes.DS_W1(:,2) = (paraRes.DS_W1(:,1)/paraRes.DS_W1(6,1) -
    1) * 100;
% paraRes.DS_W7(:,2) = (paraRes.DS_W7(:,1)/paraRes.DS_W7(9,1) -
    1) * 100;

clf

Cd_interp = 2.2562065318068;
for i=1:(length(paraRes.Cd_Layers_W0(:,1)))
    paraRes.errorVal(i,1) = paraRes.Cd_Layers_W0(i,1) - Cd_interp
        ;
    paraRes.errorPer(i,1) = (100*paraRes.Cd_Layers_W0(i,1)/
        Cd_interp) - 100;
    fprintf('Layer Count = %d. Cd = %.4f; Error = %.4f %%\n',
        paraSet.layers_case2(1,i), paraRes.Cd_Layers_W0(i,1),
        paraRes.errorPer(i,1))
end
```

```
fprintf('\nMax Error %.4f %%\n',max(abs(paraRes.errorPer)))
fprintf(    'Min Error %.4f %%\n',min(abs(paraRes.errorPer)))

vnv_tbl = horzcat(transpose(paraSet.layers_case2),paraRes.
   Cd_Layers_W0,paraRes.errorPer)
vnv_tbl2 = vnv_tbl
vnv_tbl2(:,2) = round(vnv_tbl(:,2),5);
vnv_tbl2(:,3) = round(vnv_tbl(:,3),5);

figure(1)
%subplot(2,1,1)
ax11.line1 = plot(paraSet.layers_case2,paraRes.Cd_Layers_W0(:,1)
   ,'b*-');
hold on
ax11.line2 = plot([0,8],[2.256206532, 2.256206532],'s-','Color
   ',[1 0.5 0]);
grid on
%grid minor
ylim([0 2.5]);
xlim([0 8]);
xlabel('Number of Inflation Layers')
ylabel('C_{drag}')
title('Validation of Unconfined Rectangular Cylinder')
hold off
%
% subplot(2,1,2)
% ax12.line1 = plot(paraSet.H_W1length,paraRes.H_W1(:,2),'b*-');
% hold on
% ax12.line2 = plot([2.256206532, 2.256206532],paraRes.Layers_W0
   (:,1),'s-','Color',[1 0.5 0]);
% grid on
% grid minor
% ylim([-20    ceil(max(max(paraRes.H_W7(:,2),paraRes.H_W7(:,2)))
   /10)*10]);
% xlim([0 max(paraSet.H_W7length)+2]);
% xlabel('Domain Height (m)')
% ylabel('% Relative Error')
% %title('Upstream Verification')
% legend('W1','W7')
% hold off
%
% figure(2)
% subplot(2,1,1)
% ax21.line1 = plot(paraSet.USlength,paraRes.US_W1(:,1),'b*-');
% hold on
% ax21.line2 = plot(paraSet.USlength,paraRes.US_W7(:,1),'s-','
   Color',[1 0.5 0]);
% grid on
% grid minor
```

```
% ylim([0 ceil(max(max(paraRes.US_W1(:,1),paraRes.US_W7(:,1))))])
    ;
% xlim([0 max(paraSet.USlength)+2]);
% %xlabel('Length Upstream of Cylinder Centroid (m)')
% ylabel('C_{drag}')
% title('Upstream Verification')
% hold off
%
% subplot(2,1,2)
% ax22.line1 = plot(paraSet.USlength,paraRes.US_W1(:,2),'b*-');
% hold on
% ax22.line2 = plot(paraSet.USlength,paraRes.US_W7(:,2),'s-','
    Color',[1 0.5 0]);
% grid on
% grid minor
% ylim([-20    ceil(max(max(paraRes.US_W1(:,2),paraRes.US_W7(:,2))
    )/10)*10]);
% xlim([0 max(paraSet.USlength)+2]);
% xlabel('Length Upstream of Cylinder Centroid (m)')
% ylabel('% Relative Error')
% %title('Upstream Verification')
% legend('W1','W7')
% hold off
%
% figure(3)
% subplot(2,1,1)
% ax3.line1 = plot(paraSet.DSlength,paraRes.DS_W1(:,1),'b*-');
% hold on
% ax3.line2 = plot(paraSet.DSlength,paraRes.DS_W7(:,1),'s-','
    Color',[1 0.5 0]);
% grid on
% grid minor
% ylim([0 ceil(max(max(paraRes.DS_W1(:,1),paraRes.DS_W7(:,1))))])
    ;
% xlim([0 max(paraSet.DSlength)+1]);
% %xlabel('Length Downstream of Cylinder Centroid (m)')
% ylabel('C_{drag}')
% title('Downstream Verification')
% hold off
%
% subplot(2,1,2)
% ax3.line1 = plot(paraSet.DSlength,paraRes.DS_W1(:,2),'b*-');
% hold on
% ax3.line2 = plot(paraSet.DSlength,paraRes.DS_W7(:,2),'s-','
    Color',[1 0.5 0]);
% grid on
% grid minor
% ylim([-30    ceil(max(max(paraRes.DS_W1(:,2),paraRes.DS_W7(:,2))
    )/10)*10]);
% xlim([0 max(paraSet.DSlength)+1]);
```

```
% xlabel('Length Downstream of Cylinder Centroid (m)')
% ylabel('% Relative Error')
% %title('Downstream Verification')
% legend('W1','W7')
% hold off
```

Global Simulation Scripts - In Storage Folder

Code Listing C.56: plotVnV_Verification_GridIndependence.m, Matlab file to conglomerate, calculate and plot the results of Chapter 4.

```
%Plot data

resampleData = false;
if resampleData

    clear;clc;clf
    close all
    cd /storage/dksan/
    global simFigures
    simFigures = false; %% Define as true/false if I would like
       to see figures in the following sims
    data_Verification = func_importSims('2023-0406b*');

    save("processedData_VnV_Verification_GridIndependence","
       data_Verification");
else
    fprintf('\nData has not been resampled\n')
    if exist('data_Verification','var')
        fprintf('\nData has NOT been imported\n')
    else
        fprintf('\nData HAS been imported\n')
        load("processedData_VnV_Verification_GridIndependence","
           data_Verification");
    end
end

clear z_* paraSet paraRes* ax* resampleData

if isfield(data_Verification,'a_0406b3_case2_W1_Uniform')
    data_Verification = rmfield(data_Verification,'
       a_0406b3_case2_W1_Uniform');
end

% CALCULATIONS
clear pmetric*
paraSet.flow = zeros(2,2);
%paraSet.vel(1,1) = {'W0'};
paraSet.vel(1,1) = {'W1'};
%paraSet.vel(3,1) = {'W2'};
%paraSet.vel(4,1) = {'W3'};
%paraSet.vel(5,1) = {'W4'};
%paraSet.vel(6,1) = {'W5'};
%paraSet.vel(7,1) = {'W6'};
%paraSet.vel(2,1) = {'W7'};
```

```
%paraSet.vel(1,2) = {'a_0522_case2_W0'};
paraSet.vel(1,2) = {'a_0406b_case2_W1'};
%paraSet.vel(3,2) = {'a_0417bb_case2_W2'};
%paraSet.vel(4,2) = {'a_0417bc_case2_W3'};
%paraSet.vel(5,2) = {'a_0417bd_case2_W4'};
%paraSet.vel(6,2) = {'a_0417be_case2_W5'};
%paraSet.vel(7,2) = {'a_0417bf_case2_W6'};
%paraSet.vel(2,2) = {'a_0417bg_case2_W7'};

%paraSet.flow(1,1) = 0.275127287475132;
paraSet.flow(1,1) = 5.33333333333333;
%paraSet.flow(3,1) = 9.6;
%paraSet.flow(4,1) = 13.8666666666667;
%paraSet.flow(5,1) = 18.1333333333333;
%paraSet.flow(6,1) = 22.4;
%paraSet.flow(7,1) = 26.6666666666667;
%paraSet.flow(2,1) = 30.9333333333333;

paraSet.characteristicLength = 0.364922;
paraSet.nu_water = 0.000001004;
paraSet.rho_water = 998.2;

paraSet.cellCount = [100470 142362 515461 2007219 7979443];
%paraSet.USlength = linspace(2,20,19);
%paraSet.DSlength = linspace(2,15,14);
%paraSet.H_W1length= linspace(2,16,8);
%paraSet.H_W7length= linspace(2,20,10);
paraSet.dataNames_Verification = fieldnames(data_Verification);
%paraSet.dataNames_H     = fieldnames(data_H);

% Calculate Reynolds Number
%for para_i=1:length(paraSet.flow(:,1))
%     paraSet.flow(para_i,2) = (paraSet.flow(para_i,1) * paraSet.
   characteristicLength)/paraSet.nu_water;
%end

% Merge data into a single dataset.
for paraRes_i2=1:(length(paraSet.cellCount)); paraRes.
   Cd_gridIndependence(paraRes_i2,1)      = data_Verification.(
   char(paraSet.dataNames_Verification(paraRes_i2,1))).d_mean.Cd;
    end
%for paraRes_i3=1:(length(paraSet.cellCount)); paraRes.
   yPlus_excelPts(paraRes_i3,:)    = data_Verification.(char(
   paraSet.dataNames_Verification(paraRes_i3,1))).r_yPlus.
   excelPoints; end
% for paraRes_i32=1:(length(paraSet.layers_case2)); paraRes.
   c2bottom_max_095percentile(paraRes_i32,1)    = data_Validation
   .(    char(paraSet.dataNames_Validation(       paraRes_i32,1))).
   r_yPlus.case2_bottom.max_095Prtile; end
```

```
% for paraRes_i33=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_max_095percentile(paraRes_i33,1)    = data_Validation
    .(    char(paraSet.dataNames_Validation(       paraRes_i33,1))).
    r_yPlus.case2_bottom.max_095Prtile; end
% for paraRes_i34=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_max_095percentile(paraRes_i34,1)    = data_Validation
    .(    char(paraSet.dataNames_Validation(       paraRes_i34,1))).
    r_yPlus.case2_bottom.max_095Prtile; end
% for paraRes_i35=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_max_095percentile(paraRes_i35,1)    = data_Validation
    .(    char(paraSet.dataNames_Validation(       paraRes_i35,1))).
    r_yPlus.case2_bottom.max_095Prtile; end
% for paraRes_i4=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_min_005percentile(paraRes_i4,1)    = data_Validation.(
       char(paraSet.dataNames_Validation(       paraRes_i4,1))).
    r_yPlus.case2_bottom.min_005Prtile; end
% for paraRes_i5=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_mean(paraRes_i5,1)                  = data_Validation.(
       char(paraSet.dataNames_Validation(       paraRes_i5,1))).
    r_yPlus.case2_bottom.mean(1,3); end
%for paraRes_i0=1:(length(paraSet.layers_case2)); paraRes.
    c2bottom_min_005percentile(paraRes_i0,1)    = data_Validation.(
       char(paraSet.dataNames_Validation(       paraRes_i0,1))).
    r_yPlus.case2_bottom.min_005Prtile; end
clear paraRes_i* para_i

% Calculate % Error based on prescribed value
% paraRes.H_W1(:,2)  = (paraRes.H_W1(:,1) /paraRes.H_W1(6,1)  -
    1) * 100;
% paraRes.H_W7(:,2)  = (paraRes.H_W7(:,1) /paraRes.H_W7(6,1)  -
    1) * 100;
% paraRes.US_W1(:,2) = (paraRes.US_W1(:,1)/paraRes.US_W1(9,1) -
    1) * 100;
% paraRes.US_W7(:,2) = (paraRes.US_W7(:,1)/paraRes.US_W7(9,1) -
    1) * 100;
% paraRes.DS_W1(:,2) = (paraRes.DS_W1(:,1)/paraRes.DS_W1(6,1) -
    1) * 100;
% paraRes.DS_W7(:,2) = (paraRes.DS_W7(:,1)/paraRes.DS_W7(9,1) -
    1) * 100;

clf

% Cd_interp = 2.2562065318068;
% for i=1:(length(paraRes.Cd_Layers_W0(:,1)))
%     paraRes.errorVal(i,1) = paraRes.Cd_Layers_W0(i,1) -
    Cd_interp;
%     paraRes.errorPer(i,1) = (100*paraRes.Cd_Layers_W0(i,1)/
    Cd_interp) - 100;
```

```
%     fprintf('Layer Count = %d. Cd = %.4f; Error = %.4f %%\n',
    paraSet.layers_case2(1,i), paraRes.Cd_Layers_W0(i,1), paraRes.
    errorPer(i,1))
% end
% fprintf('\nMax Error %.4f %%\n',max(abs(paraRes.errorPer)))
% fprintf(    'Min Error %.4f %%\n',min(abs(paraRes.errorPer)))

% vnv_tbl = horzcat(transpose(paraSet.layers_case2),paraRes.
    Cd_Layers_W0,paraRes.errorPer)
% vnv_tbl2 = vnv_tbl
% vnv_tbl2(:,2) = round(vnv_tbl(:,2),5);
% vnv_tbl2(:,3) = round(vnv_tbl(:,3),5);

figure(1)
%subplot(2,1,1)
semilogx(paraSet.cellCount,paraRes.Cd_gridIndependence(:,1),'b
    *-');
hold on
%ax11.line2 = plot([0,8],[2.256206532, 2.256206532],'s-','Color
    ',[1 0.5 0]);
grid on
%grid minor
ylim([0 2.5]);
xlim([5e4 1e7]);
xlabel('Number of Cells')
ylabel('C_{drag}')
%title('Traditional Grid Refinment Verification')
hold off
%
% subplot(2,1,2)
% ax12.line1 = plot(paraSet.H_W1length,paraRes.H_W1(:,2),'b*-');
% hold on
% ax12.line2 = plot([2.256206532, 2.256206532],paraRes.Layers_W0
    (:,1),'s-','Color',[1 0.5 0]);
% grid on
% grid minor
% ylim([-20    ceil(max(max(paraRes.H_W7(:,2),paraRes.H_W7(:,2)))
    /10)*10]);
% xlim([0 max(paraSet.H_W7length)+2]);
% xlabel('Domain Height (m)')
% ylabel('% Relative Error')
% %title('Upstream Verification')
% legend('W1','W7')
% hold off
%
% figure(2)
% subplot(2,1,1)
% ax21.line1 = plot(paraSet.USlength,paraRes.US_W1(:,1),'b*-');
% hold on
```

```
% ax21.line2 = plot(paraSet.USlength,paraRes.US_W7(:,1),'s-','
    Color',[1 0.5 0]);
% grid on
% grid minor
% ylim([0 ceil(max(max(paraRes.US_W1(:,1),paraRes.US_W7(:,1))))])
    ;
% xlim([0 max(paraSet.USlength)+2]);
% %xlabel('Length Upstream of Cylinder Centroid (m)')
% ylabel('C_{drag}')
% title('Upstream Verification')
% hold off
%
% subplot(2,1,2)
% ax22.line1 = plot(paraSet.USlength,paraRes.US_W1(:,2),'b*-');
% hold on
% ax22.line2 = plot(paraSet.USlength,paraRes.US_W7(:,2),'s-','
    Color',[1 0.5 0]);
% grid on
% grid minor
% ylim([-20    ceil(max(max(paraRes.US_W1(:,2),paraRes.US_W7(:,2))
    )/10)*10]);
% xlim([0 max(paraSet.USlength)+2]);
% xlabel('Length Upstream of Cylinder Centroid (m)')
% ylabel('% Relative Error')
% %title('Upstream Verification')
% legend('W1','W7')
% hold off
%
% figure(3)
% subplot(2,1,1)
% ax3.line1 = plot(paraSet.DSlength,paraRes.DS_W1(:,1),'b*-');
% hold on
% ax3.line2 = plot(paraSet.DSlength,paraRes.DS_W7(:,1),'s-','
    Color',[1 0.5 0]);
% grid on
% grid minor
% ylim([0 ceil(max(max(paraRes.DS_W1(:,1),paraRes.DS_W7(:,1))))])
    ;
% xlim([0 max(paraSet.DSlength)+1]);
% %xlabel('Length Downstream of Cylinder Centroid (m)')
% ylabel('C_{drag}')
% title('Downstream Verification')
% hold off
%
% subplot(2,1,2)
% ax3.line1 = plot(paraSet.DSlength,paraRes.DS_W1(:,2),'b*-');
% hold on
% ax3.line2 = plot(paraSet.DSlength,paraRes.DS_W7(:,2),'s-','
    Color',[1 0.5 0]);
% grid on
```

```
% grid minor
% ylim([-30    ceil(max(max(paraRes.DS_W1(:,2),paraRes.DS_W7(:,2))
    )/10)*10]);
% xlim([0 max(paraSet.DSlength)+1]);
% xlabel('Length Downstream of Cylinder Centroid (m)')
% ylabel('% Relative Error')
% %title('Downstream Verification')
% legend('W1','W7')
% hold off
```

Global Simulation Scripts - In Storage Folder

Code Listing C.57: plotVnV_Verification_HUSDS.m, Matlab file to conglomerate, calculate and plot the results of Chapter 4.

```
%Plot data

resampleData = false;
if resampleData

    clear;clc;clf
    close all
    cd /storage/dksan/
    global simFigures
    simFigures = false; %% Define as true/false if I would like
       to see figures in the following sims
    data_USDS = func_importSims('2023-032*');
    data_H1   = func_importSims('2023-0305*');
    data_H2   = func_importSims('2023-0306*');
    mergestructs = @(data_H1,data_H2) cell2struct([struct2cell(
       data_H1);struct2cell(data_H2)],[fieldnames(data_H1);
       fieldnames(data_H2)]);
    data_H = mergestructs(data_H1, data_H2);

    clear data_H1 data_H2 mergestructs

    save("processedData_VnV_Verification_H","data_H");
    save("processedData_VnV_Verification_USDS","data_USDS");
else
    fprintf('\nData has not been resampled\n')
    if exist('data_H','var')
        fprintf('\nData has NOT been imported\n')
    else%title('Downstream Verification')
        fprintf('\nData HAS been imported\n')
        load("processedData_VnV_Verification_H","data_H");
        load("processedData_VnV_Verification_USDS","data_USDS");
    end
end

clear z_* paraSet paraRes ax* resampleData


% CALCULATIONS
clear pmetric*
paraSet.flow = zeros(2,2);
%paraSet.vel(1,1) = {'W0'};
paraSet.vel(1,1) = {'W1'};
%paraSet.vel(3,1) = {'W2'};
%paraSet.vel(4,1) = {'W3'};
```

```
%paraSet.vel(5,1) = {'W4'};
%paraSet.vel(6,1) = {'W5'};
%paraSet.vel(7,1) = {'W6'};
paraSet.vel(2,1) = {'W7'};

%paraSet.vel(1,2) = {'a_0417b0_case2_W0'};
paraSet.vel(1,2) = {'a_0417ba_case2_W1'};
%paraSet.vel(3,2) = {'a_0417bb_case2_W2'};
%paraSet.vel(4,2) = {'a_0417bc_case2_W3'};
%paraSet.vel(5,2) = {'a_0417bd_case2_W4'};
%paraSet.vel(6,2) = {'a_0417be_case2_W5'};
%paraSet.vel(7,2) = {'a_0417bf_case2_W6'};
paraSet.vel(2,2) = {'a_0417bg_case2_W7'};

%paraSet.flow(1,1) = 0.275127287475132;
paraSet.flow(1,1) = 5.33333333333333;
%paraSet.flow(3,1) = 9.6;
%paraSet.flow(4,1) = 13.8666666666667;
%paraSet.flow(5,1) = 18.1333333333333;
%paraSet.flow(6,1) = 22.4;
%paraSet.flow(7,1) = 26.6666666666667;
paraSet.flow(2,1) = 30.9333333333333;

paraSet.characteristicLength = 0.364922;
paraSet.nu_water = 0.000001004;
paraSet.rho_water = 998.2;

paraSet.USlength = linspace(2,20,19);
paraSet.DSlength = horzcat(linspace(2,20,19)); % Here is where
   you add to include new simulations
paraSet.H_W1length= linspace(2,16,8);
paraSet.H_W7length= linspace(2,20,10);
paraSet.dataNames_USDS = fieldnames(data_USDS);
paraSet.dataNames_H    = fieldnames(data_H);

% Calculate Reynolds Number
for para_i=1:length(paraSet.flow(:,1))
    paraSet.flow(para_i,2) = (paraSet.flow(para_i,1) * paraSet.
       characteristicLength)/paraSet.nu_water;
end

% Merge Cd data into a single dataset.
for paraRes_i2=1:(length(paraSet.H_W1length)); paraRes.H_W1(
   paraRes_i2,1)  = data_H.(    char(paraSet.dataNames_H(     10
                                                          +
   paraRes_i2,1))).d_mean.Cd; end
for paraRes_i3=1:(length(paraSet.H_W7length)); paraRes.H_W7(
   paraRes_i3,1)  = data_H.(    char(paraSet.dataNames_H(

   paraRes_i3,1))).d_mean.Cd; end
```

```
for paraRes_i4=1:(length(paraSet.USlength));    paraRes.US_W1(
    paraRes_i4,1) = data_USDS.(char(paraSet.dataNames_USDS(

    paraRes_i4,1))).d_mean.Cd; end
for paraRes_i5=1:(length(paraSet.USlength));    paraRes.US_W7(
    paraRes_i5,1) = data_USDS.(char(paraSet.dataNames_USDS( length
    (paraSet.USlength)                                  +paraRes_i5
    ,1))).d_mean.Cd; end
for paraRes_i6=1:(length(paraSet.DSlength));    paraRes.DS_W1(
    paraRes_i6,1) = data_USDS.(char(paraSet.dataNames_USDS( length
    (paraSet.USlength)*2                                +paraRes_i6
    ,1))).d_mean.Cd; end
for paraRes_i7=1:(length(paraSet.DSlength));    paraRes.DS_W7(
    paraRes_i7,1) = data_USDS.(char(paraSet.dataNames_USDS([length
    (paraSet.USlength)*2+length(paraSet.DSlength)]   +paraRes_i7
    ,1))).d_mean.Cd; end

clear paraRes_i* para_i

% Calculate % Error based on prescribed value
paraRes.H_W1(:,2)  = (paraRes.H_W1(:,1) /paraRes.H_W1(6,1)  - 1)
    * 100;
paraRes.H_W7(:,2)  = (paraRes.H_W7(:,1) /paraRes.H_W7(6,1)  - 1)
    * 100;
paraRes.US_W1(:,2) = (paraRes.US_W1(:,1)/paraRes.US_W1(9,1) - 1)
    * 100;
paraRes.US_W7(:,2) = (paraRes.US_W7(:,1)/paraRes.US_W7(9,1) - 1)
    * 100;
paraRes.DS_W1(:,2) = (paraRes.DS_W1(:,1)/paraRes.DS_W1(6,1) - 1)
    * 100;
paraRes.DS_W7(:,2) = (paraRes.DS_W7(:,1)/paraRes.DS_W7(6,1) - 1)
    * 100;

clf

figure(1)
subplot(2,1,1)
ax11.line1 = plot(paraSet.H_W1length,paraRes.H_W1(:,1),'b*-');
hold on
ax11.line2 = plot(paraSet.H_W7length,paraRes.H_W7(:,1),'s-','
    Color',[0.8500 0.3250 0.0980]);
grid on
grid minor
ylim([0 ceil(max(max(paraRes.H_W7(:,1),paraRes.H_W7(:,1))))]);
xlim([0 max(paraSet.H_W7length)+2]);
%xlabel('Length Upstream of Cylinder Centroid (m)')
ylabel('C_{drag}')
%title('Domain Height Verification')
legend('Re = 1.94e6', 'Re = 1.12e7','Location','southeast')
hold off
```

```
subplot(2,1,2)
ax12.line1 = plot(paraSet.H_W1length,paraRes.H_W1(:,2),'b*-');
hold on
ax12.line2 = plot(paraSet.H_W7length,paraRes.H_W7(:,2),'s-','
   Color',[0.8500 0.3250 0.0980]);
grid on
grid minor
ylim([-20    ceil(max(max(paraRes.H_W7(:,2),paraRes.H_W7(:,2)))
   /10)*10]);
xlim([0 max(paraSet.H_W7length)+2]);
xlabel('Vertical Domain Height (m)')
ylabel('% Relative Error')
%legend('W1','W7')
hold off

figure(2)
subplot(2,1,1)
ax21.line1 = plot(paraSet.USlength,paraRes.US_W1(:,1),'b*-');
hold on
ax21.line2 = plot(paraSet.USlength,paraRes.US_W7(:,1),'s-','Color
   ',[0.8500 0.3250 0.0980]);
grid on
grid minor
%ylim([0 ceil(max(max(paraRes.US_W1(:,1),paraRes.US_W7(:,1))))]);
ylim([0 4]);
xlim([0 max(paraSet.USlength)+2]);
%xlabel('Length Upstream of Cylinder Centroid (m)')
ylabel('C_{drag}')
%title('Upstream Verification')
legend('Re = 1.94e6', 'Re = 1.12e7','Location','southeast')
hold off

subplot(2,1,2)
ax22.line1 = plot(paraSet.USlength,paraRes.US_W1(:,2),'b*-');
hold on
ax22.line2 = plot(paraSet.USlength,paraRes.US_W7(:,2),'s-','Color
   ',[0.8500 0.3250 0.0980]);
grid on
grid minor
ylim([-20    ceil(max(max(paraRes.US_W1(:,2),paraRes.US_W7(:,2)))
   /10)*10]);
xlim([0 max(paraSet.USlength)+2]);
xlabel('Horizontal Domain - Length Upstream from Cylinder
   Centroid (m)')
ylabel('% Relative Error')
%legend('W1','W7')
hold off

figure(3)
```

```
subplot(2,1,1)
ax3.line1 = plot(paraSet.DSlength,paraRes.DS_W1(:,1),'b*-');
hold on
ax3.line2 = plot(paraSet.DSlength,paraRes.DS_W7(:,1),'s-','Color
    ',[0.8500 0.3250 0.0980]);
grid on
grid minor
%ylim([0 ceil(max(max(paraRes.DS_W1(:,1),paraRes.DS_W7(:,1))))]);
ylim([0 4]);
xlim([0 max(paraSet.DSlength)+2]);
%xlabel('Length Downstream of Cylinder Centroid (m)')
ylabel('C_{drag}')
%title('Downstream Verification')
legend('Re = 1.94e6', 'Re = 1.12e7','Location','southeast')
hold off

subplot(2,1,2)
ax3.line1 = plot(paraSet.DSlength,paraRes.DS_W1(:,2),'b*-');
hold on
ax3.line2 = plot(paraSet.DSlength,paraRes.DS_W7(:,2),'s-','Color
    ',[0.8500 0.3250 0.0980]);
grid on
grid minor
%ylim([-30    ceil(max(max(paraRes.DS_W1(:,2),paraRes.DS_W7(:,2)))
    /10)*10]);
ylim([-40 60]);
xlim([0 max(paraSet.DSlength)+2]);
xlabel('Horizontal Domain - Length Downstream from Cylinder
    Centroid (m)')
ylabel('% Relative Error')
%legend('W1','W7')
hold off
```

C.6 Repository for Miscellaneous Items and Other Scripts

This section includes other miscellaneous items and scripts. Included in this section is the definition files for .bashrc, .vimrc and .bash_aliases, as well as a variety of scripts that did not fit into the previous section. Each item will be described based on its purpose.

C.6.1 Home Folder

Code Listing C.58: .bashrc, is the global file to define different parameters in the terminal. User edits are towards the end of this file. This file is unique to each computer, as a result, any commands or aliases that were desired to be on all external computers were added to the .bash_aliases file. Commands or aliases that were only applicable to the main computer (Feynman), were left here in .bashrc.

```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package
   bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac

# don't put duplicate lines or lines starting with space in the
   history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in
   bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize

# If set, the pattern "**" used in a pathname expansion context
   will
# match all files and zero or more directories and subdirectories
   .
#shopt -s globstar
```

```
# make less more friendly for non-text input files, see lesspipe
    (1)
[ -x /usr/bin/lesspipe ] && eval "$(SHELL=/bin/sh lesspipe)"

# set variable identifying the chroot you work in (used in the
    prompt below)
if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
    debian_chroot=$(cat /etc/debian_chroot)
fi

# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
    xterm-color|*-256color) color_prompt=yes;;
    xterm) color_prompt=yes;;
esac

# uncomment for a colored prompt, if the terminal has the
    capability; turned
# off by default to not distract the user: the focus in a
    terminal window
# should be on the output of commands, not on the prompt
force_color_prompt=yes

if [ -n "$force_color_prompt" ]; then
    if [ -x /usr/bin/tput ] && tput setaf 1 >&/dev/null; then
        # We have color support; assume it's compliant with Ecma
            -48
        # (ISO/IEC-6429). (Lack of such support is extremely rare
            , and such
        # a case would tend to support setf rather than setaf.)
        color_prompt=yes
    else
        color_prompt=
    fi
fi

# ANSI color codes
RS="\[\033[0m\]"    # reset
HC="\[\033[1m\]"    # hicolor
UL="\[\033[4m\]"    # underline
INV="\[\033[7m\]"   # inverse background and foreground
FBLK="\[\033[30m\]" # foreground black
FRED="\[\033[31m\]" # foreground red
FGRN="\[\033[32m\]" # foreground green
FYEL="\[\033[33m\]" # foreground yellow
FBLE="\[\033[34m\]" # foreground blue
FMAG="\[\033[35m\]" # foreground magenta
FCYN="\[\033[36m\]" # foreground cyan
FWHT="\[\033[37m\]" # foreground white
```

```
BBLK="\[\033[40m\]" # background black
BRED="\[\033[41m\]" # background red
BGRN="\[\033[42m\]" # background green
BYEL="\[\033[43m\]" # background yellow
BBLE="\[\033[44m\]" # background blue
BMAG="\[\033[45m\]" # background magenta
BCYN="\[\033[46m\]" # background cyan
BWHT="\[\033[47m\]" # background white

if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h
       \[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt

# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*)
    PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]
       $PS1"
    ;;
*)
    ;;
esac

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)"
       || eval "$(dircolors -b)"
    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# colored GCC warnings and errors
#export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret
   =01;32:locus=01:quote=01'

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Add an "alert" alias for long running commands.  Use like so:
```

```
#    sleep 10; alert
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo
    terminal || echo error)" "$(history|tail -n1|sed -e '\''s/^\s
    *[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'

# Alias definitions.
# You may want to put all your additions into a separate file
    like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to
    enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/
    profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

cpallNewCaseSetup() {
    cp /home/dksan/OpenFOAM/myWork/global-base-case/
       allNewCaseSetup.sh .;
    bash allNewCaseSetup.sh;
    cat $basecaseacase2/allItemsToRevisit;
}
cpallNewCaseSetupONLY() {
    cp /home/dksan/OpenFOAM/myWork/global-base-case/
       allNewCaseSetup.sh .;
    bash allNewCaseSetup.sh;
}
schown() { sudo chown -R dksan:dksan $1; ls -l
}
lsgeometry() {
    ls | grep "$1m-height.*$2m-upstream._$3m-downstream.fms\|
       $1$1m-height.*$2m-upstream._$3m-downstream.fms\|$1m-
       height.*$2$2m-upstream._$3m-downstream.fms\|$1m-height.*
       $2m-upstream._$3$3m-downstream.fms"
}
lssims() {
    ls | grep "$1m-height.*$2m-upstream._$3m-downstream\|$1$1m-
       height.*$2m-upstream._$3m-downstream\|$1m-height.*$2$2m-
```

```
        upstream.*_$3m-downstream\|$1m-height.*$2m-upstream.*
        _$3$3m-downstream"
}

today=~/OpenFOAM/myWork/2022-1207a*
storagetoday=/storage/dksan/2022-1203f*
export today

storage=/storage/dksan
storagegeometry=/storage/dksan/zBackup/global-geometry-
    notUsedRecently
storagekasun=/storage/kasun93
myWork0=~/OpenFOAM/myWork/0*
myWork1=~/OpenFOAM/myWork/1*
myWork3=~/OpenFOAM/myWork/3*
basecase=~/OpenFOAM/myWork/global-base-case
basecaseanalysis=~/OpenFOAM/myWork/global-base-case/analysis
basecaseacase2=~/OpenFOAM/myWork/global-base-case/analysis/case2*
basecaseagcm=~/OpenFOAM/myWork/global-base-case/analysis/gcm*
basecasecfmesh=~/OpenFOAM/myWork/global-base-case/cfmesh/202*
basecasecgeometryfiles=~/OpenFOAM/myWork/global-base-case/cfmesh/
    geometry/files*
basecasematlab=~/OpenFOAM/myWork/global-base-case/matlab/202*
basecasedevelop=~/OpenFOAM/myWork/global-base-case/developMap
    /202*
basecasedmatlab=~/OpenFOAM/myWork/global-base-case/developMap/
    matlab/202*
extut=~/OpenFOAM/myWork/ex-tut/
extutfunction=~/OpenFOAM/myWork/ex-tut/of9/test/postProcessing/
    channel/system

kasunglobal=~/OpenFOAM/myWork/global-base-case/analysis/kasun/
naskasun=/nas/kasun93
```

Home Folder

Code Listing C.59: .bash_aliases, referenced by .bashrc. Contains aliases, commands, and a custom PS1 which is exported to every computer that is used to run simulations.

```
# ~/.bash_aliases: a central location for storing aliases so that
    they can be sourced by other scripts, exported to other
   working computers, etc

alias matlab='~/Misc-Software/MATLAB/installed/bin/matlab'

# User specified aliases and functions
alias python=python3

## \/ Source multiple OF Installations and input onto bash PS1
   Prompt
OFVERSION=na #Default OF Version name if none is given via an
   alias
#alias of5='source $HOME/OpenFOAM/OpenFOAM-5/etc/bashrc &&
   OFVERSION=5'
alias of6='source $HOME/OpenFOAM/OpenFOAM-6/etc/bashrc &&
   OFVERSION=6'
alias of7='source $HOME/OpenFOAM/OpenFOAM-7/etc/bashrc &&
   OFVERSION=7'
alias of9='source $HOME/OpenFOAM/OpenFOAM-9/etc/bashrc &&
   OFVERSION=9'
alias of2112='source $HOME/OpenFOAM/OpenFOAM-v2112/etc/bashrc &&
   OFVERSION=2112'
export OFVERSION

alias salome9-3='~/Salome/SALOME-9.3.0-UB18.04-SRC/salome'
alias diff='colordiff -r --exclude=*~'
alias sourcebashrc='source ~/.bashrc'
alias vimbashrc='vim ~/.bashrc'
alias vimbashrcaliases='vim ~/.bash_aliases'
alias who='who | sort'
alias ncdu='ncdu --exclude "csu-OneDrive" --exclude "/nas"'

#Misc command aliases/functions
alias lsfms='ls | grep fms'
alias tree2='tree -L 2'
alias taillog='tail ---disable-inotify -f -s 1 log.*'
alias tailallRun='tail ---disable-inotify -f -s 1 log.
   helperRunningMPI log.allRun'
alias tialallRun='tailallRun'
alias tailT='tail ---disable-inotify -f -s 1 log.* | grep "
   ClockTime\|deltaT\|Time = \|Courant\|reading\|Reconstructing
   \|^[[:space:]]*$"'
alias grepRA2B2='grep -R -A 2 -B 2'
```

```
alias puttyKill='pkill -9 -t'
alias ramClear1="sudo sh -c 'echo 1 > /proc/sys/vm/drop_caches' "
alias gg="gnuplot gnu*"
alias ggforce="gnuplot gnu-forceCoeffs"
alias numahardware="numactl --hardware"
nbashallMaster () { nohup bash allMasterCleanRunNotifyClean.sh $1
    & }
nbashallMastersleep () { date; echo "Sleeping for $1 seconds";
   sleep $1; date; nohup bash allMasterCleanRunNotifyClean.sh $2
   & }
bashstatus() { bash ~/status.sh 1 userd $1 | tee ~/log.status; }

ffmpegpngTomp4()    { sudo echo; #bash ~/Desktop/mount-csu-
   onedrive.sh;
    #Use ffmpeg to make the series of images into a mp4. $1 =
       framerate. $2 is the name of the .pngs. $3 is an optional
       naming argument. Resulting mp4 is named based on the
       existing directory name
    ffmpeg -framerate $1 -i $2%04d.png -c:v libx264 -strict -2 -
       preset slow -pix_fmt yuv420p -vf "scale=trunc(iw/2)*2:
       trunc(ih/2)*2" -f mp4 $(basename $(dirname $(pwd)))_$3$1.
       mp4;
    echo; echo "mp4 created";
    mv $(basename $(dirname $(pwd)))_* $csuOneDriveanimations; #
       Move created mp4 to dedicated folder on csuOneDrive
    echo "mp4 uploading to CSU OneDrive."; sleep 2
    echo "mp4 uploading to CSU OneDrive.."; sleep 4
    echo "mp4 uploading to CSU OneDrive..."; sleep 4
    echo "mp4 moved to CSU OneDrive"; #echo; echo "sudo to
       disconnect CSU OneDrive";
}
alias dfMain='df -h | grep "/sda\|/md0\|karan\|Filesystem"'
alias duHere='du -sh .' #Get size of present directory
alias parafoamfoam='/home/dksan/OpenFOAM/Paraview_v5-11-1/bin/
   paraview foam.foam'
alias parafoamfoam591='/home/dksan/OpenFOAM/Paraview_v5-9-1/bin/
   paraview foam.foam'
alias catallItemsToRevisit='cat $basecaseacase2/allItemsToRevisit
   '
alias ns='vim cfmesh/system/meshDict analysis/globalMaster
   analysis/system/controlDict'
alias nsmeshDict='vim cfmesh/system/meshDict'

nsbashSetup() { bash allSetupSettings.sh $1 | tee log.
   allSetupSettings
}
alias nsglobalMaster='vim analysis/globalMaster'
alias nscontrolDict='vim analysis/system/controlDict'

nas=/nas/dksan
```

```
nasstorage=/nas/dksan/storage_bckp
nasexternaltransfers=/nas/dksan/external_transfers
csuOneDrive=~/csu-OneDrive/00_MS
csuOneDrivecompleted=~/csu-OneDrive/_MovingFiles/
    feynman_transfers/_completed/
csuOneDrivefeynman=~/csu-OneDrive/_MovingFiles/feynman_transfers/
csuOneDriveanimations=~/csu-OneDrive/00_MS/02_CFD-modeling/01
    _Figures-Animations/
myWork=~/OpenFOAM/myWork
global=~/OpenFOAM/myWork/global
globalfunctions=~/OpenFOAM/myWork/global/functions
globalgeometry=~/OpenFOAM/myWork/global/geometry
global100m_results=~/OpenFOAM/myWork/global/developedFlow/100*
globalsorted=~/OpenFOAM/myWork/global/developedFlow/
    sortedInletData

#pg 146 LCL - a master prompt for a revised command line
PS1="\[\033[s\033[0;70H\033[0;41m\033[K\033[1;33m\t\033[0m\033[u
    \]\[\033[1;33m\]$USER-$HOSTNAME-\$OFVERSION \w\$\[\033[0m\] "
     ## Output = USER-HOSTNAME-ofVersion. On Feynman: dksan-
    feynman-of9
export PS1
```

Home Folder

Code Listing C.60: ,.vimrc, default file modified by author to customize VIM functionality. See previous discussion on VIM for details on some of these modifications.

```
execute pathogen#infect()

set nocompatible              " be iMproved, required
filetype off                  " required

" set the runtime path to include Vundle and initialize
set rtp+=~/.vim/bundle/Vundle.vim
call vundle#begin()
" alternatively, pass a path where Vundle should install plugins
"call vundle#begin('~/some/path/here')

" let Vundle manage Vundle, required
Plugin 'VundleVim/Vundle.vim'

Plugin 'NLKNguyen/papercolor-theme'
"Plugin 'ycm-core/YouCompleteMe'
"NeoBundle 'rdnetto/YCM-Generator'

" All of your Plugins must be added before the following line
call vundle#end()            " required
"filetype plugin indent on    " required



" An example for a vimrc file.
"
" Maintainer:   Bram Moolenaar <Bram@vim.org>
" Last change:  2019 Jan 26
"
" To use it, copy it to
"     for Unix and OS/2:  ~/.vimrc
"             for Amiga:  s:.vimrc
"  for MS-DOS and Win32:  $VIM\_vimrc
"           for OpenVMS:  sys$login:.vimrc

" When started as "evim", evim.vim will already have done these
   settings, bail
" out.
if v:progname =~? "evim"
  finish
endif

" Get the defaults that most users want.
source $VIMRUNTIME/defaults.vim
```

```
if has("vms")
  set nobackup          " do not keep a backup file, use versions
      instead
else
  set backup            " keep a backup file (restore to previous
      version)
  if has('persistent_undo')
    set undofile        " keep an undo file (undo changes after
       closing)
  endif
endif

if &t_Co > 2 || has("gui_running")
  " Switch on highlighting the last used search pattern.
  set hlsearch
endif

" Put these in an autocmd group, so that we can delete them
   easily.
augroup vimrcEx
  au!

  " For all text files set 'textwidth' to 78 characters.
  autocmd FileType text setlocal textwidth=78
augroup END

" Add optional packages.
"
" The matchit plugin makes the % command work better, but it is
   not backwards
" compatible.
" The ! means the package won't be loaded right away but when
   plugins are
" loaded during initialization.
if has('syntax') && has('eval')
  packadd! matchit
endif

"DKS ADDED 4/11/2022
filetype plugin indent on
" show existing tab with 4 spaces width

set tabstop=4
" when indenting with '>', use 4 spaces width

set shiftwidth=4
" On pressing tab, insert 4 spaces

set expandtab
```

```
"set foam256_use_custom_colors=0
" colorscheme evening
set background=dark
set t_Co=256
colorscheme PaperColor
syntax on

set mouse=a

"Explicitly defined this to allow for copying of more lines (up
   to 1000 with
"<1000)
set viminfo='100,<10000,s1000,h

"Write the current date/time to the bottom of a vim windown when
   :w
"augroup SAVING
"    autocmd!
"    autocmd BufWritePost * echo strftime('%c')
"augroup END

set nobackup "Does not automatically write out a *~ file after
   editing every file

set clipboard=unnamedplus
"Automatically executes '"+ so that '"+y = y. "+ for copying to
   clipboard buffer
```

Home Folder

Code Listing C.61: status.sh, this script fires an email to the user with a summary of the simulation(s) status, estimated time of completion, number of processors being used, available storage, and number of time steps saved. This script is called from several locations including crontab, .bash_aliases and some scripts in the analysis folder.

```
#!/bin/bash

#IF THEN to allow for printf statements to be skipped IF script
   is being run from crontab.
#Purpose: To avoid an email error due to echoed statements that
   are not meaningful to crontab
#if [[ "$3" == "crontab" ]]; then
#    printf() { :; } #Printf will not be recognized unless this
   statement is commented
#fi

whoValue=$(who | grep -m 1 pts)

if [ ! -z "$whoValue" ] || ([ "$(date +%M)" == 30 ] && [ $(( $(
   echo $(date +%H) | bc) %2)) -eq 0 ]) || [ "$2" == "notif" ] ||
    [ "$2" == "alert" ] || [ "$2" == "userd" ] || [[ "$2" == *sU*
     ]]
then #I am EITHER: 1) Logged in, 2) it is an EVEN Hour and 30
   minutes, 3) crontab has initiated an alert signal, 4) the user
    has requested an update. Therefore, continue this script
    :
else #The above options are false. Do not continue the script
    exit 1
fi

cpuUsage=$(top -bn2 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%*
   id.*/\1/" | awk '{print 100 - $1"%"}' | tail -1)    #Calculate
    average cpuUsage in percent
cpuUse=$(echo $cpuUsage | sed 's/%$//')

   #Edit cpuUsage value to remove percent sign for future math
   manipulation
cpuProcessorsUsed=$(echo "scale=2;($cpuUse/100)*$(cat ~/procnum)"
    | bc)
                                                                #
   Calculate the approx num of processors being used
cpuProcessorsAvai=$(echo "scale=2;$(cat ~/procnum)-
   $cpuProcessorsUsed" | bc)
                                                              #
   Calculate the approx num of processors available

#Get available storage
```

```
dfSize=$(df / --output='pcent' | grep -o "[0-9]*")
dfSizeGBAvailable=$(echo "scale=2;(1-($dfSize/100))*$(cat ~/
   dfsize)" | bc)

#Get information on Simulation Stats
simPID=$(pgrep -u `whoami` mpirun)
printf "%s" "Current Time MST: $(date)"
if [ -z "$simPID" ]
then
    printf "%s\n" "  > No MPI simulation running"
else
    printf "%s\n" "  > MPI simulation running"
    simFolders=$(pwdx $simPID | sed 's/^[0-9]\+: //g') #Get
       working directory and remove PID values
    #Save base folder name to an array to be run through as a
       loop in the section below
    mapfile -t simNameArray < <( basename -a $(dirname
       $simFolders) )
fi

j=0;
BEGtrack=0;

#Process each running MPI simulation and do math
for i in "${simNameArray[@]}"; do
    numa=$(cat ~/OpenFOAM/myWork/$(echo "$i")/analysis/log.allRun
                        | grep Numa | cut -c-6)
    printf "\n%s\n" "------- $numa: Simulation $i Stats -------"

    #IMPORT timeVarsEND
    #From log.pimpleFoam at the end of the file, pull the latest
       values for TIME and ClockTime
    timeVarsEND=$(tac ~/OpenFOAM/myWork/$(echo "$i")/analysis/log
       .pimpleFoam           | sed -n '/Exec/,$p' |  sed -n '/^
       Time/,$p' | tac | tail -n 5)
    varClockTime=$(echo "$timeVarsEND"              | sed -n -e 's
       /^.*ClockTime = //p'  | sed 's/ s//')
    vardeltaT=$(echo "$timeVarsEND"                 | sed -n -e 's
       /^.*deltaT = //p'     | sed 's/ s//')
    varSimTime=$(echo "$timeVarsEND"                | sed -n -e 's
       /^.*Time = //p'       | sed 's/ s//' | tail -1 | awk -F"E"
        'BEGIN{OFMT="%10.14f"} {print $1 * (10 ^ $2)}' | sed 's
       /0*$//')

    writeInt=$(cat ~/OpenFOAM/myWork/$(echo "$i")/analysis/system
       /controlDict | grep writeInt | sed 's/;.*/;/')
    numOfIterations=$(cat ~/OpenFOAM/myWork/$(echo "$i")/analysis
       /log.pimpleFoam | grep ExecutionTime | wc -l)
    savedTimeSteps=$(echo "scale=0;$(ls ~/OpenFOAM/myWork/$(echo
       "$i")/analysis/processor1 | wc -l)-2" | bc)
```

```
purgeWrite=$(cat ~/OpenFOAM/myWork/$(echo "$i")/analysis/
   system/controlDict | grep purgeWrite | sed 's/purgeWrite
       //' | sed 's/;.*//')
BEGtrack=$(echo "scale=0;$BEGtrack+1" | bc)
if [[ "$3" == "" ]]
then
    #- Subtract XX iterations and round down the value to the
       next lowest 200th value until the value is greater
      than $numLimit
    numLimit=48700 #7500
    numTest=$(awk '{printf "%d00\n", $0 / 100}' <<< $(echo "
      scale=0;$numOfIterations-1000" | bc))
    #numTest=$(awk '{printf "%d000\n", $0 / 1000}' <<< $(echo
       "scale=0;$numOfIterations-1000" | bc))
    if [ $numTest -lt $numLimit ]; then
        timeVarsBEGdelay=$numTest
        BEGtrack=$(echo "scale=0;$BEGtrack-1" | bc)
    else
        timeVarsBEGdelay=$numLimit; #For air: 5000 is enough
           for uniform flow regimes. 10000 is required for
           developed flow regimes
    fi
else
    timeVarsBEGdelay=$3;
    BEGtrack=$(echo "scale=0;$BEGtrack-1" | bc)
fi

#IMPORT timeVarsBEG.
#Purpose: Import timeVars once simulation is in steady state
   to create an accurate time of completion estimate from the
    beginning
        timeVarsBEG=$(cat ~/OpenFOAM/myWork/$(echo "$i")/
           analysis/log.pimpleFoam | awk '/^ExecutionTime/,/^
           Time/{if(++m==1)n++;if(n==$timeVarsBEGdelay')
           print;if(/^Time/)m=0}')
        varClockTimeBEG=$(echo "$timeVarsBEG"          | sed
           -n -e 's/^.*ClockTime = //p' | sed 's/ s//')
        vardeltaTBEG=$(echo "$timeVarsBEG"             | sed
           -n -e 's/^.*deltaT = //p'    | sed 's/ s//')
        varSimTimeBEG=$(echo "$timeVarsBEG"            | sed
           -n -e 's/^.*Time = //p'      | sed 's/ s//' |
           tail -1 | sed -E 's/([+-]?[0-9.]+)[eE]\+?(-?)
           ([0-9]+)/(\1*10^\2\3)/g')

if [[ "$varClockTimeBEG" == "" ]] #If $timeVarsBEGdelay is
   larger than the size of the log file, then reset these
   variables to 0 to allow the calculation to still proceed
then
    varClockTimeBEG=0;
    varSimTimeBEG=0;
```

```
    printf "%s\n" "At $numOfIterations iterations  ~~
       $savedTimeSteps time steps currently saved out of
       $purgeWrite ~~"
else
    printf "%s\n" "** $timeVarsBEGdelay iterations exceeded
       **   >> Currently at $numOfIterations iterations <<
       ~~ $savedTimeSteps time steps currently saved out of
       $purgeWrite ~~"
fi

#IMPORT controlDictVars
controlDictVars=$(tac ~/OpenFOAM/myWork/$(echo "$i")/analysis
   /system/controlDict  | awk '!flag; /endTime/{flag = 1};'
          | awk '/writeInterval/{flag = 1}; flag' | tac)
varendTime=$(echo "$controlDictVars"        | sed -n -e 's
   /^.*endTime //p'      | sed 's/ //g' | sed 's/;.*//' )
varwriteInterval=$(echo "$controlDictVars"  | sed -n -e 's
   /^.*writeInterval //p'| sed 's/ //g' | sed 's/;//')

#Do some math - see spreadsheet
#Following calculations assume starting simulation time is
   timeVarsBEG seconds
calcEstHrsForOneSimulatedSec=$(         echo "scale=3; (
   $varClockTime-$varClockTimeBEG)/3600/($varSimTime-
   $varSimTimeBEG)"| bc | awk '{printf "%.4f\n",$1}')
calcEstTotalSimulationTime_Hrs=$(       echo "scale=2; (
   $calcEstHrsForOneSimulatedSec*$varendTime)"
                        | bc | awk '{printf "%.2f\n",$1}')
calcEstTotalSimulationTime_Days=$(      echo "scale=2; (
   $calcEstHrsForOneSimulatedSec*$varendTime)/24"
                      | bc)
calcEstRemainingSimulationTime_Hrs=$(   echo "scale=2;((
   $varendTime-$varSimTime)*$calcEstHrsForOneSimulatedSec)"
          | bc | awk '{printf "%.2f\n",$1}')
calcEstRemainingSimulationTime_Days=$(  echo "scale=2; (
   $varendTime-$varSimTime)*$calcEstHrsForOneSimulatedSec/24
   "       | bc)

#Calculate Estimated Completion Time
currentTime=$(date +%H:%M)
currentTimeVar=$(date -d $currentTime +%s)
calcEstCompletedTime=$( echo "scale=0;$currentTimeVar + (
   $calcEstRemainingSimulationTime_Hrs*3600)" | bc)
estCompletedTime=$(                     date --date=
   @$calcEstCompletedTime +"%a %b %e, %I:%M %p %Z %Y")

printf "%s\n" "< deltaTBEG      = $vardeltaTBEG sec   |
   SimTimeBEG = $varSimTimeBEG sec"
printf "%s\n" "< deltaTNOW      = $vardeltaT sec      |
   $writeInt"
```

```
    printf "%s\n" "< CPU ClockTime  = $varClockTime sec  |
       Simulated Time = $varSimTime sec   |    >> Finishing at
       $varendTime sec <<"
    printf "%s\n" "< Generate 1 simulated sec (hrs):
                        $calcEstHrsForOneSimulatedSec"
    printf "%s\n" "< Est Total Simulation Time (hrs/days):
                $calcEstTotalSimulationTime_Hrs  /
       $calcEstTotalSimulationTime_Days"
    printf "%s\n" "< Est Remaining Simulation Time (hrs/days):
       $calcEstRemainingSimulationTime_Hrs  /
       $calcEstRemainingSimulationTime_Days"
    printf "%s\n" "< Est Completion Time MST:
                                $estCompletedTime"
    j=$(echo "scale=0;$j+1" | bc)
done

if [ $dfSize -gt $1 ] #Check if storage capacity is less than
   notification threshold
then
echo -e ">>> Simulations Running <<<
$(for i in "${simNameArray[@]}"; do echo "$i"; done)

>>> Detailed Simulation Stats <<<    Time calc removing first
   $timeVarsBEGdelay iterations
$(echo -e "$(cat ~/log.status)")

>>> Storage Stats <<<
Percentage USED: $dfSize%
Space Available: ~$dfSizeGBAvailable GB

>>> CPU Stats <<<
Average CPU Load: $cpuUsage
Num of Processors Used:      $cpuProcessorsUsed
Num of Processors Available: $cpuProcessorsAvai

" | mail -s "$(cat ~/mailname) $BEGtrack $2 - $1% topped @
   $dfSize% | Proc >> $cpuProcessorsAvai Free, $cpuProcessorsUsed
    Used" This-Is-Your-Email@DomainName.com
fi
```

Code Listing C.62: crontab, access via "crontab -e". A task scheduler, used in this case to automatically fire status.sh. The intervals given below dictate a higher hard drive capacity resulting in more frequent emails to the user, hopefully to alert them to a storage issue that must be remedied.

```
# m h  dom mon dow   command
*/30 * * * * bash /home/dksan/status.sh 1 status > log.
   status
*/2 * * * * bash /home/dksan/status.sh 98 alert > log.
   status
*/5 * * * * bash /home/dksan/status.sh 95 alert > log.
   status
*/20 * * * * bash /home/dksan/status.sh 90 alert > log.
   status
30 1,3,5,7,15,17,19,21,23 * * * bash /home/dksan/status.
   sh 24 notif > log.status
```

Home Folder

Code Listing C.63: notification.sh, a script that is run at the conclusion of a simulation (see helperMaster.sh). This script alerts the user of the simulation completion, as well as a general summary of the log file results to help with checking the simulation via email.

```
#!/bin/bash

echo "--- RUNNING notification.sh ---"

#Debugging information and gathering time stamps
completedTime=$(date)       ; #echo "debug: completedTime = "
   $completedTime

#Save the latest run's log information which will be included in
   the email to be sent to me
time=$(cat $todayFolder/log.allRun | grep 'mpirun RunTime\|
   Simulation RunTime')
logs=$( tail -n 40 $todayFolder/log.* )
#numa=$(cat ~/OpenFOAM/myWork/$todayFolder/analysis/log.allRun
                    | grep Numa | cut -c-6)

#Send email to user
echo -e "Simulation Folder: $todayFolder \nOFv-$OFVERSION.
   Completed $completedTime. \n\nComputational Time$time\n\nlogs
   \n$logs"  |
    mail -s "$(cat ~/mailname) $(cat $(echo "$todayFolder")/log.
       allRun  | grep Numa | cut -c 6). Simulation Completed -
       $todaySimulationName" This-Is-Your-Email@DomainName.com
```

C.6.2 Desktop Folder

Code Listing C.64: mount-file-server.sh, to mount the external file server.

```
#!/bin/sh

umount /nas
mount -t cifs //DOMAIN-NAME/people /nas -o username=USERNAME -o
    dom=engr_dom
```

Code Listing C.65: mount-csu-onedrive.sh, mounts OneDrive folder. Requires setting up a rclone remote.

```
#!/bin/sh

rclone --vfs-cache-mode writes mount "OneDrive_CSU": ~/csu-
    OneDrive/ &
```

Code Listing C.66: umount-csu-onedrive.sh, unmounts OneDrive folder.

```
#!/bin/sh

sudo umount ~/csu-OneDrive
```

Desktop Folder

Code Listing C.67: allMountDrives.sh, strings together previous 2 mounting scripts and testing methods to expedite remounting locations after restarting system.

```
#!/bin/sh

echo "--- RUNNING allMountDrives.sh ---"

echo "> Mounting CSU OneDrive"
bash mount-csu-onedrive.sh
echo "> CSU OneDrive Mounted"

echo "> Mounting NAS File Server"
sudo bash mount-file-server.sh
echo "> NAS File Server Mounted"

echo
echo "> Testing mounted file connections:"
echo "> cd /nas/dksan/; ls"
cd /nas/dksan
ls

echo
echo "> cd ~/csu-OneDrive/00_MS/; ls"
cd ~/csu-OneDrive/00_MS
ls

echo
echo "> Mounting test complete"
echo

echo "--- allMountDrives.sh FINISHED ---"
```

C.6.3 MyWork Folder

Code Listing C.68: allSimImport-fromNAStoStorage.sh, for simulations that were run on other computers, this script transfers the data that was sent to NAS from the original processing computer to the internal storage harddrive for additional data integrity.

```
#!/bin/bash

# Run this script from FEYNMAN $myWork
# This script automatically copies the $1 folder from NAS to /
   storage for dual backup location
# Data on original local computer can now be deleted
echo "--- RUNNING allSimImport-fromNAStoStorage.sh ---"
echo "> File size of entire working directory is $(du -sh /nas/
   dksan/external_transfers/$1)"

#Backup from /nas to /storage
echo;echo "> Backing up $1 from /nas to /storage"
sudo rsync -a --info=progress2  /nas/dksan/external_transfers/$1
      /storage/dksan/
sudo mv                         /nas/dksan/external_transfers/$1
      /nas/dksan/storage_bckp/
echo ">  Backup of $1 from /nas to /storage COMPLETE"; echo

#Copying other typical files to /storage
sudo rsync -a /home/dksan/OpenFOAM/myWork/global-base-case/
   allMatlabSetup.sh    /storage/dksan/$1/
sudo rsync -a /home/dksan/OpenFOAM/myWork/global-base-case/
   allStatsStorage.sh   /storage/dksan/$1/

#Reset permissions on /storage version
sudo chown -R dksan:dksan /storage/dksan/$1

#Move folder from $myWork/1* to simComplete folder for future
   deletion
echo "> Moving original data to myWork/1_*/simComplete folder to
   be deleted"
mv /home/dksan/OpenFOAM/myWork/1_exportedToOneDrive/$1  /home/
   dksan/OpenFOAM/myWork/1_exportedToOneDrive/simComplete-
   dataOnNASandStorage/

echo; echo "--- allSimImport-fromNAStoStorage.sh FINISHED ---"
```

MyWork Folder

Code Listing C.69: allSimExport-fromFeynmanToOneDrive.sh, exports a simulation and associated files from Feynman to OneDrive. This simulation will be downloaded via OneDrive by using allSimExport-fromOneDriveToLocal.sh

```
#!/bin/bash

# Run this script from FEYNMAN $myWork
# This script automatically copies the $1 folder to OneDrive and
   updates associated support files if needed.
echo "--- RUNNING allSimExport-fromFeynmanToOneDrive.sh ---"

rm -rf                      $1/cfmesh/1 $1/cfmesh/constant
echo "> File size of entire working directory is $(du -sh $1)";
   echo

#Copy $1 folder to OneDrive/feynman folder
echo "> Copying $1 to ~/csu-OneDrive/_MovingFiles/
   feynman_transfers/"
rsync -a --delete --info=progress2 $1 ~/csu-OneDrive/_MovingFiles
   /feynman_transfers/

echo; echo "> Verifying transfer is complete with rsync -av --
   delete command on $1"
sleep 2
rsync -av --delete          $1 ~/csu-OneDrive/_MovingFiles/
   feynman_transfers/

echo; echo "> Moving $1 to myWork/exportedToOneDrive"
mv $1 1_exportedToOneDrive/

#Update OneDrive/global to match feynman global folder
echo; echo "> Updating OneDrive/global folder to match Feynman/
   global contents"
rsync -av --delete --exclude 'developedFlow' global /home/dksan/
   csu-OneDrive/_MovingFiles/feynman_transfers/
#rsync -av --delete global /home/dksan/csu-OneDrive/_MovingFiles/
   feynman_transfers/

#Update other files on OneDrive
echo; echo "> Updating other files on OneDrive"
rsync -av --delete ~/.bash_aliases      /home/dksan/csu-OneDrive/
   _MovingFiles/feynman_transfers/ | grep -v 'sent\|total size'
rsync -av --delete ~/.vimrc             /home/dksan/csu-OneDrive/
   _MovingFiles/feynman_transfers/ | grep -v 'sent\|total size\|
   increm'
```

```
rsync -av --delete ~/status.sh          /home/dksan/csu-OneDrive/
    _MovingFiles/feynman_transfers/ | grep -v 'sent\|total size\|
    increm'
rsync -av --delete ~/notification.sh    /home/dksan/csu-OneDrive/
    _MovingFiles/feynman_transfers/ | grep -v 'sent\|total size\|
    increm'

echo; echo "--- allSimExport-fromFeynmanToOneDrive.sh FINISHED
    ---"
```

C.6.4 OneDrive Folder

Code Listing C.70: allSimExport-fromOneDriveToLocal.sh, this script transfers the new simulation on OneDrive to a different computer.

```
#!/bin/bash

# Run this script from LOCAL computer inside OneDrive/
   feynman_transfers
# This script automatically copies the $1 and global folders to
   the LOCAL computer
echo "--- RUNNING allSimExport-fromOneDriveToLocal.sh ---"
echo "> File size of entire working directory is $(du -sh $1)"

#Copy $1 folder
echo "> Copying $1 to Local Computer"
rsync -a --info=progress2 $1                          $HOME/OpenFOAM/
   myWork/

echo; echo "> Verifying transfer is complete with rsync -av --
   delete command on $1"
sleep 2
rsync -av --delete        $1                          $HOME/OpenFOAM/
   myWork/

#echo; echo "> Moving $1 to _transferred-to-local/$HOSTNAME"
#mv  $1  ~/csu-OneDrive/_MovingFiles/feynman_transfers/
   _transferred-to-local/$HOSTNAME/

#rsync -av allExportSimTo_completedOneDrive.sh   $HOME/OpenFOAM/
   myWork/

#Update global to match feynman global folder
echo; echo "> Updating global folder to match Feynman/global
   contents"
rsync -av --delete global/                          $HOME/OpenFOAM/
   myWork/global

#Update OneDrive .bash_aliases version
echo; echo "> Updating files in ~/"
rsync -av --delete .bash_aliases                     ~/ | grep -v '
   sent\|total size'
rsync -av --delete .vimrc                            ~/ | grep -v '
   sent\|total size\|increm'
rsync -av --delete status.sh                         ~/ | grep -v '
   sent\|total size\|increm'
rsync -av --delete notification.sh                   ~/ | grep -v '
   sent\|total size\|increm'
```

```
rsync -av --delete allSimExport-fromLocalFinalToNAS.sh    $HOME/
   OpenFOAM/myWork/ | grep -v 'sent\|total size\|increm'


echo " > Exporting $1 to LOCAL computer COMPLETE"; echo

echo "!! >> User must remove files from OneDrive. Consider using
   GUI application in 1-2 days to prevent errors. << !!"; echo

echo "--- allSimExport-fromOneDriveToLocal.sh FINISHED ---"
```

OneDrive Folder

Code Listing C.71: allSimExport-fromLocalFinalToNAS.sh, this script transfers a completed simulation on an external computer to the NAS server.

```
#!/bin/bash

# Run this script from LOCAL computer inside $myWork
# This script automatically copies the completed $1 folder from
   the LOCAL computer to NAS
echo "--- RUNNING allSimExport-fromLocalFinalToNAS.sh ---"
echo "> File size of entire working directory is $(du -sh $1)"

#Save current working directory
folderName=$1
sudo mkdir  /nas/dksan/external_transfers/$folderName
sudo mkdir  /nas/dksan/external_transfers/$folderName/cfmesh
sudo mkdir  /nas/dksan/external_transfers/$folderName/0-figures

#Copy $1 folder to NAS
echo; echo "> Copying $1 to /nas/dksan/external_transfers/"
sudo rsync -a --info=progress2   $1/analysis /nas/dksan/
   external_transfers/$1/
echo;
sudo rsync -a $1/cfmesh/system/meshDict $1/cfmesh/log* /nas/dksan
   /external_transfers/$folderName/cfmesh
#sudo rsync -a $1/cfmesh/*.fms $1/cfmesh/system/meshDict $1/
   cfmesh/log* $1/cfmesh/*.hdf  /nas/dksan/external_transfers/
   $folderName/cfmesh
sudo rsync -a $1/global_values.m  $1/log.*
                                                  /nas/dksan/
   external_transfers/$folderName/
#sudo rsync -a /home/dksan/OpenFOAM/myWork/global-base-case/
   allMatlabSetup.sh     /storage/dksan/$folderName/
#sudo rsync -a /home/dksan/OpenFOAM/myWork/global-base-case/
   allStatsStorage.sh    /storage/dksan/$folderName/


echo; echo "> Moving local folder to exportedNAS/"
mv $1 exportedNAS/

echo; echo "--- allSimExport-fromLocalFinalNAS.sh FINISHED ---"
```

C.7 Writing Recommendations

Writing in general takes a considerable amount of time. The complicated factor of writing a thesis is that you are synthesizing information that you have thought about, generated and processed over the last 1-3 years. Because of this, the author found it helpful to take the advise of my advisor, which was to consistently work on writing for a 1-4 hours per week, even as early as the literature review stage. The author followed this advise early on, and it resulted into rough outlines of the final chapters for the thesis, and a brain dump of some of the literature review content that was later used in the final thesis.