Urban Wake Field Generation Using LES for Application to Quadrotor Flight

by

Mark Sutherland

A Thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfilment of

the requirements for the degree of

Master of Applied Science

in

Aerospace Engineering

Department of Mechanical and Aerospace Engineering

Carleton University

Ottawa, Ontario, Canada

April 2015

Abstract

A method is presented for using LES to generate urban wake velocity fields for use in studying wake effects on autonomous quadrotor's flight performance. The wake velocities are stored in a database and accessed by a MATLAB/Simulink flight simulator based on a custom quadrotor platform. This simulator and database is used to study the difference in flight performance between wake fields generated by RANS methods and LES with five flight missions. Results of holding position in a constant freestream show both CFD methods produce similar results and can hold position in all three directions within approximately ± 1.5 body lengths. When the quadrotor is in or on the boundary of the building wake the maximum deviation volumes, as calculated when using a RANS or LES background wind, can differ by 3 orders of magnitude. This is a result of the resolved turbulent fluctuations in the LES wake field causing a greater degree of non-isotropic flow in comparison to RANS. Furthermore, the LES wake field can cause skewed deviations by as much as 5 to 1 in a given direction for both holding position and moving along a desired flight path. Since the LES wake database more accurately reflects the wake fields present behind real world structures, using a wake field replicated by a RANS simulation will significantly over estimate the performance for position hold or slow moving flight paths for multirotor UAVs on the order of 0.5m in size and 2kg in mass.

Acknowledgments

Without question I would like to thank my supervisor Professor Jason Etele of Carleton University for providing such a unique opportunity and unparalleled guidance. Little did I know your patience, trust, feedback, and humor would make this endeavor incredibly pleasant, dare I say, fun.

Similarly, thank you to Dr. Giovanni Fusina of Defense Research and Development Canada for the chance to work on such a project and the consequent large amount of leeway and freedom on it.

Certainly a resounding thank you to friends and family for your wonderful support leading up to and during this enterprise. Considering your the capital investment and committed review time you are almost as qualified and deserving as I am.

Finally to new and old friends alike in Ottawa, you have made my time here, and overall experience, just that much more enjoyable.

Thank you to everyone.

Table of Contents

List of Tables

List of Figures

List of Acronyms

Acronyms	Definition
ABL	Atmospheric Boundary Layer
AIJ	Agriculture Institute of Japan
CFD	Computation fluid dynamics
CG	Center of gravity
CLF	Courant-Friedrichs-Lewy
ESC	Electronic Speed Controller
FAA	Federal Aviation Authority
GCS	Ground Control Station
GPS	Global Positioning System
GUI	Graphical User Interface
LES	Large-Eddy Simulation
LiDAR	Light Direction And Ranging
MUA	Micro Unmanned Aircraft
NED	North-East-Down

OEE	One-Equation Eddy
PID	Proportional-Integral-Derivative
PISO	Pressure Implicit with Splitting of Operators
PWM	Pulse Width Modulation
RANS	Reynolds-Averaged Navier-Stokes
RPA	Remotely Piloted Aircraft
RSM	Reynolds Stress Model
SGS	Sub-grid scale
SGMV	Systematic Grid and Model Verification
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
SMA	Smagorinky
SST	Shear Stress Transport
SUA	Small Unmanned Aircraft
TKE	Turbulent Kinetic Energy
UA	Unmanned Aircraft
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UCA	Unmanned Combat Aircraft
UWD	Urban wind database
VTOL	Vertical Take off and Landing

List of Symbols

Symbols	Definition	Units
Re	Reynolds number	
F	Force	$kg/(m \cdot s^2)$
G	LES filter kernel	
h	Characteristic grid quality	m
H	Building height	m
I	Inertia matrix	
k	Specific kinetic energy	m^2/s^2
K	Control gain	
L	Rotation matrix	
m	Mass	kg
M	Quadrotor motor	
M	Moment vector	
p	Pressure (density nominalized)	m^2/s^2
r	Cell centered local reference frame	
R	Building side dimension	m

S	Surface area	
S	Strain-rate tensor	
t	Time	s
T	Time average window	s
T	Thrust	$kg/(m \cdot s^2)$
u	Velocity component	m/s
U	Total uncertainty	
U	Control output	
V	Total velocity	m/s
W	Wind velocity	m/s
x	Streamwise co-ordinate direction	
y	Cross wind co-ordinate direction	
z	Vertical co-ordinate direction	
X	x co-ordinate position	m
Y	y co-ordinate position	m
Z	z co-ordinate position	m
δ	Kronecker delta	
Δ	LES filter width	m
κ	Von Karman constant	
μ	Dynamic viscosity	$kg/(m \cdot s)$
ν	Kinematic viscosity	m^2/s
ϵ	Error	
ε	Specific kinetic energy dissipation rate	m^2/s^3

ϕ	Roll angle	
Φ	Flow variable	
ψ	Yaw angle	
τ	Stress tensor	m^2/s^2
θ	Pitch angle	
ω	Angular velocity	$1/s$
ζ	Incidence angle	$\circ$

Accents

$\dot{}$	Rate
$\bar{}$	Mean
$\tilde{}$	Filtered
$\rightarrow$	Vector

Superscript

$'$	Fluctuating/SGS component
res	Resolved
tot	Total
p	Numerical accuracy order

Subscript

$\parallel$	Wind parallel
$\perp$	Wind perpendicular
0	Effective surface roughness
$*$	Friction velocity
B	Body frame

c	Filter cutoff width
d	Drag
D	Characteristic building length
E	Earth frame
f	Force
g	Gravity
i	Co-ordinate direction
j	Co-ordinate direction
k	Richardson constant
m	Mean
SGS	Sub-grid scale
t	Turbulent
W	Wind
∞	Freestream
x	Streamwise co-ordinate direction
y	Cross wind co-ordinate direction
z	Vertical co-ordinate direction

Chapter 1

Introduction

The development, utilization, and attention of unmanned aerial vehicles (UAVs) has dramatically increased over the last decade as a result of various global conflicts and ease of public access. A quick distinction is made here between an aerial munition such a cruise missile and a UAV, as the former is a one-time use weapon while the latter is designed to be reused and perform various types of missions. As the complexity grew the term UAV became outdated and enhanced with the concept of an unmanned aerial system (UAS) designed to incorporate the unmanned aircraft (UA), ground control station (GCS), command and communication data links, and any additional required system elements [1]. Further classification of UAs is made with subcategories for remotely piloted aircraft (RPA) and fully autonomous navigation and control systems.

The usefulness of any UAS becomes apparent when considering the additional possible mission types otherwise too mundane or dangerous for piloted aircraft. Therefore, similar to their 'full size' counterparts both fixed-wing and rotor-craft UA platforms

have been developed to make use of the different advantages for various mission profiles. One of the most iconic fixed-wing unmanned combat aircraft (UCA), an armed subclass as a UA, is General Atomics MQ-1 Predator as shown in Figure 1a. In addition to removing valuable pilots from combat situations, the UAS excels in information gathering and relaying for reconnaissance mission profiles. The realization for such a platform came at height of the Cold War when prior to spy satellites the U-2 spy plane was primary means of reconnaissance. After the Soviets shot down the U-2s of Francis Gary Powers in 1960 and Rudolf Anderson, Jr. in 1962, it became clear an unmanned intelligence gathering vehicle could be beneficial. This ultimately culminated in the development of the recent generation platforms such of the Northrop Grumman MQ-4C Triton surveillance UA pictured in Figure 1b.

(a) MQ-1 Predator [2] **(b)** MQ-4C Triton [3]

Figure 1: Sample fixed-wing unmanned aircraft

While not as common as the fix-wing platforms, various rotor-craft UAs have been developed to take advantage of the rotor-crafts vertical take off and landing (VTOL) capabilities. This allows for simpler vehicle deployment off smaller navel vessels and highly localized reconnaissance or engagement of a desired area. The Northrop

Grumman MQ-8 Fire Scout and Schiebel Camcopter S-100, shown in Figures 2a and 2b respectively, are examples of rotor-craft UAs developed to fit such mission types.

(a) MQ-8 Fire Scout [4] **(b)** S-100 Camcopter [5]

Figure 2: Sample rotor-craft unmanned aircraft

The continued success and development of the various UAS platforms, coupled with advances in computational hardware, lead to the miniaturization and the small unmanned aircraft (SUA). This wide general classification includes a range of vehicles from human portable miniature UAs to insect sized micro unmanned aircraft (MUA). Lower development and production costs, an overall simpler systems requirements, and ease of hardware access has lead to an incredible increase in the research, advancement and implementation of SUAs. This boom of interest has occurred in a multitude of different sectors such as the military, academics, industry, and hobbyists. Two examples of the numerous available SUAs are the fixed-wing EMT Aladin and quadrotor Aeryon Scout illustrated in Figures 3a and 3b respectively.

(a) EMT Aladin [6]

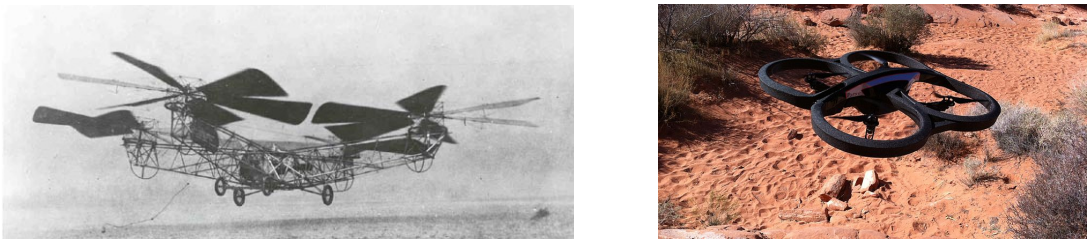
(b) Aeryon Scout [7]

Figure 3: Sample small unmanned aircraft

With comparably less flight time in regards to their 'full size' equivalents, the size and cost of the small UASs still open opportunities not previously available leading to their use in numerous industrial and civilian applications such as, precision agriculture, geological or mine site surveying, forest fire examination or, remote search and rescue operations.

Mutlirotor aircraft designs have have existed since the Breguet-Richet Gryoplane in 1907 and the de Bothezat helicopter in 1922, pictured in Figure 4a, as a solution to the counter torque problems experienced with traditional helicopters. The multirotor platform has recently grown in popularity for several reasons resulting in vehicles such as the Parrot AR Drone in Figure 4b.

The reasons for this increased popularity are due to multirotor vehicles expanding on the usefulness and benefits of standard rotor-craft helicopter platforms with the additional gains of design simplicity, lower costs and smaller frame size. Furthermore, the rise of small and powerful computational hardware has solved



(a) de Bothezat Helicopter [8] **(b)** Parrot AR Drone [9]

Figure 4: Evolution of multirotor aircraft

previous problems of high pilot workload when dealing with the multirotor's complex dynamics, outlined in Chapter 5. This has lead to the multirotor becoming extremely popular, most notably for aerial photography applications, and the focus of much research to further expand its applications. In general a multirotor can be simplified to a flying vehicle with a rotating constant pitch propeller propulsion system at the end of a set of symmetric arms. Therefore, the lift of the quadrotor required for flight is produced solely by the combined thrust of the propellers. While this is less efficient than a comparable helicopter with an actuated blade system, the simplicity of only using a constant pitch results in an economical propulsion system and easier maintenance.

While multirotors can have any number of motors, typical configurations have three, four, six, or eight arms as shown in Figure 5. The thrust of each individual motor is then used to control the attitude and position of the vehicle. The details of how different motors are used to control the vehicles attitude and position are presented in Section 4.2.

Similar to any aircraft design, the frame selection is a function of the intended

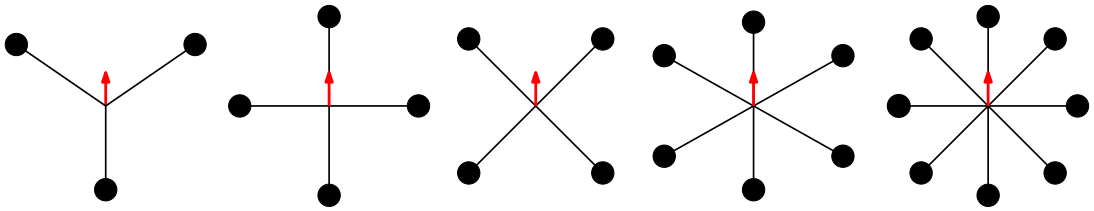


Figure 5: Typical multirotor frames, tri, quad-plus, quad-cross, hex, and octa

mission profile and desired flight characteristics such as speed, agility, endurance or reliability. The quadrotor is a popular configuration for both hobbyists and researches, while it lacks the power and redundancy of the larger multirotors, it offers manufacturing simplicity and lower frame cost leaving budget for other components and systems.

However, the size and weight benefits of SUA's and MUA's, such as a quadrotor, results in a greater susceptibility to external environmental effects such as wind. Various studies have been conducted to analyze this effect such as; thermal updrafts [10] and ridge soaring [11] with constant winds, or flocking [12], trajectory planning [13], and estimation and rejection [14] with wind gusts. Consequently, generating wind is accomplished with methods ranging from simplistic 2D approximations [15], prescribed constant (or linearly varying) [13, 16, 17], wind gust models [11, 14], and computational fluid dynamics (CFD) [18, 19].

The growth in wind modeling complexity is driven by necessity as research and development of various UASs continue and the boundaries of both the current and desired system capabilities are expanded. With an estimated 53% of the worlds population living in an urban environment, and increasing 2% annually [20], there is

little surprise in the growing interest in using SUA's and MUA's in urban environments. While both fixed-wing and rotor-craft platforms are versatile and capable for use in an urban environment, focus in this work is placed on a quadrotor SUA for various missions profiles such as; law enforcement [21], general reconnaissance and surveillance [22], aerial photography for building inspection [23], urban mapping, first responses tool [24], forensic analysis [25], traffic camera, chemical sensors, parcel delivery [26, 27], and catering services [28, 29].

It has been found operating a SUA or MUA such as a quadrotor within an urban environment produces additional challenges due to the vehicle's small size, light weight, and the winds interaction with urban structures [30]. The resulting random transient wind forces generated in a building's wake can cause significant trajectory and pathing deviations for both RPA or fully autonomous unmanned systems. To overcome this environmental influence two main branches of study have formed; design a controller to estimate the wind's external influence and counteract it [14] or design a robust intelligent controller to dynamically adapt to the external fores [31].

This work is performed to complement the design and testing of robust autonomous control algorithms by generating appropriate and representative urban wake fields. However, to do this, a step-back is taken to introduce the required basics of fluid dynamics in the form of boundary layers and bluff body aerodynamics. The concept of the boundary layer, the viscous interaction bridging a solid surface and inviscid outer layer, was first described by Ludwig Prandetl in 1904 [32]. The fluid viscosity coupled with a no-slip condition at the solid boundary results in a characteristic

boundary layer shape and velocity profile with a flat surface as illustrated in Figure 6. From this it is seen the velocity is spatially varying within the boundary layer up to the uniform inviscid freestream.

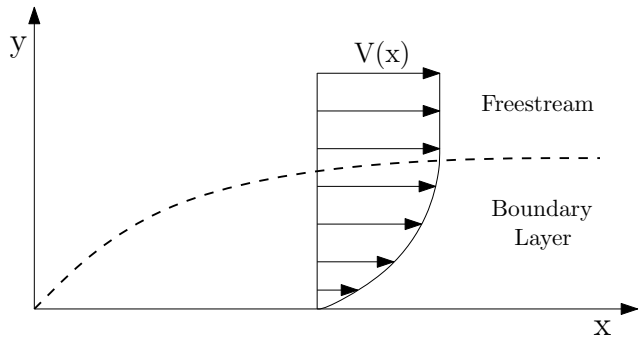


Figure 6: Standard flat surface boundary layer definition

As the lowest part of the Earth's atmosphere moves over the planet's surface a planetary scaled boundary layer is formed, often referred to as the atmospheric boundary layer (ABL) [33]. Similar to the previous ideal boundary layer, the ABL is comprised of two sections, the outer and surface layers, where the latter is further subdivided based on the resulting flow-surface interactions. The flow in the outer freestream is ideally generalized as geostrophic, which is a balance of the pressure and Coriolis forces from Earth's rotation [34]. The inner surface layers and the velocity profile within are highly dependent on the surface roughness and is the layer through which urban environment SUA and MUA flight is most likely to occur.

There are four length scales used in classifying flow in urban environments, regional (100-200 km), city (10-20 km), neighborhood (1-2 km) and street level (100-200 m) [35]. At the regional and city scales the flow around individual buildings is averaged out and the structure's drag results in flow similar to that over a rough surface

resulting in the ABL shape. This scale produces the large scale turbulence and urban heat island effects important for dictating the wind flow models in large pollutant cloud transportation [35]. The surface shear stress produces a friction velocity u_*, a scaling factor for the Monin-Obukhov similarity theory, and in combination with a displacement height d, produces a logarithmic profile for the wind velocity in the surface layer.

$$V(z) = \frac{u_*}{\kappa} \ln\left[\frac{z-d}{z_0}\right] \tag{1}$$

Where κ is the Von Karman constant and z_0 is the effective surface roughness factor. The factor can take values such as 0.0005 m for "smooth" terrain with vegetation like beaches or open country, or 2.0 m for "chaotic" terrain like city centers with a mix of low and high-rise buildings [36]. However, this velocity formation is only applicable for altitudes greater than $2H_m$ were H_m is the mean building height.

The altitude of H_m for typical urban environment is approximately 100 to 200 m, falling into the street level scale. At the street level scale the bottom most portion of the roughness sublayer contains the urban canopy layer where the flow is directly affected by the size and orientation of local objects such as buildings [35]. Since the flow and dispersion at street level scale is the result of the interaction of one or two streets, buildings, or intersections, it is important for studies such as; scalar concentration dispersion in the form of pollutant dissemination [37–40], application of wind loading on buildings [41] and pedestrian comfort level in urban environments [42]. Figure 7 summarizes the logarithmic ABL velocity profile and the reduced velocity the structures produce in the urban canopy layer.

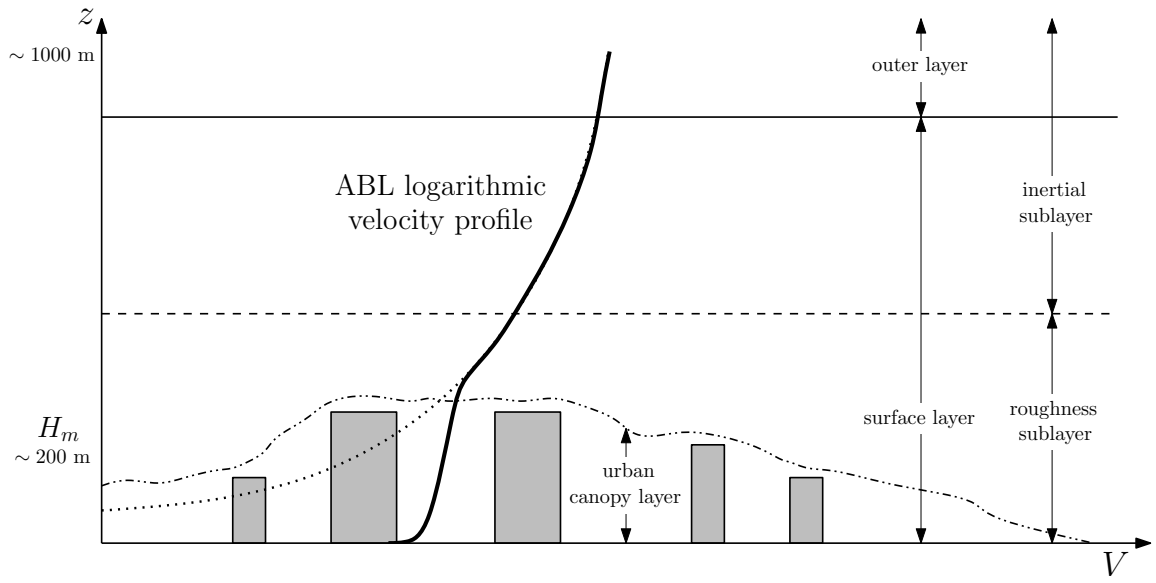


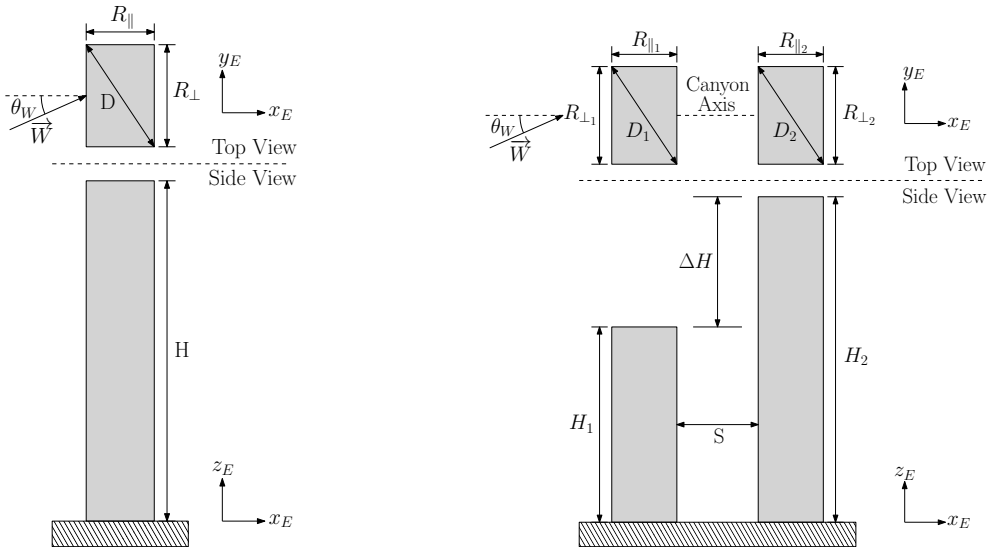
Figure 7: Spatially averaged mean atmospheric boundary layer velocity profile near an urban area. Adapted from Britter [35] and Bottema [43].

For aircraft or UAVs flying outside of the urban effects the ABL velocity profile can be used in conjunction with continuous gust models to represent atmospheric turbulence [44]. Two most common are the Dryden and von Karman models, which describe the velocity components using power spectral density functions [44] while assuming a stationary Gaussian process. Both models are functions of a turbulent length scale and turbulence intensity scale dependent on altitude due to varying global effects such as wind shear and temperature gradients. While the models assume many unrealistic simplifications, they are computationally simple to implement and produce satisfactory turbulent results for general flight dynamics study. For this reason the Dryden model has become a FAA standard when designing aircraft and has been used in the pursuit of developing UAV control schemes to incorporate general external turbulent disturbances [14].

However, the simplifications of the continuous gust models become inappropriate for finding the wind velocity components inside the urban canopy layer due to the dependence on specific geometry. The majority of North American cities are a collection of rectangular prisms and act as a bluff body in the presence of wind. Knowledge of the flow structures, specifically the turbulent wake region, is a result of research in general bluff body aerodynamics [45–48]. Therefore, to resolve the geometrically dependent urban wake fields, computational fluid dynamics (CFD) can be used.

It can be advantageous and simpler to simulate an entire urban domain on the city or neighborhood scales, and some studies have used CFD at this scale for pollutant dispersion [49, 50] or large fixed wing UAVs [18], but this would be computationally expensive to resolve flow scales on the quadrotor size. Since the desired fixed-wing and quadrotor platforms have scales two orders of magnitude lower than the street level, the previous work of Galway et al. [19, 51, 52] is used to reduce the size and computational cost of resolving the wind velocities in a typical North American city. This consists of breaking the urban environment into simplified geometric building blocks for the CFD simulation and then combining different combinations of these blocks to replicate a desired urban environment. The two most basic building blocks are a single building in isolation and an urban canyon, illustrated in Figures 8a and 8b respectively.

Each building block is parametrized with a set of defining characteristics. A single building is parametrized with the ratio of the building length to width ($R_{\perp}/R_{\parallel}$), the



(a) Single building **(b)** Urban canyon

Figure 8: Primary building block parameters

Reynolds number (Re_D), and the wind incidence angle (θ_W). The Reynolds number is based on the freestream wind velocity V_∞, the characteristic length $D = \sqrt{R_\perp + R_\parallel}$, and the kinematic viscosity ν such that,

$$Re_D = \frac{V_\infty \, D}{\nu} \tag{2}$$

The wind vector $\vec{W}$ can be offset at a wind incidence angle θ_W representing the buildings orientation with respect to the freestream wind. An urban canyon geometry is formed when single buildings are close enough to generate flow interactions not otherwise found with a single building in isolation. One such interaction is the generation of a turbulent canyon vortex contained within the canyon street length. The urban canyon is parametrized with a set of characteristic lengths for each building with the addition of the street separation distance S, and a height differential ΔH. The ΔH shown in Figure 8b is negative by convention and

classified as a step up notch.

Therefore by varying the non-dimensional parameters and performing multiple CFD simulations, a database of urban flow fields can be generated and combined to buildup a suitable urban flight environment. This environment can then be used with a flight simulator to design control algorithms and to test autonomous flight performance.

The previous work of Galway et al. [19, 51, 52] generated a database of wake fields using Reynolds-Averaged Navier-Stokes (RANS) CFD methods for testing a fixed wing Aerosonde UAV [52] and Yamaha R-50 rotor-craft [51]. This study builds on the previous work by generating a subset of the database for the application of testing a quadrotor in urban wake fields. For this a single building geometry, with parameters of $Re = 7.30 \times 10^6$, $\theta_W = 0$ and $(R_\perp/R_\parallel)_{ww} = 1$, are used to generate the urban wake field using a large-eddy simulation (LES) CFD method.

As outlined in Sections 2.3 and 2.4 LES is computationally more expensive than RANS due to the additional resolved turbulent motions, however, it is believed resolving these motions are important in the pursuit of designing and testing quadrotor control algorithms for urban flight. With the ever increasing availability of large computation power, compared to even a decade ago, the use of LES methods for urban wind problems has also increased for applications such pollution dispersion problems [37–40], wind loading on buildings [41], and pedestrian comfort [42]. Since LES resolves additional transient turbulent motions, in comparison to constant

prescribed wind, gust models, or RANS based CFD methods, it produces a more complete approximation of a true urban wake field. Therefore this work compares the autonomous flight performance of a quadrotor with various background wind conditions such as constant wind, RANS generated wind and LES generated wind to determine if the additional computational cost of LES is required in the development of appropriate autonomous flight control methods.

Chapter 2

Methodology

2.1 OpenFOAM

The urban wake fields are generated using the Open Source Field Operation and Manipulation (OpenFOAM) CFD package [53]. While the workflow is similar to a commercial product, OpenFOAM is a collection of C++ libraries which provide various solver and utility applications [53]. As illustrated in Figure 9, the solvers of OpenFOAM are able to address many different continuum problems; from incompressible fluid flow to modeling an electric field. Similarly OpenFOAM comes with many useful utilities for problem setup and post-processing data manipulation. The specific utilities employed to generate the wake fields are outlined on the far right of Figure 9 such as the snappyHexMesh meshing utility or the PISO solving algorithm. Version 2.2.x of OpenFOAM is used in combination with the open source 3D visualization application Paraview version 3.12. Both applications run on a workstation with a 64-bit OpenSUSE 12.4 Linux OS, dual Intel Xenon E5-2687W octo-core CPUs and 64 GB of RAM.

Figure 9: Sample of used OpenFOAM solver and utilities libraries.

OpenFOAM is used to regenerate a subset of the wake fields based on previous work using RANS modeling as well as performing the proposed LES methods. OpenFOAM utilizes the finite volume approach for spatial discretization of the governing equations. These equations are solved using one of two solving algorithms, the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) or the Pressure Implicit with Splitting of Operators (PISO) method [54]. The PISO algorithm iteratively calculates the pressure-velocity coupling using a predictor and corrector approach as illustrated in Figure 10. The PISO algorithm is an extension of the SIMPLE method where no under-relaxation is applied and two momentum corrector steps are performed.

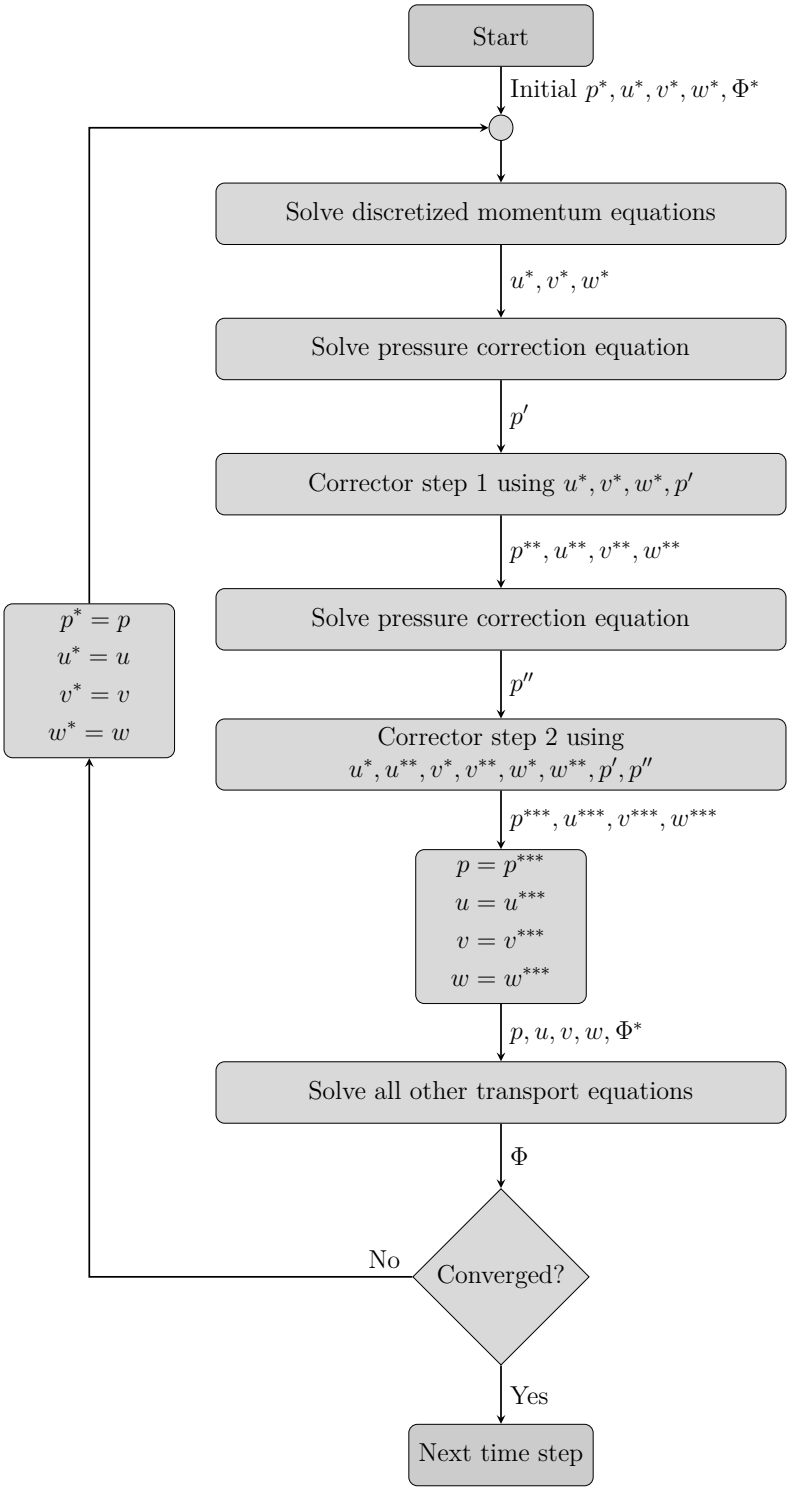


Figure 10: Two corrector step PISO algorithm

The SIMPLE algorithm is used when generating a turbulent velocity field with a steady RANS simulation in advance of the LES to decrease the start up time between the initial uniform conditions and the generation of turbulent structures. The RANS simulation uses the PISO algorithm to transiently solve the governing equations outlined in subsection 2.3 and the timestep is specified such that the Courant-Friedrichs-Lewy (CFL) number is 1.0 in the smallest grid cells. The LES is also solved with the transient PISO algorithm but the timestep is specified such that the CFL number is maintained within a range of 0.4-0.6 in the smallest grid cells. This range of CFL is used to ensure stability as well as allowing for the appropriate turbulent timescales to be resolved [55].

Since OpenFOAM is just a collection of libraries and utilities there are several ways to interact with them. While there are options for graphical user interfaces (GUIs), such as HELYX^{OS}, OpenFOAM has the ability to run under Linux bash shell script control or through Python scripts with pyFoam. To aid the setup, execution and proceeding of additional simulations to generate a LES based wind database the standardized file structure used is shown in Figure 57. After setting the simulation specific details in the various Setup files, such as the freestream wind velocity, geometry names, meshing densities, and parallel processors, the Allrun script is called to perform the steady RANS simulation, map the results and run the LES. The additional collection of scrips in the PostProcessing sub-folder allow for LES wake data extraction, trimming and uploading. The details of the post processing scripts and the file structure are outlined in Appendix A.

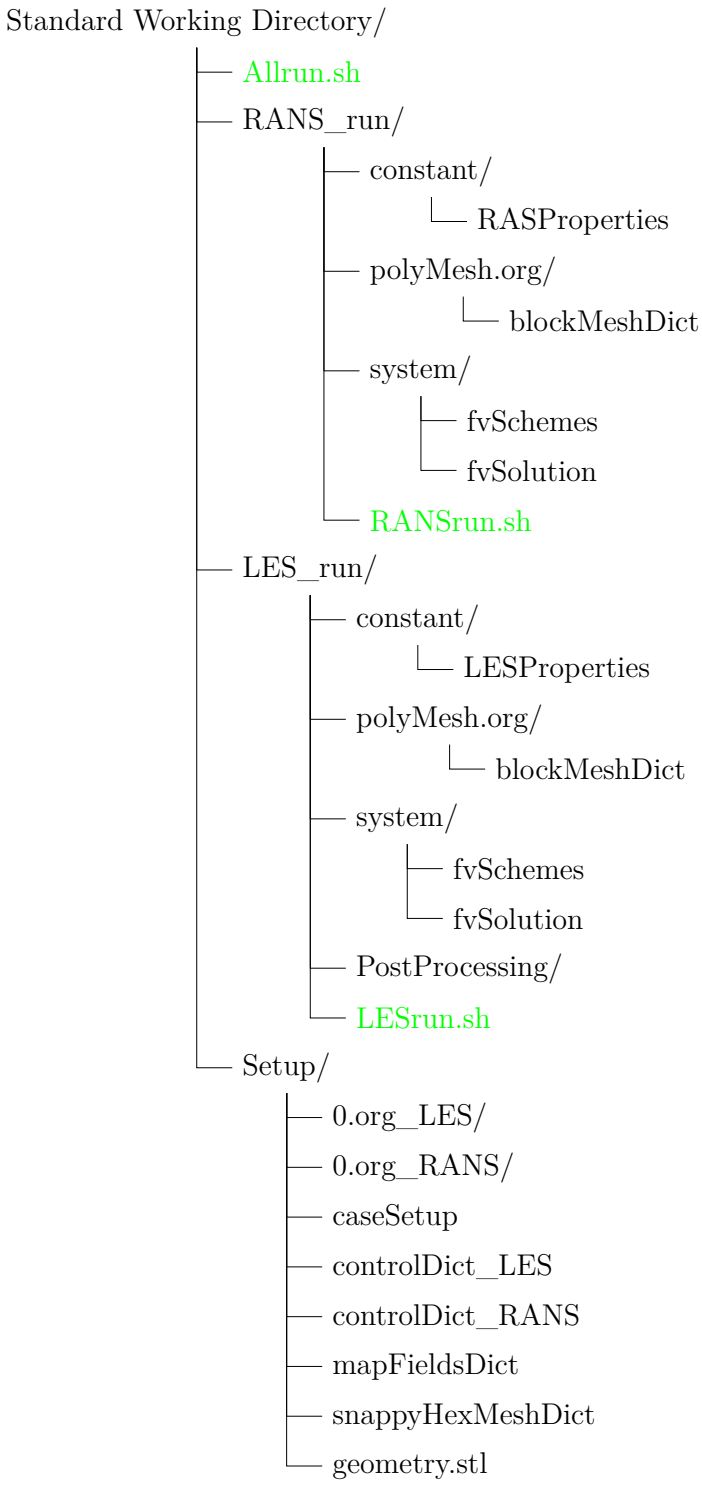


Figure 11: Outline of the standard LES working directory

2.2 Urban Wake Field CFD Modeling

The idea and complication of turbulence in fluid flow has existed since the classic pipe experiment by Osborne Reynolds in 1883 [56]. Reynolds also proposed and popularized a single dimensionless parameter, the Reynolds Number, which describes the flow behavior as a ratio of inertial to viscous forces. While previously introduced for application to the urban environment the general definition takes the form,

$$Re = \frac{VL}{\nu} \tag{3}$$

Over the ensuing decades, the importance of understanding and subsequently the desire to model turbulent flows forced much advancement in the subject area. The notion of turbulent transition was expanded with the concepts such as turbulent length and time scales and the energy cascade. For the sake of brevity the development history of simulating turbulent flow in terms of CFD is not presented [57], rather only the resulting methods and models.

The difficulty arises when the nonlinear differential equations of fluid motion, the Navier-Stokes equations, are further complicated by the addition of terms from the viscous-turbulence relationship. The methods used to numerically solve turbulent flow fall into one of three categories; Reynolds-averaged Navier-Stokes (RANS), large-eddy simulation (LES) and direct numerical simulation (DNS). As previously mentioned this work expands on previous RANS based methods with LES and for this reason the following sections will only outline their characteristics and equations.

2.3 Reynolds-Averaged Navier-Stokes Method

One method to reduce the extensive computational requirements to resolve all of the spatial and temporal scales in turbulent flow is to apply Reynolds decomposition. The basis of the decomposition is time averaging of the flow properties defined as,

$$\bar{\Phi}(x_i) \equiv \frac{1}{T}\int_0^T \Phi(x_i, t)dt \tag{4}$$

where the domain flow quantities $\Phi(x_i, t)$, are broken into mean $\overline{\Phi}(x_i)$ and a fluctuation components $\Phi'(x_i, t)$ such that,

$$\Phi(x_i, t) = \overline{\Phi}(x_i) + \Phi'(x_i, t) \tag{5}$$

and illustrated by,

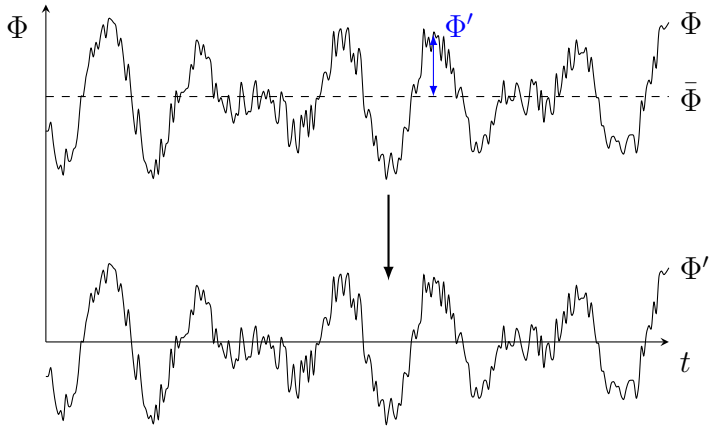


Figure 12: Reynolds decomposition of Φ over time

The time averaging and decomposition gives rise to Reynolds operators, linear algebraic operators on the governing functions. From the definitions of the time averaging in Equation (4) and the fluctuation component in Figure 12, the two most important Reynolds decomposition properties arise such that,

$$\overline{\overline{\Phi}} = \overline{\Phi} \tag{6a}$$

$$\overline{\Phi'} = 0 \tag{6b}$$

Combining these with the incompressible continuity and momentum equations,

$$\frac{\partial U_i}{\partial x_i} = 0 \tag{7}$$

$$\frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial P}{\partial x_i} + \nu \frac{\partial^2 U_i}{\partial x_j \partial x_j} \tag{8}$$

where U is the total velocity, P is the pressure, and ν is the kinematic viscosity. Substituting the Reynolds decomposition and simplifying results in the incompressible RANS continuity equation for the mean flow,

$$0 = \frac{\partial U_i}{\partial x_i} \tag{9a}$$

$$0 = \frac{\partial \left(\overline{u}_i + u_i'\right)}{\partial x_i} \tag{9b}$$

$$0 = \frac{\partial \overline{u}_i}{\partial x_i} + \frac{\partial u_i'}{\partial x_i} \tag{9c}$$

$$0 = \frac{\overline{\partial \overline{u}_i}}{\partial x_i} + \frac{\overline{\partial u'}_i}{\partial x_i} \tag{9d}$$

$$0 = \frac{\partial \overline{u}_i}{\partial x_i} \tag{9e}$$

To derive the RANS momentum equations, the Reynolds decomposition is applied to both the velocity and pressure quantities in Equation (8),

$$\underbrace{\frac{\partial \left(\overline{u}_i + u_i'\right)}{\partial t}}_{I} + \underbrace{\left(\overline{u}_j + u_j'\right)\frac{\partial (\overline{u}_i + u_i')}{\partial x_j}}_{II} = \underbrace{-\frac{1}{\rho}\frac{\left(\overline{p} + p'\right)}{\partial x_i}}_{III} + \underbrace{\nu \frac{\partial^2 (\overline{u}_i + u_i')}{\partial x_j \partial x_j}}_{IV} \tag{10}$$

The linearity of the temporal discretization I, pressure III and viscous stress terms IV allow simple application of the Reynolds operators and time averaging following the same procedure as the continuity equation. However, application of the RANS averaging on the convection acceleration term, II, requires additional work due to the products arising from the multiplication,

$$(\overline{u}_j + u'_j)\frac{\partial(\overline{u}_i + u'_i)}{\partial x_j} \tag{11a}$$

$$\overline{u}_j\frac{\partial \overline{u}_i}{\partial x_j} + \overline{u}_j\frac{\partial u'_i}{\partial x_j} + u'_j\frac{\partial \overline{u}_i}{\partial x_j} + u'_j\frac{\partial u'_i}{\partial x_j} \tag{11b}$$

$$\underbrace{\overline{\overline{u}_j\frac{\partial \overline{u}_i}{\partial x_j}}}_{I'} + \underbrace{\overline{\overline{u}_j\frac{\partial u'_i}{\partial x_j}}}_{II'} + \underbrace{\overline{u'_j\frac{\partial \overline{u}_i}{\partial x_j}}}_{III'} + \underbrace{\overline{u'_j\frac{\partial u'_i}{\partial x_j}}}_{IV'} \tag{11c}$$

Applying the definition of time averaging, Equation (4), to the second and third terms, II' and III', reduces them to zero following,

$$\overline{\overline{u}_j\frac{\partial u'_i}{\partial x_j}} = \frac{1}{T}\int_0^T \left(\overline{u}_j\frac{\partial u'_i}{\partial x_j}\right) dt \tag{12a}$$

$$= \overline{u}_j\frac{\partial}{\partial x_j}\left(\frac{1}{T}\int_0^T (u'_i)dt\right) \tag{12b}$$

$$= \overline{u}_j\frac{\partial \overline{u'_i}}{\partial x_j} \tag{12c}$$

$$= 0 \tag{12d}$$

Therefore collecting the averaged terms and simplifying Equation (10),

$$\frac{\partial \overline{u}_i}{\partial t} + \underbrace{\overline{\overline{u}_j\frac{\partial \overline{u}_i}{\partial x_i}}}_{I''} + \underbrace{\overline{u'_j\frac{\partial u'_i}{\partial x_i}}}_{II''} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu\frac{\partial^2 \overline{u}_i}{\partial x_j \partial x_j} \tag{13}$$

A substitution is made for the remaining nonlinear terms, I'' and II'', culminating in the time-averaged momentum equations,

$$\frac{\partial \overline{u}_i}{\partial t} + \overline{\frac{\partial \overline{u}_i \overline{u}_j}{\partial x_j}} + \overline{\frac{\partial u_i' u_j'}{\partial x_j}} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu \frac{\partial^2 \overline{u}_i}{\partial x_j \partial x_j} \tag{14a}$$

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial \overline{u}_i \overline{u}_j}{\partial x_j} + \frac{\partial \overline{u_i' u_j'}}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu \frac{\partial^2 \overline{u}_i}{\partial x_j \partial x_j} \tag{14b}$$

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial}{\partial x_j}(\overline{u}_i \overline{u}_j) = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu \frac{\partial^2 \overline{u}_i}{\partial x_j \partial x_j} - \frac{\partial}{\partial x_j}\overline{u_i' u_j'} \tag{14c}$$

The $\overline{u_i' u_j'}$ term is a product of the non-linear convection term and represents the convective momentum transfer from turbulent eddies [54]. This quantity is called the Reynolds stress tensor which cannot be related to the unknown mean velocity and pressure fields and therefore results in the 'turbulent closure problem' [57]. Therefore to predict the Reynolds stresses based on the mean flow and close the system of equations, turbulence models are employed.

All current models use one of two methods to ascertain the unknown Reynolds stresses, the eddy viscosity relation or the Reynolds stress model (RSM). The most common models use eddy viscosity which is loosely based on Newton's law of viscosity where the shear stress is proportional to the strain rate,

$$\tau = \mu \frac{\partial u}{\partial x} \tag{15}$$

Where μ is the fluid's dynamic viscosity. The Boussinesq hypothesis expands this expression for a turbulent flow case to relate Reynolds stresses to the mean rates of viscous deformation,

$$-\overline{u_i' u_j'} = \nu_t \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right) - \frac{2}{3} k \delta_{ij} \tag{16}$$

Where ν_t is the turbulent eddy viscosity, k is the specific turbulent kinetic energy and δ_{ij} is the Kronecker delta. The contribution of this additional term ensures a correct relation for the normal Reynolds stress components [54]. While this approximation aids in the closure problem, it introduces an additional unknown and complex variable ν_t. Therefore the principal goal of any turbulence model is to calculate ν_t either with simple relationships, such as Prandtl's mixing length model or through additional transport equations, such as the $k - \varepsilon$ or $k - \omega$ SST models.

The two equation $k - \varepsilon$ model is briefly outlined as it is used to generate an urban wake field based on previous urban wind generation work of Galway et al. [19,51,52]. The $k - \varepsilon$ model introduces a transport equation for the turbulent kinetic energy k and the specific turbulent kinetic energy dissipation rate ε to provide velocity and length scales to find the unknown turbulent eddy viscosity. The specific turbulent kinetic energy is defined as,

$$k = \frac{1}{2} \overline{u_i u_i} = \frac{1}{2} (\overline{u'^2} + \overline{v'^2} + \overline{w'^2}) \tag{17}$$

and the specific turbulent kinetic energy dissipation rate is defined as,

$$\varepsilon = 2\nu \overline{s_{ij}' s_{ij}'} \tag{18}$$

where s_{ij}' is the fluctuating strain-rate tensor,

$$s_{ij}' = \frac{1}{2} \left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i} \right) \tag{19}$$

Using these definitions, the RANS momentum Equation (14c), and considerable manipulation, produces the additional transport equations shown from Tennekes

and Lumley [58] or Launder and Spalding [59],

$$\frac{k}{\partial t} + \overline{u}_j \frac{\partial k}{\partial x_j} = -\overline{u_i' u_j'} \frac{\partial \overline{u}_i}{\partial x_j} - \varepsilon + \frac{\partial}{\partial x_j}\left[(\nu + \nu_t/\sigma_k)\frac{\partial k}{\partial x_j}\right] \tag{20a}$$

$$\frac{\varepsilon}{\partial t} + \overline{u}_j \frac{\partial \varepsilon}{\partial x_j} = -C_1 \frac{\varepsilon}{k} \overline{u_i' u_j'} \frac{\partial \overline{u}_i}{\partial x_j} - C_2 \frac{\varepsilon^2}{k} + \frac{\partial}{\partial x_j}\left[(\nu + \nu_t/\sigma_\varepsilon)\frac{\partial \varepsilon}{\partial x_j}\right] \tag{20b}$$

Finally the system of equations are closed by relating these transport equations to the eddy viscosity using dimensional analysis resulting in,

$$\nu_t = C_\nu \frac{k^2}{\varepsilon} \tag{21}$$

The final requirement in closing the model is addressing the five unknown constants. The standard $k - \varepsilon$ constants are used, as shown below, and are the result of comprehensive data fittings from experimental results,

$$C_\nu = 0.09 \qquad C_1 = 1.44 \qquad C_2 = 1.92 \qquad \sigma_\varepsilon = 1.30 \qquad \sigma_k = 1.00$$

2.4 Large-Eddy Simulation Method

While RANS based methods are the most common in the engineering industry, the available turbulent models presume universal behavior of the various turbulent scales. However, only the small turbulent eddies are more universal and isotropic in nature in comparison to the large energy containing eddies. While the influence of the small eddies on the flow is required, resolving down to the Kolmogorov length scale is computationally expensive in terms of spacial and temporal resolution. LES was created to directly resolve the transient and geometry dependent motions while the energy draining effects of the small eddies on the resolved flow is modeled to

save computational cost [60]. Because the large-scale unsteady motions are directly resolved with LES, it can be expected to be more accurate than RANS methods when modeling flow with large scale unsteadiness, such as flow over bluff bodies like urban structures [60].

At its core LES is a low-pass filter, applied to attenuate high frequency turbulent motions while leaving the low frequency motions unaltered. This is performed by applying a spacial filter with a cutoff width to define the resolved and modeled length scales. Figure 13 illustrates the turbulent kinetic energy spectrum as a function of wavenumber and how the filter cutoff width (k_c) separates the resolved and sub-grid-scale (SGS) length scales.

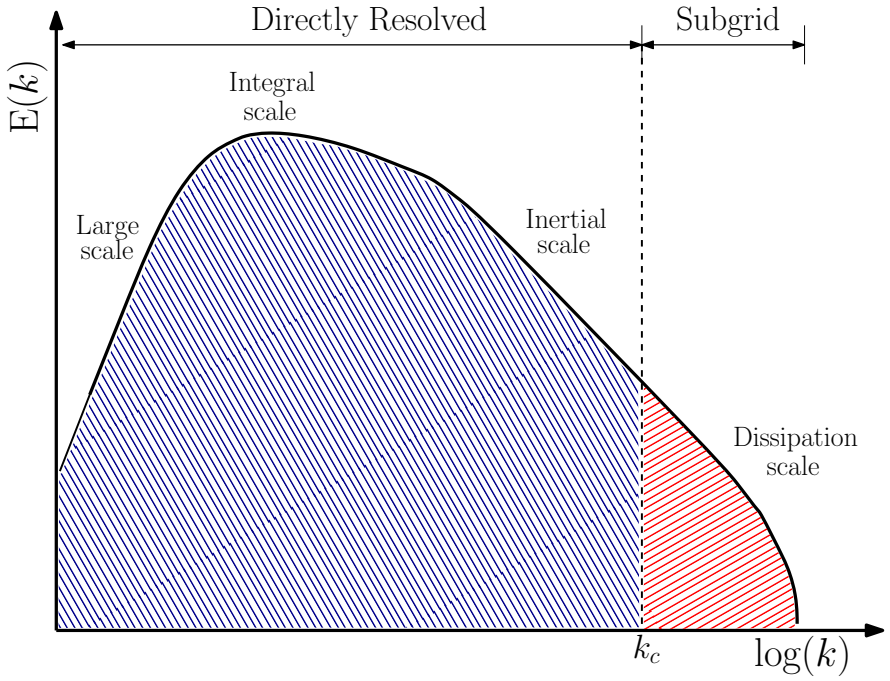


Figure 13: Turbulent kinetic energy spectrum with LES cutoff width

The definition of the spatial filtering process, as introduced by Leonard [61], takes the form of,

$$\tilde{\Phi}(x,t) = \int_V G(r,x,\Delta)\Phi(x-r,t)\mathrm{d}r \tag{22}$$

or,

$$\widetilde{\Phi} = G \cdot \Phi \tag{23}$$

where G is a filter function, Δ is the filter width, Φ is the original unfiltered flow variable, $\tilde{\Phi}$ is the filtered flow variable, x is a global coordinate frame, and r is a cell local axis. The residual field is defined such that,

$$\Phi(x,t)' \equiv \Phi(x,t) - \tilde{\Phi}(x,t) \tag{24}$$

or,

$$\Phi' = (1-G) \cdot \Phi \tag{25}$$

where the total flow variables (Φ) are separated into filtered ($\widetilde{\Phi}$) and subgrid-scale components (Φ'). While visually similar to the Reynolds decomposition of Equation (5), the filtering does not generally follow the rules of a Reynolds operator, most notably,

$$\widetilde{\widetilde{\Phi}} \neq \widetilde{\Phi} \tag{26a}$$

$$\tilde{\Phi}' \neq 0 \tag{26b}$$

since,

$$\widetilde{\widetilde{\Phi}} = G \cdot G \cdot \Phi = G^2 \cdot \Phi \neq \widetilde{\Phi} = G \cdot \Phi \tag{27a}$$

$$\widetilde{\Phi'} = G \cdot (1-G) \cdot \Phi \neq 0 \tag{27b}$$

The decomposition of Equation (24) and filter properties of Equations (26a) and (26b) are artistically illustrated in Figure 14. From this it is very clear that; a double filter is not equal to a single filter application ($\widetilde{\widetilde{\Phi}} \neq \widetilde{\Phi}$) and the filtered residual is not zero ($\tilde{\Phi}' \neq 0$).

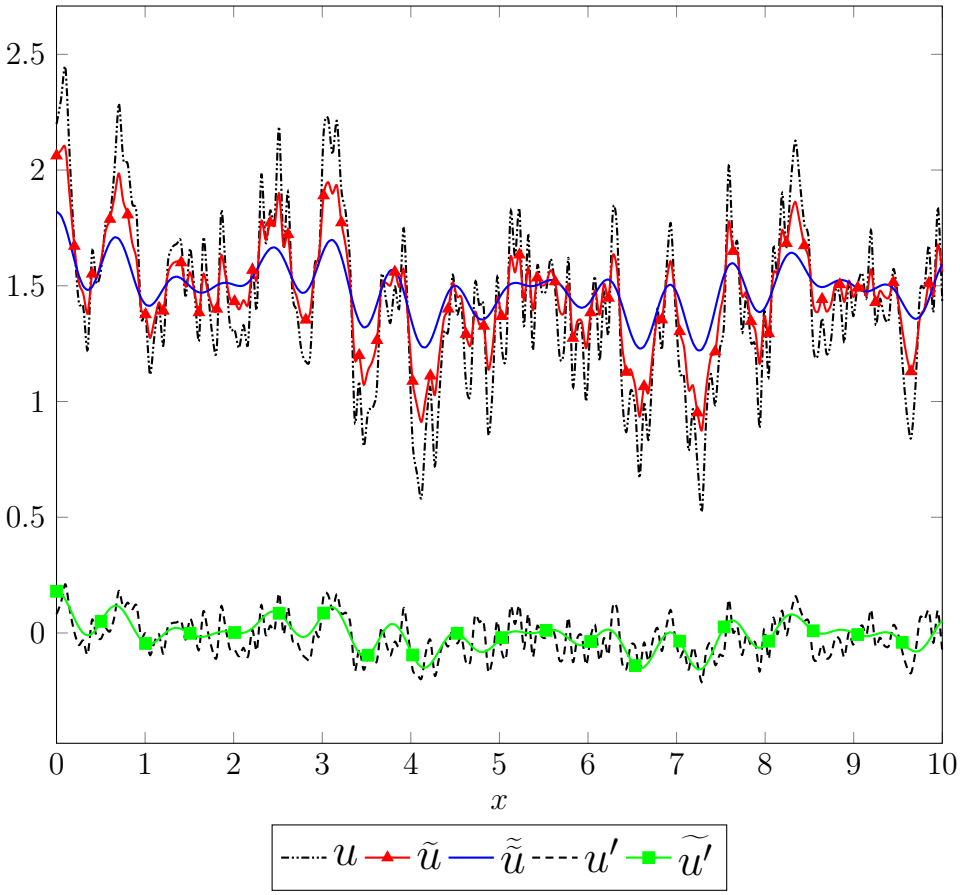


Figure 14: LES decomposition and filter properties. Adapted from Pope [60].

The common most forms of the filter function (G) for LES include the box, Gaussian, and sharp spectral respectively defined in one dimension as,

$$G(x,\Delta) = \begin{cases} 1/\Delta & : |x-r| \leq \Delta/2 \\ 0 & : otherwise \end{cases} \tag{28a}$$

$$G(x,\Delta) = \sqrt{\frac{6}{\pi\Delta^2}} \exp\left(-\frac{6(x-r)^2}{\Delta^2}\right) \tag{28b}$$

$$G(x,\Delta) = \frac{\sin[\pi(x-r)/\Delta]}{\pi(x-r)} \tag{28c}$$

These filters are shown in Figure 15, illustrating both the active interval, normalized by Δ, and the filters function's value.

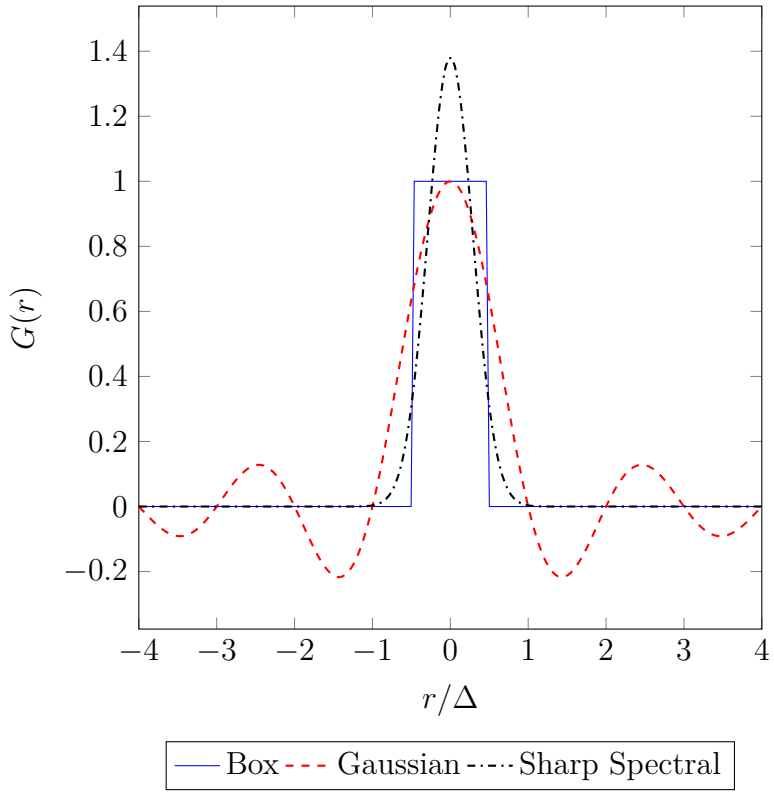
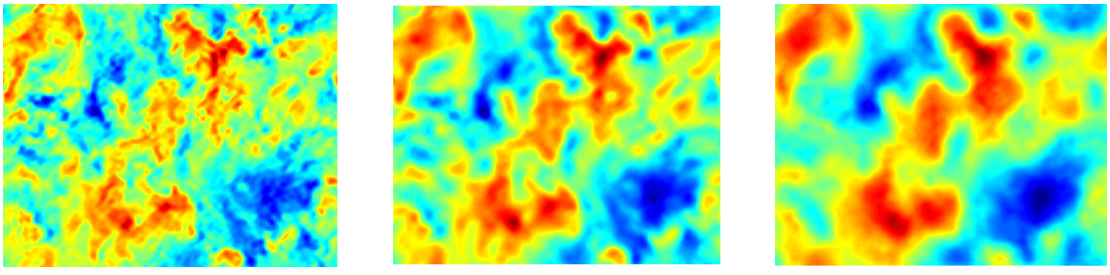


Figure 15: Common LES filter functions. Adapted from Pope [60].

The box filter is applied in finite volume implementations of LES, where the Gaussian and spectral cutoff filters are preferred in the research literature [54]. The Gaussian filter has the advantage of being smooth and differentiable [60], while the sharp spectral eliminates all wave numbers above a chosen frequency [54]. However, since the box filter is just an average over the filter interval, and the flow variables are piecewise linear functions of x for finite volume methods, if the filter width is equal to the grid-spacing, a box filter is simply a local cell spatial average. [55]. In other words, the filter width is indicative of the size of eddies retained (see Figure 13), and can be chosen to be any size. Since finite volume methods only retain a single node value for each cell there is no resolution benefit to specifying a filter width smaller than the grid size [54]. This is classified as implicit LES as the filter width is implicitly determined from the cell size, and the most accepted method of defining the filter width is to use the cube root of the cell volume,

$$\Delta = \sqrt[3]{\Delta_x \Delta_y \Delta_z} \tag{29}$$

An example of a box filter application is shown in Figure 16 using the velocity field resolved from a DNS [62]. A box filter is applied with an increasing filter width between Figures 16b and 16c and illustrates how the smaller higher frequency turbulent eddies are averaged and smoothed out (lower values of k_c in Fiugre 13).



(a) DNS velocity field [63] **(b)** Box filter, $\Delta = L/32$ [63] **(c)** Box filter, $\Delta = L/16$ [63]

Figure 16: Box filter example using DNS velocity field, domain size is L^3.

Derivation of the LES governing equations begins with the incompressible continuity and momentum equations, Equations (7) and (8), and application of the filter function,

$$\widetilde{\frac{\partial u_i}{\partial x_i}} = 0 \tag{30}$$

An important simplification can be made if the filter function is able to commute with temporal and spatial differentiation. It is for this reason the box filter shown in Equation (28a) is only a spatial filter (independent of time) and made locally independent in space through $|x - r|$ (cell based). While this does not hold for all possible filter functions, application of the box filter results in filtered continuity equation,

$$\frac{\partial \widetilde{u}_i}{\partial x_i} = 0 \tag{31}$$

To derive the LES momentum equations, the LES filtering is applied and the linear terms are simplified in a similar manner to their RANS counterparts,

$$\frac{\partial \widetilde{u}_i}{\partial t} + \frac{\partial}{\partial x_j}(\widetilde{u_i u_j}) = -\frac{1}{\rho}\frac{\partial \widetilde{p}}{\partial x_i} + \nu \frac{\partial^2 \widetilde{u}_i}{\partial x_j \partial x_j} \tag{32}$$

Once again the non-linear convection term $\widetilde{u_i u_j}$ is troublesome to express in terms of known flow variables as the filtered products is different than the product of filtered velocities ($\widetilde{u_i u_j} \neq \widetilde{u}_i \widetilde{u}_j$). Therefore a modelling approximation is introduced to account for the residual-stress, called the SGS stress tensor,

$$\tau_{ij} = \widetilde{u_i u_j} - \tilde{u}_i \tilde{u}_j \tag{33}$$

which is not dissimilar to the Reynolds stress tensor in Equation (14c),

$$\overline{u_i' u_j'} = \overline{u_i u_j} - \overline{u}_i \overline{u}_j \tag{34}$$

The tensor has the property such that $|\tau_{ij}| \to 0$, as $\Delta \to 0$, ultimately resulting in a DNS solution in the limit of a small mesh spacing [55]. Substituting Equation (33) into Equation (32) results in the filtered or LES momentum equations,

$$\frac{\partial \widetilde{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\widetilde{u}_i \widetilde{u}_j\right) = -\frac{1}{\rho}\frac{\partial \widetilde{p}}{\partial x_i} + \nu \frac{\partial^2 \widetilde{u}_i}{\partial x_j \partial x_j} - \frac{\tau_{ij}}{\partial x_j} \tag{35}$$

Similar to RANS Equation (14c) for $\overline{u}_i$, the governing LES equations are unclosed and require modelling of the SGS stress tensor τ_{ij}.The type and size of filter is indirectly introduced onto the velocity field though this SGS stress tensor [60]. Applying the decomposition of Equations (24) to the first term on the right side of Equation 33 it can be separated into terms with some physical significance,

$$\tau_{ij} = \widetilde{(\widetilde{u}_i + u_i')(\widetilde{u}_i + u_i')} - \widetilde{u}_i \widetilde{u}_j \tag{36a}$$

$$\tau_{ij} = \widetilde{\widetilde{u}_i \widetilde{u}_j} + \widetilde{\widetilde{u}_i u_j'} + \widetilde{\widetilde{u}_j u_i'} + \widetilde{u_i' u_j'} - \widetilde{u}_i \widetilde{u}_j \tag{36b}$$

$$\tau_{ij} = \underbrace{\widetilde{\widetilde{u}_i \widetilde{u}_j} - \widetilde{u}_i \widetilde{u}_j}_{L_{ij}} + \underbrace{\widetilde{\widetilde{u}_i u_j'} + \widetilde{\widetilde{u}_j u_i'}}_{Cij} + \underbrace{\widetilde{u_i' u_j'}}_{R_{ij}} \tag{36c}$$

where L_{ij} is the Leonard stress tensor, C_{ij} is the cross-stress tensor component, and R_{ij} is the Reynolds subgrid tensor. The Leonard term can be computed from the filtered velocity field and represents the interaction between the resolved and SGS scales. The cross-stress relates the (typical) energy transfer from the large filtered and smaller modeled eddies. The Reynolds stress term is analogous to the components of the stress tensor in the RANS formulation, and represents the effect of the small eddy interaction with one another. In comparison to the RANS method, the Leonard and cross stress terms arise from the difference in the double time averaging and double filtering as $\widetilde{\widetilde{\Phi}} \neq \widetilde{\Phi}$. Further discussion on the significance of the stresses is found in Section 13.5.2 of Pope [60].

While there were early attempts to model each subgrid stress component independently, all of the stresses are grouped into a single subgrid stress tensor and modeled as a whole. This is mainly due to the preservation of the Galilean invariant (independent of inertial frame) properties of the continuity and momentum equations, since the filtering of individual terms such as the cross and Leonard stresses are not Galilean invariant but the total sum is [55, 57, 64].

Similar to the problem faced with the RANS momentum equations a model is appled to close Equations (31) and (35) and relate the SGS stress tensor to known flow quantities. The Boussinesq eddy viscosity assumption is once again applied where the SGS stress tensor is proportional to the local filtered rate of strain tensor and SGS viscosity,

$$\tau_{ij} = -\nu_{SGS}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right) + \frac{1}{3}\tau_{ii}\delta_{ij} \tag{37}$$

while the term τ_{ii} is included to ensure the sum of the modeled normal SGS stresses are equal to the kinetic energy of the SGS eddies [54] (it is grouped in with the filtered pressure term and therefore does not require additional modeling). However, a sub-grid scale model is required for the new unknown sub-grid scale eddy-viscosity ν_{SGS}. Similar to the requirements of a RANS turbulence model, a SGS model is used to close the system of equations and imitate the energy cascade energy drain on the resolved flow.

In this work the one-equation eddy viscosity model, with one variant given by

Yoshizawa and Horiniti [65] is used to relate a characteristic length scale and a velocity scale of SGS eddy size. This model improves the equilibrium assumption of the original Smagorinsky model which becomes less accurate for flow conditions such as free shear layers, separating and reattaching flows, boundary layers and wall dominated domains like pipes and channels [55]. In addition to, this the one-equation model possesses the ability to predict backscatter and has higher numerical stability making it computationally easier than the Smagorinsky model [66]. By using the definition of the velocity scale the one-equation eddy model has shown to perform better to model transitional or large scale unsteadiness flows in comparison to an algebraic relation [55]. The length scale is taken to be the filter width, or grid spacing, while the velocity scale is taken as the square root of the specific SGS turbulent kinetic energy $\sqrt{k_{SGS}}$ such that,

$$\nu_{SGS} = C_k \Delta \sqrt{k_{SGS}} \tag{38}$$

Analogous to RANS like turbulent modeling, a transport equation is introduced to account for the effects of convection, diffusion, production and destruction of this energy,

$$\frac{k_{sgs}}{\partial t} + \widetilde{u}_j \frac{\partial k_{sgs}}{\partial x_j} = \frac{\partial}{\partial x_j}\left[(\nu_{sgs} + \nu)\frac{\partial k_{sgs}}{\partial x_j}\right] - \tau_{ij}\left(\frac{\partial \widetilde{u}_i}{\partial x_j} + \frac{\partial \widetilde{u}_j}{\partial x_i}\right) - C_\varepsilon(k_{sgs}^{3/2}/\Delta) \tag{39a}$$

where the specific SGS kinetic energy dissipation rate are defined as,

$$\varepsilon_{SGS} = C_\varepsilon(k_{SGS}^{3/2}/\Delta) \tag{40}$$

Lastly the equations are closed with the standard one-equation eddy constants,

$$C_k = 0.094 \qquad C_\varepsilon = 1.048$$

Chapter 3

Grid Refinement Study

3.1 Computational Domain

A computational grid is generated with OpenFOAM's native meshing utility snappyHexMesh which produces a 3-dimensional mesh consisting of hexahedra and split-hexahedra elements. The utility operates by edge splitting local cells of the block background mesh and snapping them to the edges of the surface geometry in Stereolithography (.stl) format [53]. The cells located inside the geometry are then removed to produce the Cartesian hex mesh. The size of the domain is generated following the standards outlined in the COST Action 732 [67] and the Agriculture Institute of Japan (AIJ) guidelines [68] for using CFD to simulate flows in urban environments. The domain sizing of the single square building and the boundary conditions are shown in Figure 17, where $R_{\parallel}$ is the wind parallel side length.

A mapped inlet boundary condition is used to generate synthetic transient inlet turbulence. The velocity components are sampled at a plane 12 building widths

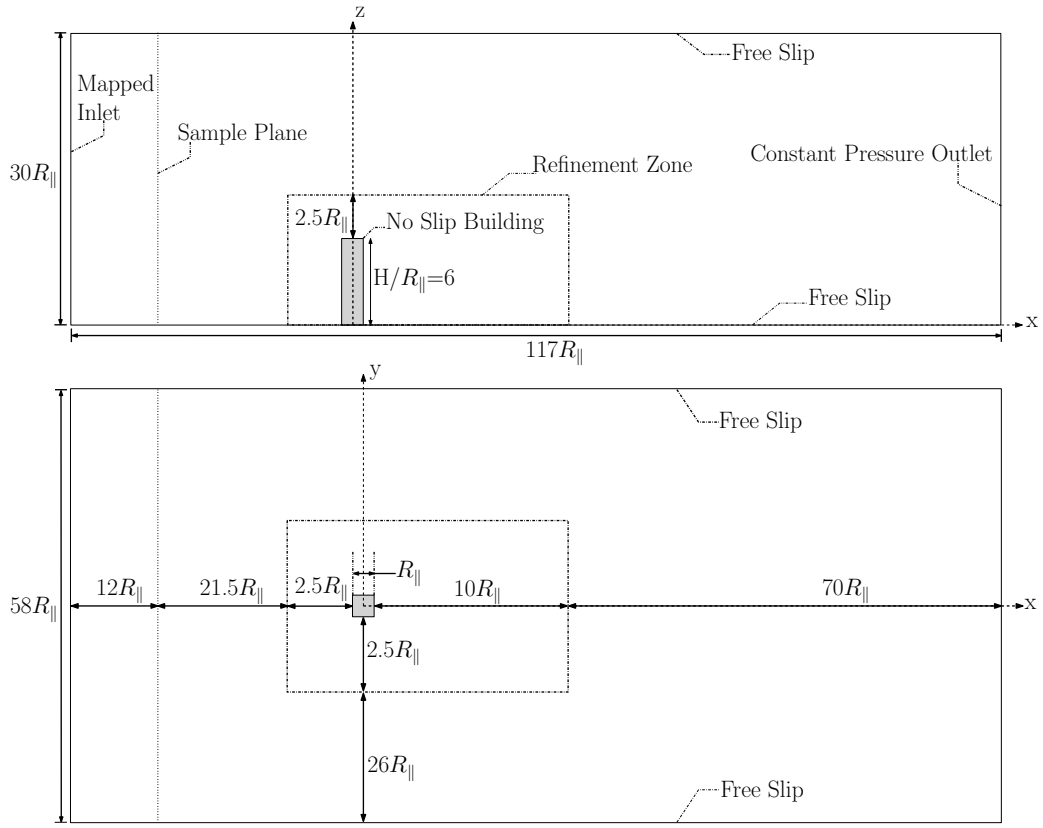
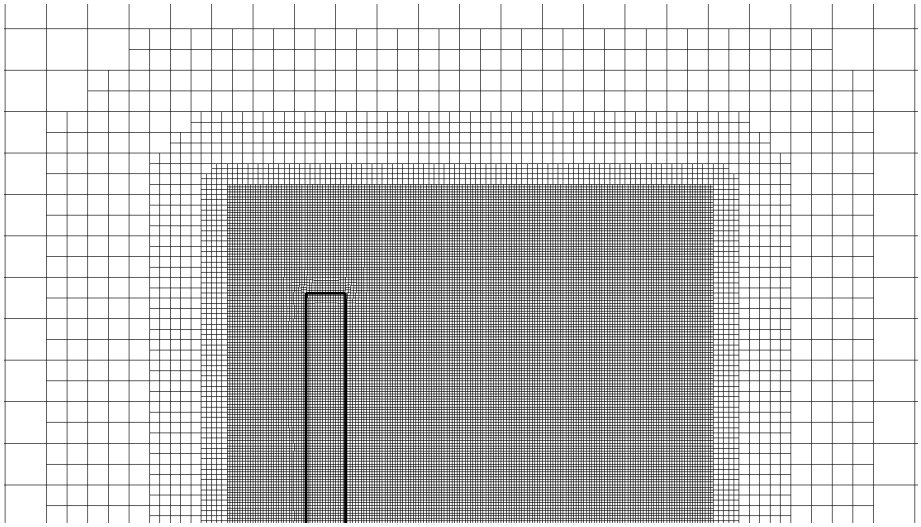


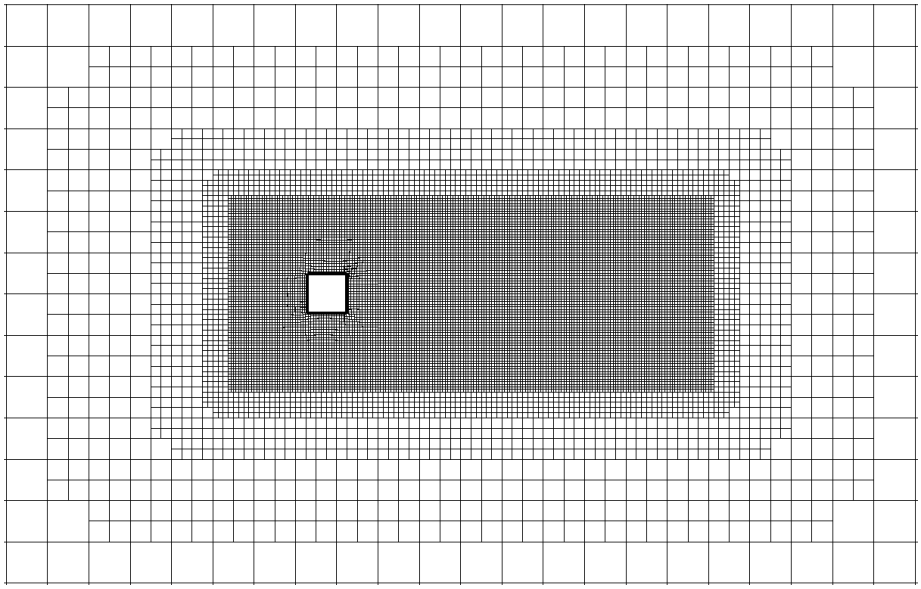
Figure 17: Computational domain, boundary conditions and wake refinement zone

downstream from the inlet, scaled to ensure the bulk flow rate is constant, and applied to the inlet plane [55]. A free slip boundary condition is applied to the ground, side walls and top. Local grid refinement is used to reduce the computational cost and limit the most refined region to the wake as shown in Figure 18. This introduces a commutation error which can produce errors on the order of the SGS stress [55] due to the changing filter width which violates the commutation with differentiation assumption (which requires a constant filter size). To minimize the influence of this error, gradual levels of successive refinement are used to reduce sudden changes in the filter width (shown in Figures 18a and 18b). The refinement boundaries are also

placed away from the regions of direct interest (near the building and within the wake) as the error will be greatest at the refinement boundaries and will reduce as the distance from these refinement boundaries increases [55].



(a) Side view, X-Z plane



(b) Top view, Y-Z plane

Figure 18: Cartesian hex mesh with gradual cell refinement

3.2 LES Verification and Validation Study

In this work the errors associated with spatial discretization are assessed by comparing time averaged velocity profiles across the wake and by calculating the LES Index Quality [69] (LES_IQ) between successive levels of grid refinement. A grid independent solution with LES is more difficult to determine than for RANS simulations in that the degree of model approximation applied, in addition to the numerical accuracy of that model, depends directly on the grid size [70] through the filter width. Therefore, the LES_IQ uses results from two different grid densities to perform a Richardson extrapolation for establishing how the results compare to a theoretical solution in which the filter width is so small as to eliminate the use of the sub-grid scale approximation. It is then possible to establish what percentage of the turbulence is modelled directly in the LES simulations for each grid (and conversely how much of the turbulence is represented by the SGS model).

Table 3: Refined area grid details

Name	# building cells	Total # cells	CPU Hours
Coarse	10x10x60	1.13×10^6	0.838
Medium	15x15x90	2.43×10^6	3.383
Fine	22x22x132	7.67×10^6	17.696

Three grids of increasing density are used for verification purposes. Each successive refinement increases the node count on the building surface by 50% in each direction as summarized in Table 3. The coarsest grid has 10 nodes along the building side parallel to the freestream wind ($R_{\|}$) while in the finest grid this value is increased to 22. The computational time shown is that required to generate 30 seconds of LES results.

The first 6 seconds are used to allow the mapped flow field from a steady RANS simulation to transition into the domain and produce coherent vortical structures, while the flow is then time averaged over the remaining 24 seconds. All verification and validation comparisons are performed across three vertical and three horizontal lines downstream of the building as shown in Figure 19. These locations are a subset of measurement locations used in the wind tunnel experiments of Meng and Hibi [71].

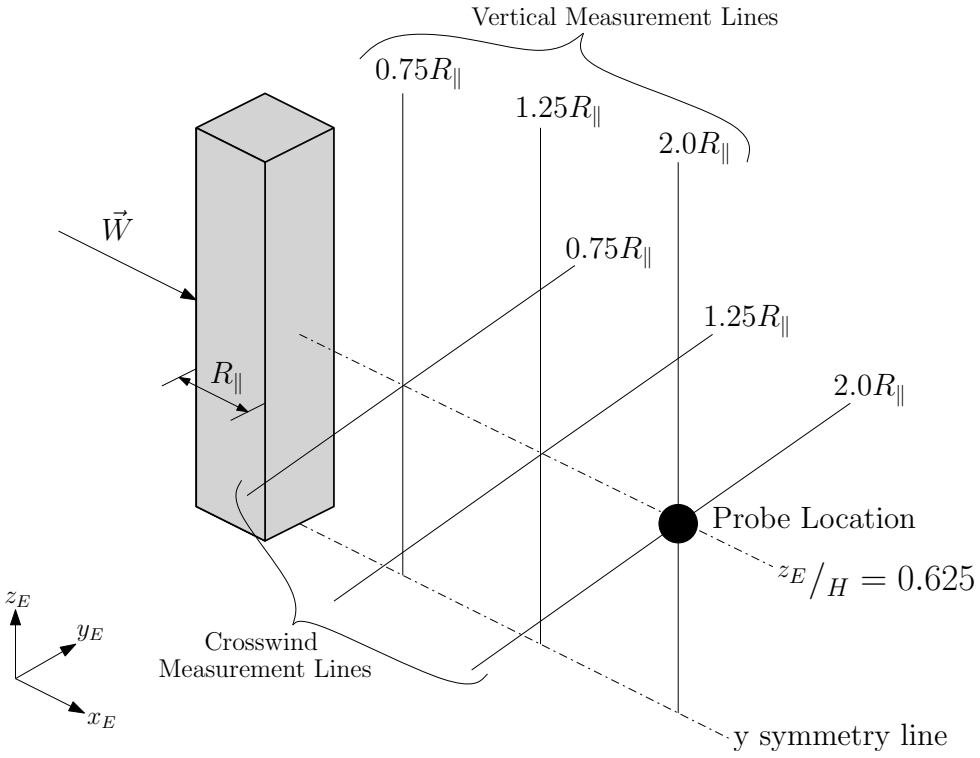


Figure 19: Data sample and plotting locations

3.2.1 Q-Criterion Turbulence Visualization

The concept of a vortex is intuitive however a formal definition, and visualization methods, are not as straightforward. A study by Jeong and Hussain [72] show the downfall of using either a local pressure minimum, streamlines, or vorticity magnitude as measure for a vortex (also see de Villers [55]). The result of the study concluded the vortex core can be defined with complex eigenvalues of ∇V, and satisfy the characteristic equation,

$$\lambda^3 + \boldsymbol{P}\lambda^2 + \boldsymbol{Q}\lambda - \boldsymbol{R} = 0 \tag{41}$$

For incompressible flow the first invariant $\boldsymbol{P}$ is zero ($\nabla \cdot V = 0$) and the third invariant $\boldsymbol{R}$ is equal to the determinant of ∇V. The second invariant $\boldsymbol{Q}$ is defined as,

$$\boldsymbol{Q} = \frac{1}{2}\left(\Omega_{ij}\Omega_{ij} - S_{ij}S_{ij}\right) \tag{42}$$

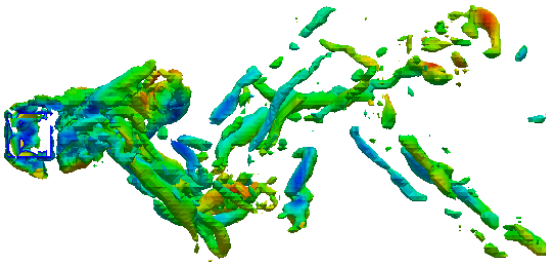
where S_{ij} is rate-of-strain tensor and Ω_{ij} is the rate-of-rotation tensor (vorticity tensor), which are the symmetric and antisymmetric parts of ∇V respectively and defined as,

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{43a}$$

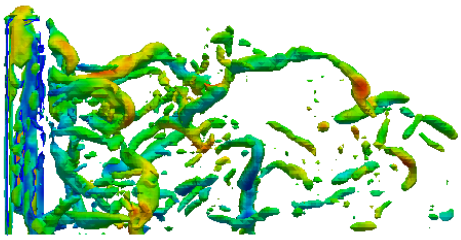
$$\Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) \tag{43b}$$

This Q-criterion, proposed by Hunt et al [73], defines a vortex as a connected fluid region with a positive second invariant of the velocity gradient tensor ∇V [74]. Since when $\boldsymbol{Q}$ is positive it represents locations in the flow were the rotation dominates the strain and shear [73–75]. While not a measure of grid quality the Q-criterion provides

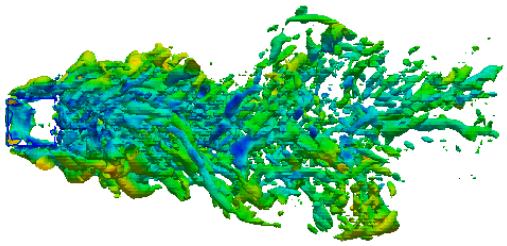
the ability to visualize vortical structures as isosurfaces as shown in Figure 20. As the grid spacing is successively reduced, finer and smaller vortical structures are captured resulting in an increase in the amount of directly resolved vortical structures. This visualization provides an initial estimate that the coarse grid has inadequate density resolving the desired level of turbulent motions.



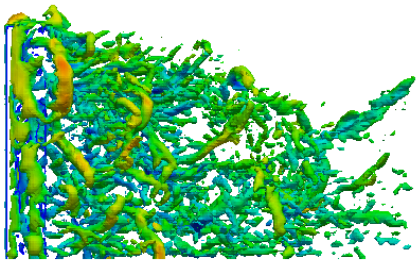
(a) Coarse mesh, top view



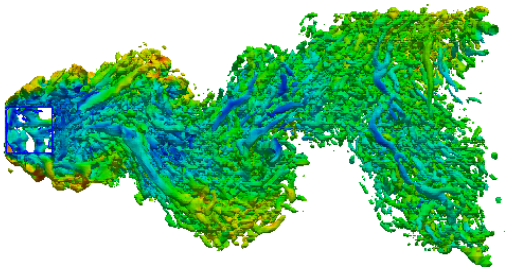
(b) Coarse mesh, side view



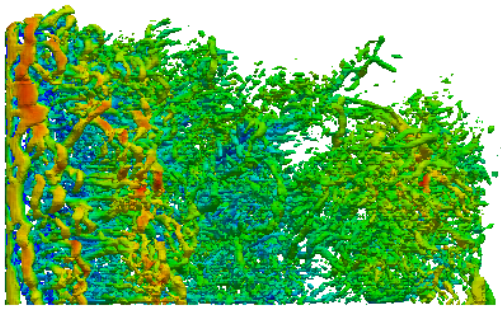
(c) Medium mesh, top view



(d) Medium mesh, side view



(e) Fine mesh, top view



(f) Fine mesh, side view

Figure 20: Q-criterion vortex visualization with $Q = 200$ isosurfaces, t=30 s.

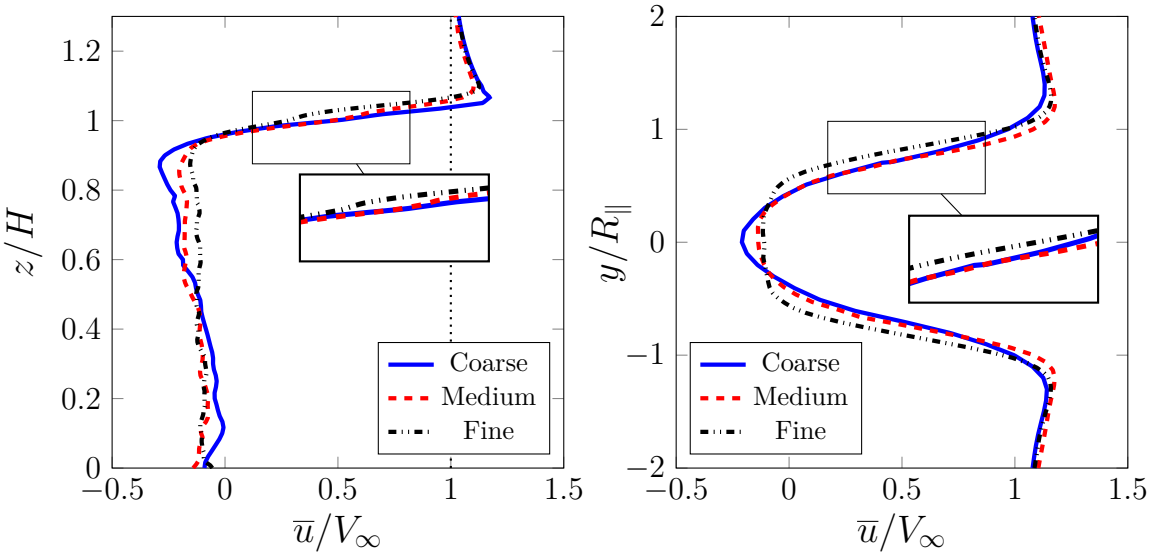
3.2.2 Time Averaged Velocity Profile

The traditional single-grid estimator of comparing the time averaged velocity profiles is first used to study the change in numerical results across the changing grid size. The velocity profiles are compared for each grid density along the six measurement lines outlined in Figure 19.

The velocities along the vertical measurement lines at $x/R_{\|} = 0.75$ downstream of the building, the results shown in Figure 21a, illustrate a slightly larger average difference between the coarse and medium grids (an average difference of 4.51%) than between the medium and fine grids (an average difference of 3.81%). While all three grid densities resolve similar velocity magnitudes in the wake of the building, $z/H < 1$, the fine grid predicts a change in velocity across the upper edge of the wake (between approximately $z/H = 0.9$ and $z/H = 1.0$ as highlighted by the insert in Figure 21a) of 17.3 (m/s)/m while the same slope is 22.2 (m/s)/m and 29.4 (m/s)/m when using the medium and coarse grids respectively.

In the direction parallel to the ground passing through the wake at a height of 62.5% of the building height (i.e. $z/H = 0.625$) the results from all three grids are in close agreement (Figure 27b). The percent difference in peak velocity at the center of the wake between the coarse and medium grid is 34% and 15.7% between the medium and fine mesh. The gradients at the edge of the wake, highlighted by the insert in Figure 21b, are also in close agreement between the medium and fine grids where a value of 28.6 (m/s)/m is predicted using the medium grid and 30.0 (m/s)/m is predicted using the fine grid, a difference of 4.9%. In this case, the coarse grid

again shows more of a discrepancy in that the velocity gradient at the wake edge is predicted to be 26.6 (m/s)/m a difference of 7.5% from the medium mesh.

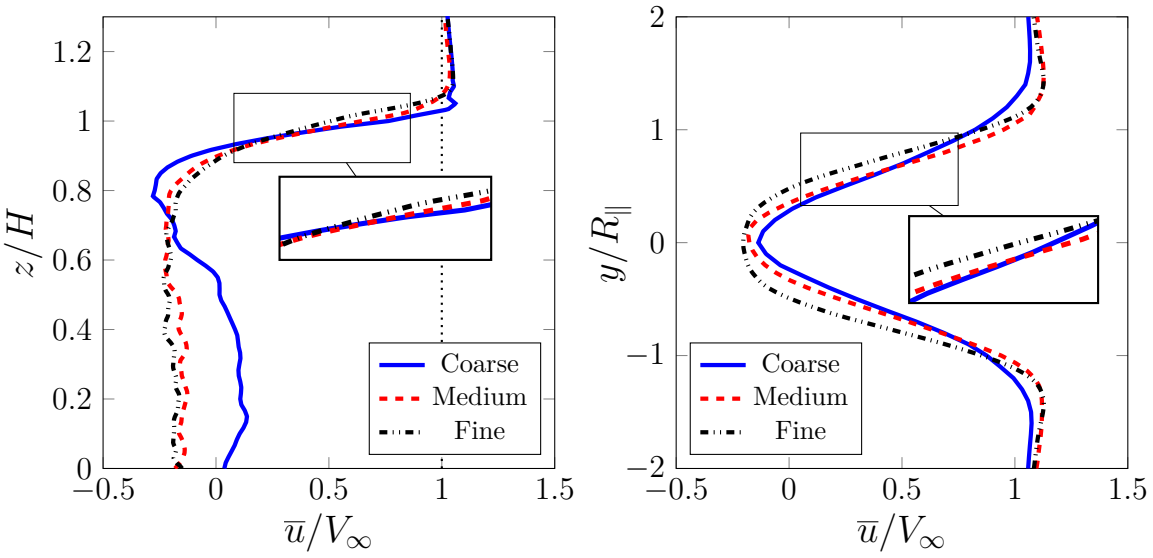


(a) Vertical measurement line **(b)** Crosswind measurement line

Figure 21: Time averaged streamwise velocity profiles at a distance $x/R_\parallel = 0.75$ downstream of the building

Along the vertical measurement line $x/R_\parallel = 1.25$ downstream of the building, similar trends are found, as Figure 22a illustrates a larger difference between the coarse and medium grids (an average difference of 13.3%) than between the medium and fine grids (an average difference of 3.5%). Studying the wake, beneath approximately $z/H = 0.6$, it can be noted that the course grid does not predict any re-circulation except near the building rooftop, whereas both the medium and fine grids predict a re-circulation zone behind the entire height of the building as evidenced by the negative value of $\overline{u}/V_\infty$. This recirculation zone in the wake is expected from the experimental results shown in Figure 28a of Section 3.2.4. The fine grid predicts a change in velocity across the upper edge of the wake (between approximately

$z/H = 0.9$ and $z/H = 1.0$ as highlighted by the insert in Figure 22a) of 11.6 (m/s)/m while the same slope is 15.0 (m/s)/m and 20.7 (m/s)/m when using the medium and coarse grids respectively. The upper edge of the wake (as defined as the point where $\overline{u}/V_\infty = 1$) is calculated to be $z/H = 1.03$, $z/H = 1.08$, and $z/H = 1.07$ for the coarse, medium, and fine grids respectively. This is illustrated in Figure 22a with the vertical dotted line. Additionally, the coarse grid has a slight oscillation in this region unseen in both the medium and fine grids.



(a) Vertical measurement line **(b)** Crosswind measurement line

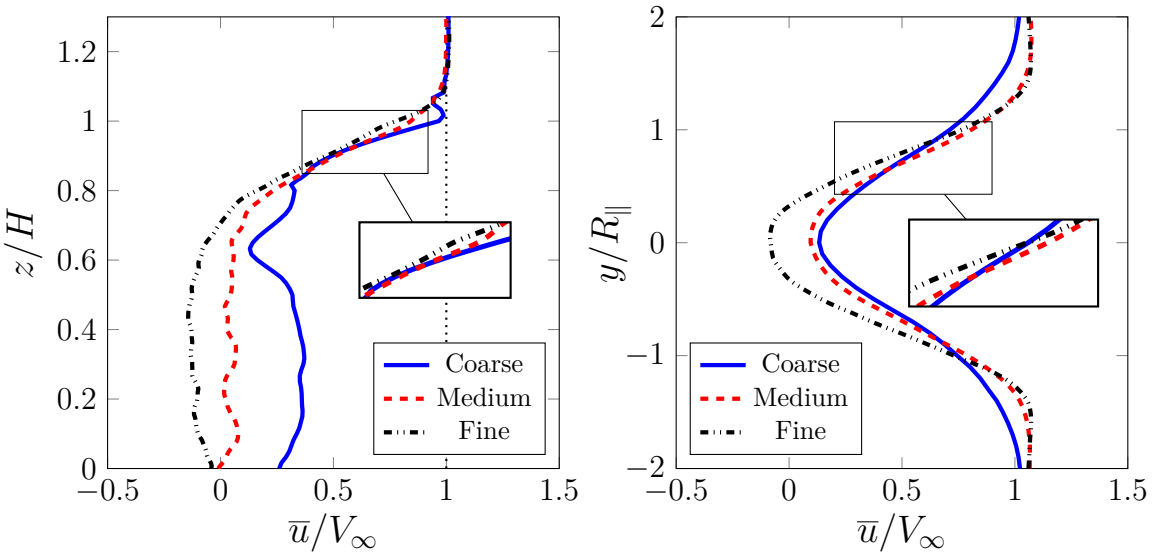
Figure 22: Time averaged streamwise velocity profiles at a distance $x/R_\parallel = 1.25$ downstream of the building

For the horizontal measurement line passing through the wake at a height of 62.5% of the building height (i.e., $z/H = 0.625$) the results from all three grids are in close agreement. The minimum velocity at the center of the wake using the medium grid is within 13.4% of that obtained using the fine grid, while the coarse grid yields a minimum that is 32.9% away from the fine grid. The gradients at the edge of the

wake, highlighted by the insert in Figure 22b, are also in close agreement between the medium and fine grids where a value of 20.7 (m/s)/m is predicted using the medium grid and 22.1 (m/s)/m is predicted using the fine grid, a difference of 6.8%. In this case, the coarse grid again shows more of a discrepancy in that the velocity gradient at the wake edge is predicted to be 15.8 (m/s)/m which is 23.7% below the medium grid result.

When the same comparison is applied to the vertical measurement line $x/R_{\|} = 2.00$ downstream it is seen from Figure 23a that the course grid greatly under predicts the velocity and has a large oscillations around $z/H = 0.6$ and small oscillation around $z/H = 1$, both unseen in the medium and fine grids. Overall the average difference between the coarse and medium meshing is 15.6%, while between the medium and fine is 6.6%. The slops found as highlighted by the insert in Figure 23a, are 7.6 (m/s)/m, 6.8 (m/s)/m, and 6.4 (m/s)/m for the coarse, medium, and fine meshes respectively. This corresponds to a 10.5% increase from the coarse to medium grids with only 5.9% change from medium to fine.

Along the horizontal measurement line at $x/R_{\|} = 2.00$, the fine mesh resolves a much lower peak velocity with a 41% difference between the course and medium mesh in contrast to a 22% difference between the medium and fine meshes. The wake edge velocity gradients are found to be 9.3 (m/s)/m, 13.1 (m/s)/m, and 16.0 (m/s)/m for the coarse, medium, and fine grids respectively.



(a) Vertical measurement line **(b)** Crosswind measurement line

Figure 23: Time averaged streamwise velocity profiles at a distance $x/R_\parallel = 2.00$ downstream of the building

3.2.3 LES Index Quality

A second grid independence technique is performed in the form of a multi-grid estimator based on the resolved turbulent kinetic energy (TKE). A multi-grid estimator uses the results from at least two different grid densities and a Richardson extrapolation to study how the results compare to the theoretical exact solution. The LES Index Quality (LES_IQ) is a ratio of the resolved TKE to the total [69],

$$LES\_IQ_k = \frac{k^{res}}{k^{tot}} = 1 - \frac{|k^{tot} - k^{res}|}{k^{tot}} \tag{44}$$

where k^{res} is the LES resolved TKE and k^{tot} is the theoretical maximum TKE. This maximum is defined as the sum of the resolved scale and the sub-grid scale turbulence

which can be represented as,

$$k^{tot} = k^{res} + a_k h^p \tag{45}$$

where a_k is determined using a Richardson extrapolation, h is a characteristic of the grid quality, and p is the order of accuracy of the numerical scheme. For a finite volume method the characteristic grid quality h can be taken as the cube root of the cell volume. In this work the order of accuracy of the LES numerical simulations is two. The extrapolation constant between two different grid densities can be expressed as,

$$a_k = \frac{1}{h_2^p}\left[\frac{k_2^{res} - k_1^{res}}{(h_1/h_2)^p - 1}\right] \tag{46}$$

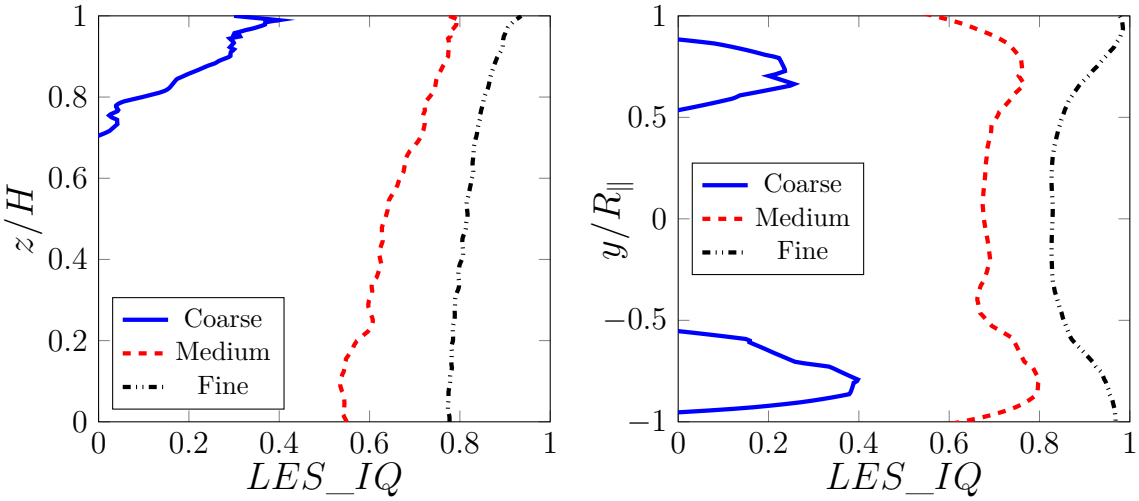
where the subscript corresponds to a grid of given density (the higher number represents a more dense grid). The index quality for each grid is then defined as,

$$LES\_IQ_k^1 = 1 - \frac{|a_k h_1^p|}{k_1^{res} + a_k h_1^p} \tag{47a}$$

$$LES\_IQ_k^2 = 1 - \frac{|a_k h_2^p|}{k_2^{res} + a_k h_2^p} \tag{47b}$$

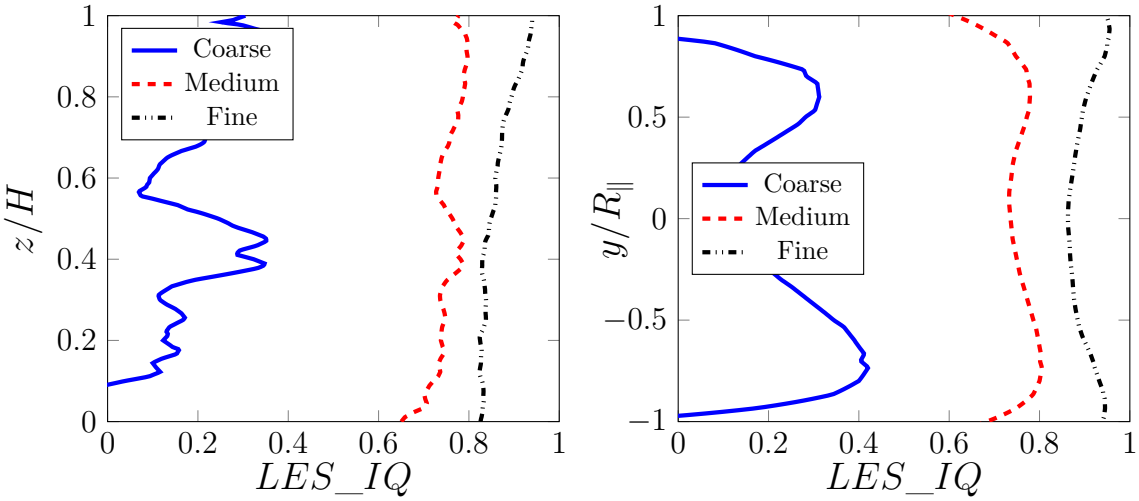
The LES Index Quality is unity (1) when all turbulent motion scales are resolved directly. However, since LES filters out the TKE at high wavenumbers this ratio is less than unity for a finite filter width. In general, when less than 20% of the TKE is approximated using the SGS model, a LES simulation is judged to be of sufficient quality [60, 69] (under these circumstances the LES_IQ would be ≥ 0.80). The calculated LES_IQ along the same horizontal and vertical measurement lines in Figure 19 are shown in Figures 24, 25, 26. From these it is seen the coarsest grid

is far too sparse to capture even 40% at most and produces unphysical extrapolation results. With a doubling of grid density the medium mesh resolves a range from 55-80%, and the fine mesh is almost always above 80% in all locations.



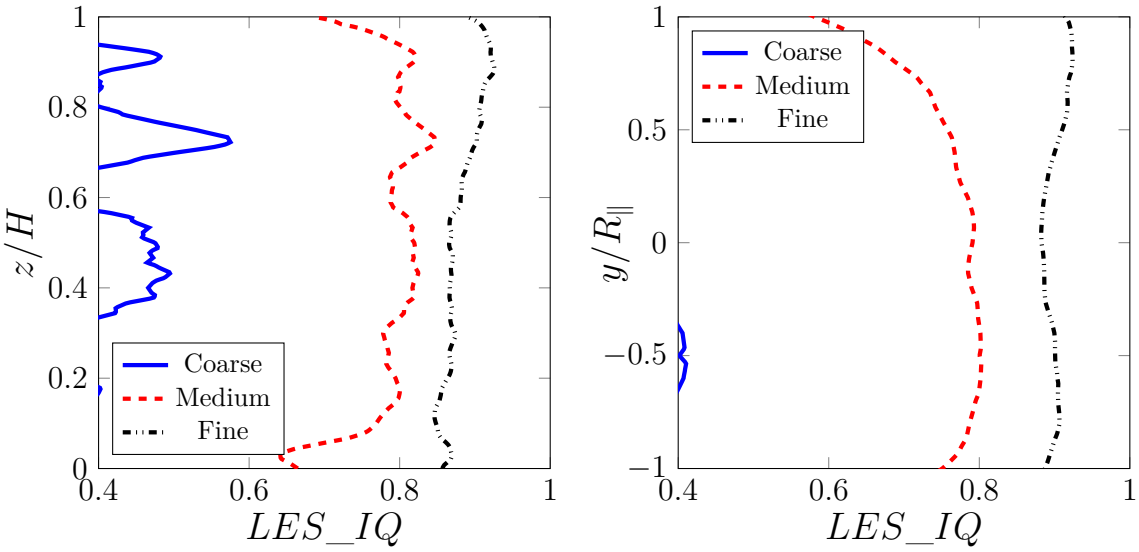
(a) Vertical measurement line **(b)** Crosswind measurement line

Figure 24: LES_IQ at $x/R_{\|} = 0.75$ downstream of the building



(a) Vertical measurement line **(b)** Crosswind measurement line

Figure 25: LES_IQ at $x/R_{\|} = 1.25$ downstream of the building

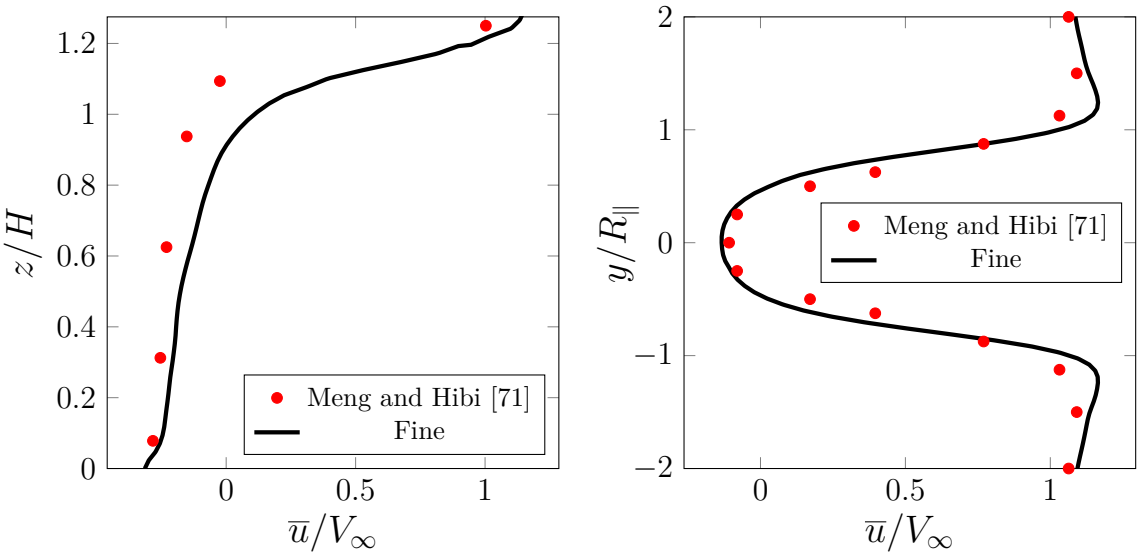


(a) Vertical measurement line **(b)** Crosswind measurement line

Figure 26: LES_IQ at $x/R_{\|} = 2.00$ downstream of the building

3.2.4 Experimental Comparison

Based on the above results of the time averaged velocity profiles and LES_IQ, the fine grid density is chosen to validate the LES simulations against the experimental results of Meng and Hibi [71]. The simulated flow conditions are modified slightly in that the building now has dimensions of 0.04 m by 0.04 m by 0.08 m (the height to characteristic length is reduced to 1.41 from the grid convergence study value of 4.24) and the Reynolds number is reduced to 1.69×10^4 based on the characteristic length D (This value corresponds to the height based Reynolds number of 2.4×10^4 used in Meng and Hibi [71]). The same time averaging process used for the grid verification study is applied to these experimental results and the extracted velocity profiles from the LES simulations at $x/R_{\|} = 0.75$ are shown in Figures 27a and 27b. The average error is found through the mean of the errors between each experimental

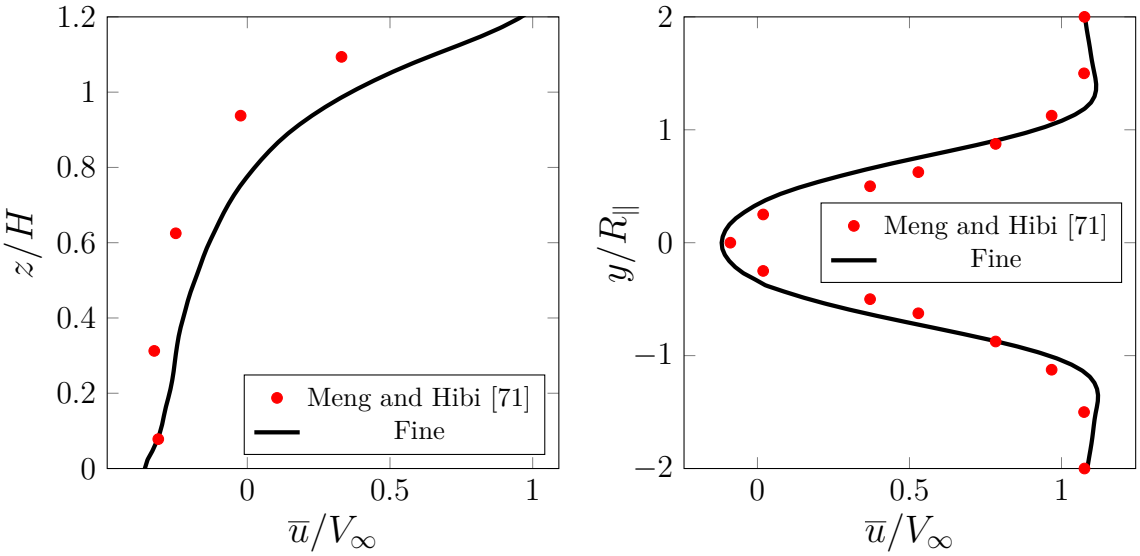


(a) Vertical measurement line

(b) Crosswind measurement line

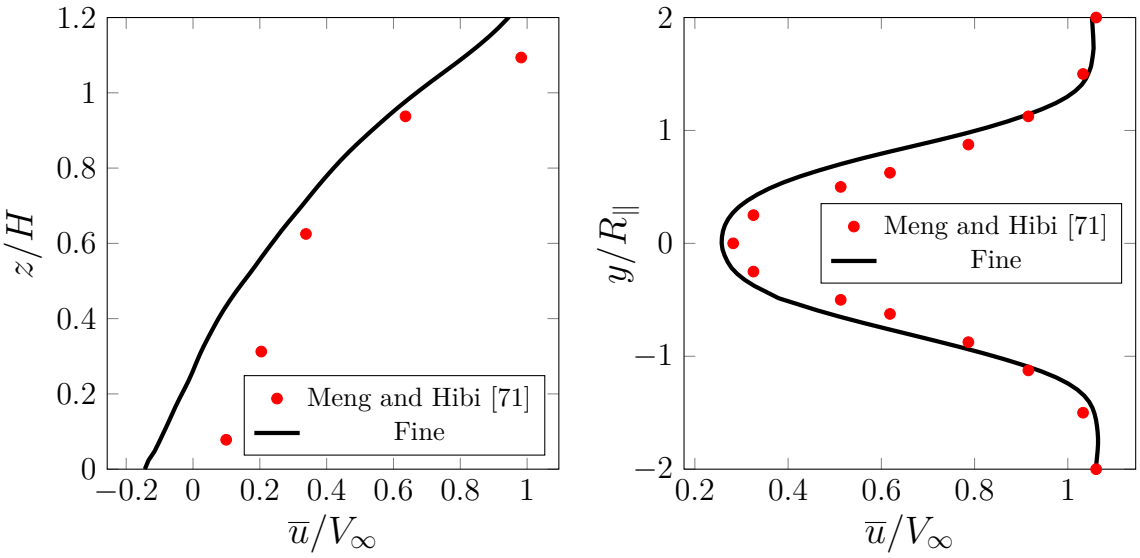
Figure 27: Experimental vs. numerical time averaged streamwise velocity comparison at a distance of $x/R_\parallel = 0.75$ downstream of the building.

point and the numerical value at the same spatial location. For the vertical and horizontal measurement lines at $x/R_\parallel = 0.75$ the average differences are 9.7% and 17.1%. Similarly the results of the measurement lines at $x/R_\parallel = 1.25$ and $x/R_\parallel = 2.00$ are shown in Figures 28, and 29. The average error between the experimental and numerical results for each respective position are 12.9%, 17.6%, 9.7%, and 17.4%. From these figures an acceptable level of agreement between the experimental and numerical results is observed where the numerical results always under predict the vertical velocity profile and over predict the horizontal profiles.



(a) Vertical measurement line **(b)** Crosswind measurement line

Figure 28: Experimental vs. numerical time averaged streamwise velocity comparison at a distance of $x/R_\parallel = 1.25$ downstream of the building.



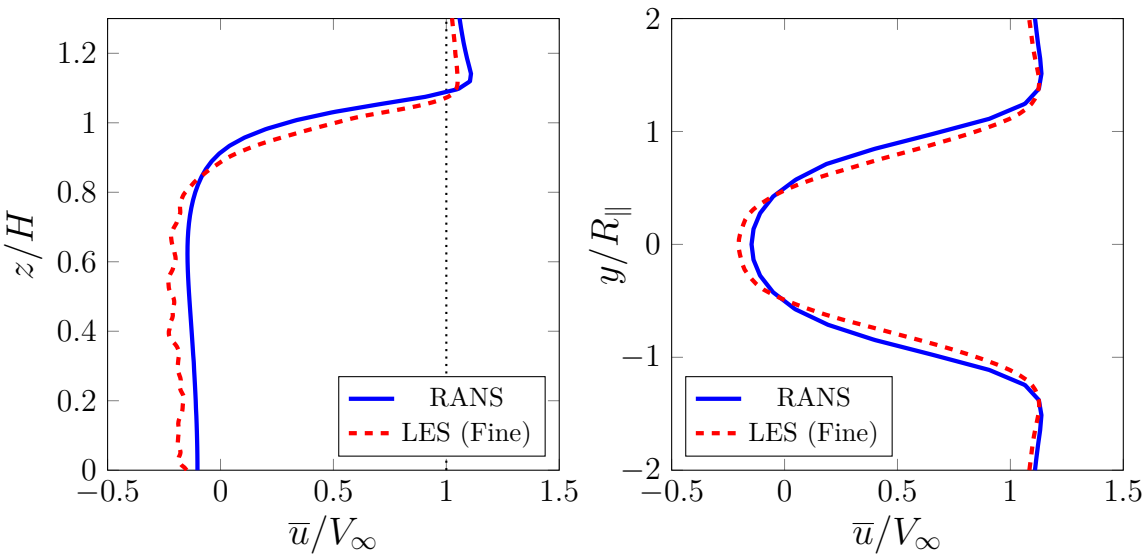
(a) Vertical measurement line **(b)** Crosswind measurement line

Figure 29: Experimental vs. numerical time averaged streamwise velocity comparison at a distance of $x/R_\parallel = 2.00$ downstream of the building.

From the vortex visualization of the Q-Criterion, the time averaged velocity profiles, and the experimental comparison, the fine grid density (with 22 cells per building width) is selected for use with the flight simulator and all future database entry CFD simulations. The coarse grid shows the largest variance between the three densities and the expected experimental results. Furthermore, from the LES_IQ it is clear the coarse grid is unacceptable in terms of LES quality. While the required computational time if five times more for the fine in comparison to the medium density, and they both exhibit similar gradients and minima velocities, the LES_IQ recommends using the fine mesh to resolve at least 80% of the TKE.

3.3 RANS Wake Field

A RANS simulation is performed to reproduce the CFX based CFD simulations of Galway et al. [19] in OpenFOAM, using the suggested gird density and turbulence model. The geometry, domain size and boundary conditions are the same as the proposed LES, as shown in Figure 17, with the $k - \epsilon$ turbulence closure model. Comparison of the RANS and time averaged LES velocity profiles at $x/R_{\|} = 1.25$ downstream of the building, shown in Figure 30, illustrate the close agreement between the simulation methods. The wake field velocities from this RANS simulation is used to evaluate the effects of a RANS versus a LES simulated wake velocities on simulated quadrotor flight, as contained in Chapter 5.



(a) Vertical measurement line

(b) Crosswind measurement line

Figure 30: RANS and time averaged LES streamwise velocity profiles at a distance $x/R_\parallel = 1.25$ downstream of the building

Chapter 4

Simulation Methodology

4.1 Urban Wake Database

As proposed by the previous work of Galway [19] the use of an urban wake database is two fold. Primarily it separates the dependence on running a CFD simulation in parallel with the flight simulator for the wake velocities. This separation is required due to the different time requirements to run either simulation, to generate the 30 seconds of wake velocities with RANS and LES methods takes approximately 0.97 and 17.7 CPU hours respectively. Furthermore, the database allows for multiple simulated urban building block configurations with only one CFD simulation through matching of the non-dimensional parameters such as the Reynolds number.

Before the simulated wake velocities are uploaded to the database, the results of the CFD simulation are processed through a series of spatial and temporal trimming scripts. The first temporal filtering is achieved through the write interval specified at run time for the CFD simulator, the resolution of this saving interval is detailed

in Section 4.1.1 below. Next the pressure and velocity components for the entire numerical domain are clipped to a specified region, taken as half a meter inward of each wake refinement boundary to minimize the effect of the refinements on the results. The final step before uploading is an additional temporal trimming in the form of finding an appropriate loop interval for the transient velocity fields given the finite length of simulated CFD time. Finding this loop interval is an iterative process and studies multiple flow results as outlined in the Section 4.1.2. A sample work flow and some database processing scripts are contained in Appendix C

4.1.1 Temporal Resolution Study

The grid refinement study in Chapter 3 resulted in the selection of the fine mesh for spatial resolution for the CFD simulations. While the time step for the LES results has been specified such that all spatial and temporal motions are resolved, a resolution study is performed to ascertain an appropriate save interval for the wake fields. Each table in the database represents a time step and each line in a table contains a location in 3D space and three wake velocity components. For the presented single building case, the CFD simulation is run for over 37,000 time steps and the wake area contains over one and a half million nodes. Therefore, the temporal resolution study is used to find a simulation time step save interval which is fine enough for the flight simulator but minimies the required database storage disk space and database querying time.

To find an acceptable time interval the results of a LES saved initially every 0.005 seconds and uploaded to the database. The database is then queried for the streamwise velocity component at the probe location shown in Figure 19 for time steps of 0.005s, 0.01s, 0.05s, 0.1s, and 0.5s (by skipping over the appropriate number of tables in the database). Figure 31 illustrates the returned streamwise velocities for a 1 second sample window.

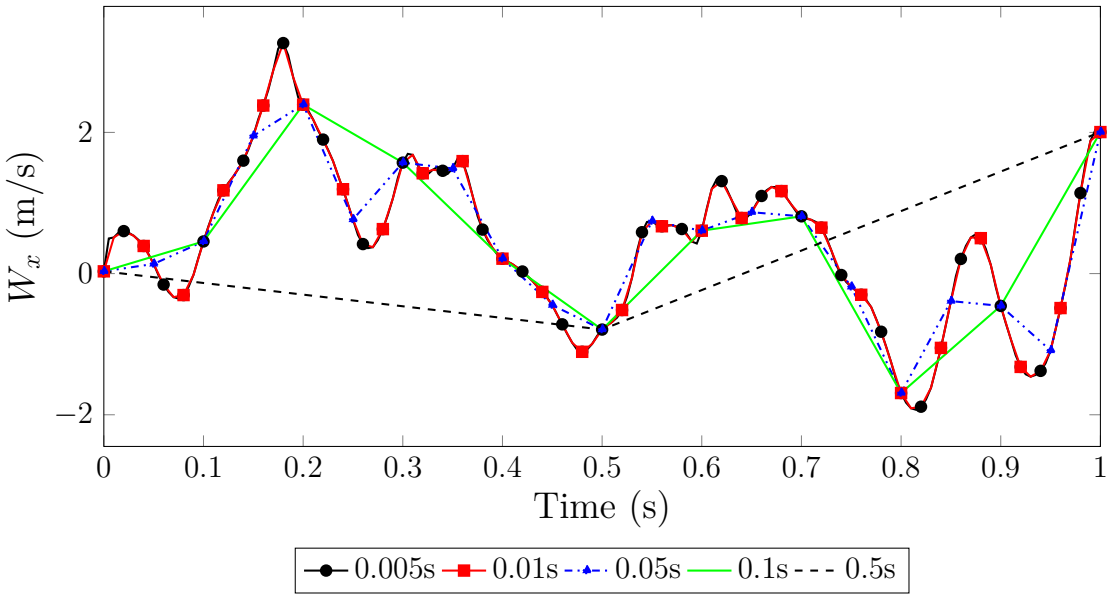


Figure 31: LES wake database timestep resolution study

From Figure 31 it is clear the 0.5s resolution is far too coarse to capture an appropriate level of velocity change. Conversely, 0.005s and 0.01s are nearly indistinguishable making the former unnecessary. To select between the remaining three resolutions the total querying times for the 1 second sampling window is considered, as collected in Table 4.

Table 4: Urban wind database query time

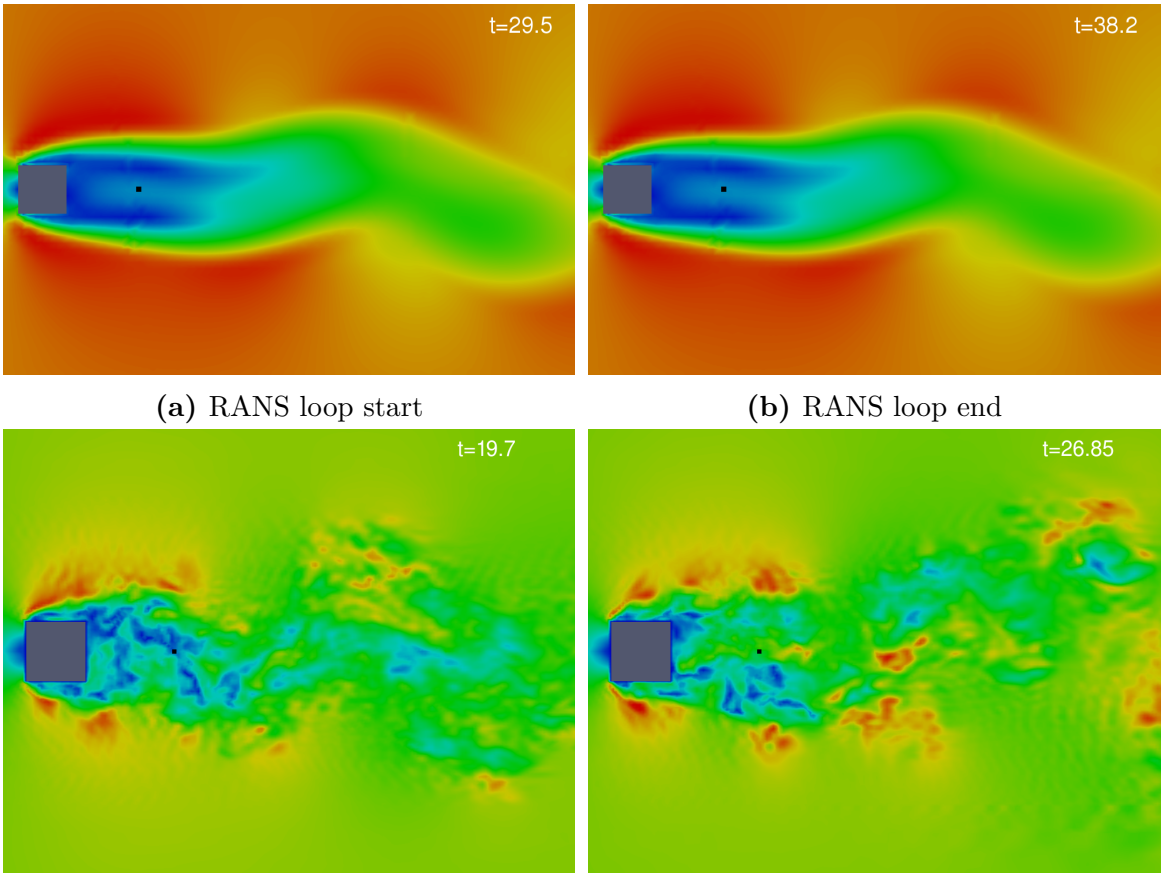
	0.005s	0.01s	0.05s	0.1s	0.5s
Query Time (s)	79.6	17.6	3.8	2.8	0.6

Since the increase in query time for 0.05s is only 36% more in comparison to 0.1s and offers double the resolution, it becomes a choice between 0.01s and 0.05s second save intervals. While 0.01 captures all of the velocity changes (peaks and troughs) it comes at the cost of a 360% increase the query time. Considering this substantial increase and the order of the missing resolution with respect to the 1 second sample window (i.e. 1 second of flight simulation), the 0.05s resolution is selected for the database save interval. selected and used for all future database generation simulations.

4.1.2 Database Loop Interval

As previously mentioned the finite end to the CFD results requires a portion of the transient wake field to be sectioned off, saved to the database, and looped over for flight simulations longer than the loop time. The database loop time for both the RANS and LES simulations are determined so that the start and end times reflect a consistent pattern within the flow field. An observation of the entire wake field velocity contours (as shown in Figure 32) is used to establish coarse start and end times.

These times are further refined by examining the side force generated on the building (Figures 33a and 33b) and the crosswind velocity component at a sample point within the wake (Figures 33c and 33d) during the time period identified by the velocity contours in Figure 32. The final start and end times of the



(a) RANS loop start

(b) RANS loop end

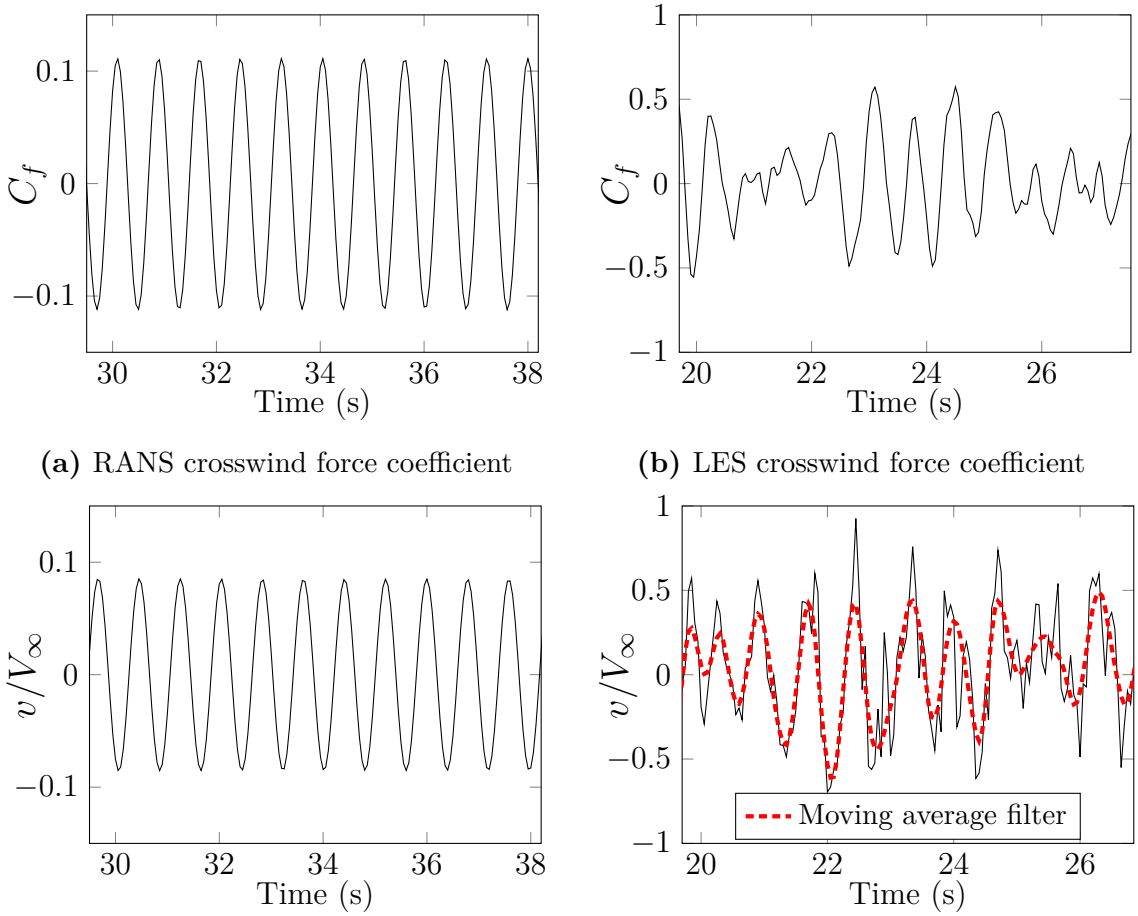
(c) LES loop start

(d) LES loop end

Figure 32: Total velocity contours for visual database loop interval estimation. The small black square illustrates the probe location (Figure 33).

database loop are modified so that crosswind velocity at the wake sample point are approximately equal at the beginning and end of the loop data (thereby minimizing any discontinuous jumps in the wind data).

The oscillations in both crosswind and building side force resulting from a periodic shedding of vortices within the building wake are clearly visible in the RANS wind data (see Figure 33a). The LES results in Figure 33b do not show the same clarity



(a) RANS crosswind force coefficient

(b) LES crosswind force coefficient

(c) RANS crosswind velocity probed at $x/R_{\|} = 2.00$, $z/H = 0.625$.

(d) LES crosswind velocity probed at $x/R_{\|} = 2.00$, $z/H = 0.625$.

Figure 33: Database loop crosswind force coefficient and probed velocity

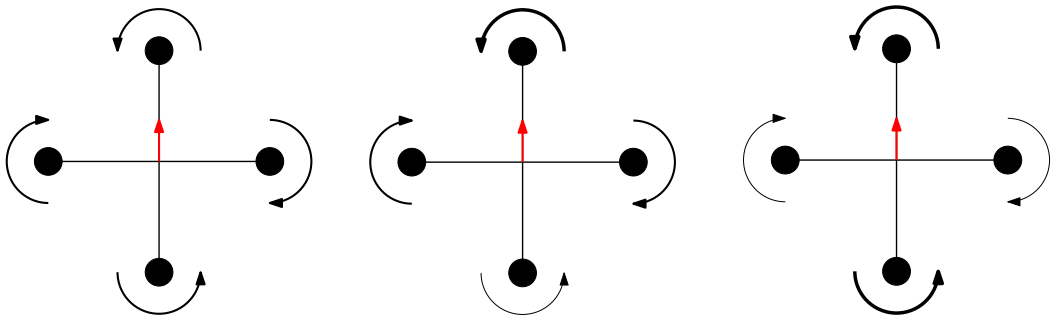
of a repeated pattern due to the greater resolution of turbulent structures which are inherently chaotic. However, applying a smoothing moving average filter to the velocity in Figure 33d produces a cleaner pattern with similar frequency to that of the RANS velocity oscillation in Figure 33c. A segment of approximately 9 seconds and 7 seconds of wind data is stored for the RANS and LES loops respectively.

4.2 Flight Controller

There are several configurations for a multi-rotor vehicle in regards to the number of motors used. As the name implies a quadrotor is a multi-rotor with four motors arranged in a plus or cross configuration. There is little difference in the performance between the two frame styles. The plus configuration is used in the flight simulator and it allows a more conceptional and clear derivation of a quadrotor's motion.

One of the benefits of a quadrotor is the simplicity of the propulsion system when compared to that of a helicopter, however this propulsion method introduces challenges for vehicle control. A quadrotor has six degrees of freedom, three positions $[x,\ y,\ z]$ and three rotations in space $[\psi,\ \theta,\ \phi]$. The control challenge arises since a quadrotor can only vary the angular velocity of the four motors and therefore results in an underactuated system. This produces a coupling between two of the six degrees of freedom. For the quadrotor the x and y translations are coupled with the pitch θ and roll ϕ angles. Figure 34 illustrates how the pair of motors on a common axis rotate in the same direction and how changing their rotation velocities is used to control the attitude and position of the quadrotor.

If all four motor rotations, and thereby thrusts, are equal and assuming no external disturbances, the quadrotor's symmetry will produce a force and moment balance resulting in a hover. To translate only along the z-axis the rotation of all four motors is either uniformly increased or decreased to ascend or descend respectively. To pitch or roll the quadrotor, thereby inducing a x-axis or y-axis



(a) Hover, Ascent, Decent **(b)** Pitch,Roll **(c)** Yaw

Figure 34: Quadrotor, plus configuration, attitude and position control

translation, the pair of off-axis motors are unchanged while one on-axis motor is increased and the other is decreased. The example in Figure 34b shows the forward motor spinning faster while the back motor being reduced which results in the quadrotor pitching up and translating along the negative x-axis. The quadrotor's yaw authority is achieved by taking advantage of the motor pair's induced torque on the airframe. By increasing the angular velocity of one pair and decreasing the other, the total thrust force is unchanged thereby preventing translation along the z-axis however there is a net torque differential about the z-axis which induces a yaw. The example in Figure 34c shows the counter-clockwise rotating motors are increased while the clockwise motors are decreased. This will cause a positive yaw angle, the quadrotor will rotate in the clockwise direction, as the torque on the body acts in the opposite direction to the motor's rotation.

To develop the equations of motion for the position and attitude of the quadrotor two coordinate systems are used, an inertial Earth fixed frame $\mathcal{F}_i$, and a body frame $\mathcal{F}_b$. The Earth frame uses a North-East-Down (NED) convention with axis notation

of $\mathcal{F}_i = \{x_E,\ y_E,\ z_E\}$, where the body frame is fixed at the quadrotor's center of gravity and follows the convention of Etkin [76] such that $\mathcal{F}_B = \{x_B,\ y_B,\ z_B\}$ are aligned out the front, right and down respectively. Figure 35 illustrates a simplified quadrotor and the two coordinate frames.

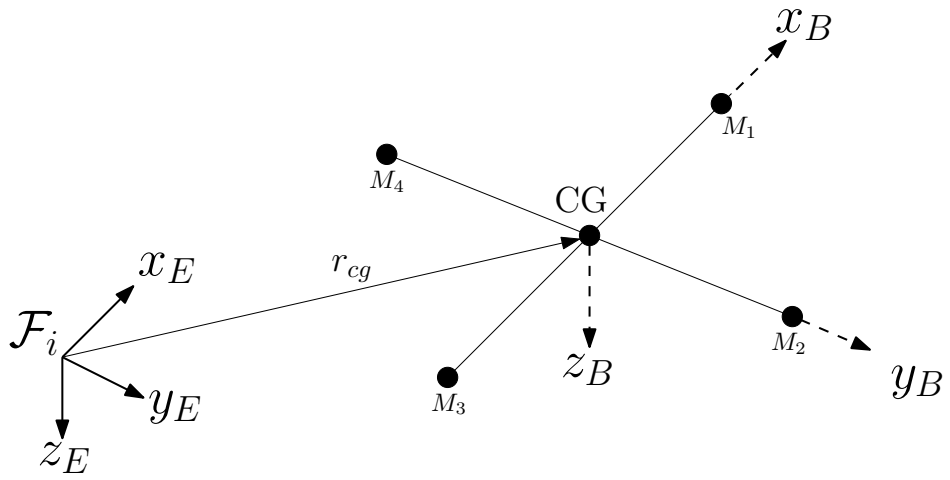


Figure 35: Inertial Earth fixed frame and body quadrotor fixed frame definitions

A series of three consecutive rotations, Euler angles [76], are used transform from one frame to the other. The order of these rotations about each axis are important and the yaw, pitch, roll or 3-2-1 order is used here to go from the Earth to the body frame,

$$L_{BE} = L_1(\phi)L_2(\theta)L_3(\psi) \tag{48}$$

Where the sequence of the angles is opposite to that of the rotations due to matrix premultiplication [76]. Conversely, to find the body to Earth rotation matrix the reverse rotation sequence is performed,

$$L_{EB} = L_3(-\psi)L_2(-\theta)L_1(-\phi) \tag{49}$$

Resulting in the body to Earth rotation matrix,

$$L_{EB} = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{50}$$

Where the angles are limited such that,

$$-\pi \leq \phi < \pi \quad \text{or} \quad 0 \leq \psi \leq 2\pi$$
$$-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$$
$$-\pi \leq \phi < \pi \quad \text{or} \quad 0 \leq \phi \leq 2\pi$$

This limiting is due to one of the constraints of Euler angles where the possibility of singularities at certain sets of angles can occur. Focusing on the quadrotor body frame the free body diagram of Figure 36 is generated.

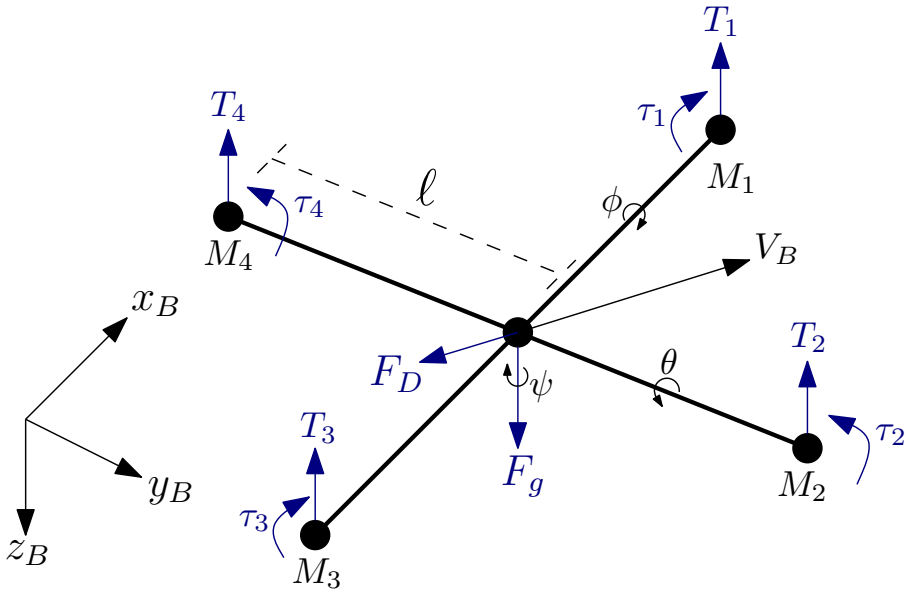


Figure 36: Quadrotor free body diagram and notation convention

Where M_i represents the ith motor and T_i and τ_i are the thrust and torque of that motor respectively. The formal illustration of the already introduced yaw, pitch and roll angles (ψ, θ, ϕ) are shown in their positive convention. External forces on the quadrotor such as gravity F_g and drag F_D are shown where the latter acts in the opposite direction to the body frame velocity or airspeed vector V_B. From this freebody diagram the governing equations of motion are derived starting with the quadrotor as a single rigid body with six degrees of freedom. The force and momentum equations relative to the inertial Earth frame are,

$$\boldsymbol{F}_E = m\dot{\boldsymbol{V}}_E \tag{51a}$$

$$\boldsymbol{M}_E = \dot{\boldsymbol{h}}_E = \frac{d}{dt}\left(\boldsymbol{I}_E \cdot \boldsymbol{\omega}\right) \tag{51b}$$

where $\boldsymbol{F}_E$ is the sum of external forces acting on the quadrotor, m is the quadrotor's mass and $\dot{\boldsymbol{V}}_E$ is the quadrotor's acceleration such that $\dot{\boldsymbol{V}}_E = \ddot{\boldsymbol{r}}_c = [\ddot{X}_E \;\; \ddot{Y}_E \;\; \ddot{Z}_E]^T$. $\boldsymbol{M}_E$ is the external moment vector about the quadrotor center of gravity, $\dot{\boldsymbol{h}}_E$ is the angular momentum, $\boldsymbol{I}_E$ is the inertia tensor and $\boldsymbol{\omega}$ is the angular velocity vector. The components of the angular velocity vector are the angular rates of the quadrotor such that,

$$\boldsymbol{\omega} = [p \quad q \quad r]^T \tag{52}$$

and the inertia matrix is defined as,

$$\boldsymbol{I}_E = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yz} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \tag{53}$$

While an inertial frame is required for valid application of Equation (51a), it results in a time derivative for the angular momentum where both $\boldsymbol{I}_E$ and $\boldsymbol{\omega}$ change with motion and become variables. Therefore the forces and moments are expressed in the body frame by applying the Euler angle rotation matrix $\boldsymbol{L}_{EB}$ defined in Equation (50). Starting with the force equation, extra care is required when using the body frame due to the body fixed velocity vector $\boldsymbol{V}_B$. Since this vector by definition rotates about $\mathcal{F}_B$, and the origin of $\mathcal{F}_B$ can rotate about $\mathcal{F}_E$, the relative motion between $\boldsymbol{V}_B$ and $\mathcal{F}_E$ causes the direction cosines in the rotation matrix to change with time.

$$\boldsymbol{L}_{EB}\boldsymbol{F}_B = m\frac{d}{dt}\left(\boldsymbol{L}_{EB}\boldsymbol{V}_B\right) = m\left(\boldsymbol{L}_{EB}\dot{\boldsymbol{V}}_B + \dot{\boldsymbol{L}}_{EB}\boldsymbol{V}_{\boldsymbol{B}}\right) \tag{54}$$

Where $\boldsymbol{V}_B = [u, v, w]^T$. The last term $\dot{\boldsymbol{L}}_{EB}\boldsymbol{V}_{\boldsymbol{B}}$ is the effect of the relative rotation. From the definition of this derivative of a transformation matrix, outlined in Appendix A.6 of Etkin [76],

$$\dot{\boldsymbol{L}}_{EB} = \boldsymbol{L}_{EB}\tilde{\boldsymbol{\omega}}_B \tag{55}$$

Where $\tilde{\boldsymbol{\omega}}_B$ is the skew-symmetric matrix of $\boldsymbol{\omega}_B$ from the definition of vector multiplication,

$$\tilde{\boldsymbol{\omega}}_B = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{56}$$

After substitution the body frame force equation becomes,

$$\boldsymbol{F}_B = m\dot{\boldsymbol{V}}_B + \tilde{\boldsymbol{\omega}}_B m\boldsymbol{V}_B \tag{57}$$

From free body diagram of Figure 36, the external forces on the quadrotor are the motor's thrust, gravity, and airframe drag such that $\boldsymbol{F}_B = \boldsymbol{F}_{T_B} + \boldsymbol{F}_{g_B} + \boldsymbol{F}_{D_B}$. Where

$\boldsymbol{F}_{T_B}$ is the total thrust force from a summation of each motor's individual thrust. It is assumed the motor and propeller plane remains orthogonal to the body frame, i.e. there is no blade flapping or elastic effects, and therefore the motor thrust always acts in the negative z-axis.

$$\boldsymbol{F}_{T_B} = \begin{bmatrix} 0 \\ 0 \\ -\sum_{i=1}^{4} T_i \end{bmatrix} \tag{58}$$

The gravitational acceleration always acts in the positive z-axis of the inertial Earth frame therefore force is transformed from the Earth to body frame using the rotation matrix of Equation (50) such that $\boldsymbol{F}_{g_B} = \boldsymbol{L}_{BE}\boldsymbol{F}_{g_E}$,

$$\boldsymbol{F}_{g_B} = mg \begin{bmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} \tag{59}$$

Where m is the quadrotor mass and g gravitational acceleration. To find the body frame drag force due to the to airframe, the body frame velocity vector is resolved into its components and the drag force is calculated for each axis,

$$\boldsymbol{F}_{D_B} = -\frac{\rho}{2}\begin{bmatrix} C_{D_x} V_B^2 S_x \\ C_{D_y} V_B^2 S_y \\ C_{D_z} V_B^2 S_z \end{bmatrix} \tag{60}$$

Where ρ is the density of the air, C_d is the airframe drag coefficient, equal to that of a cube, and S_i is the normal airframe surface area. Therefore collecting Equations (57), (58), (59), and (60), results in the system of equations for linear acceleration of the quadrotor in the body frame,

$$\ddot{x}_B = \dot{u} = rv - qw - g\sin\theta - \frac{F_{D_x}}{m} \tag{61a}$$

$$\ddot{y}_B = \dot{v} = pw - ru + g\cos\theta\sin\phi - \frac{F_{D_y}}{m} \tag{61b}$$

$$\ddot{z}_B = \dot{w} = qu - pv - \frac{1}{m}\sum_{i=1}^{4} T_i + g\cos\theta\cos\phi - \frac{F_{D_z}}{m} \tag{61c}$$

The next task is to study the moment balance of the quadrotor. Starting with the derived Earth frame moment in Equation (51b), the Earth to body rotation matrix is applied. The moment of inertia matrix is now in the body frame and becomes independent of time as the frame is assumed to be rigid resulting in,

$$\boldsymbol{L}_{EB}\boldsymbol{M}_B = \boldsymbol{I}_B \frac{d}{dt}\left(\boldsymbol{L}_{EB}\dot{\boldsymbol{\omega}} + \dot{\boldsymbol{L}}_{EB}\boldsymbol{\omega}\right) \tag{62}$$

Applying the definition of Equation (55) and simplifying the body frame moment equation becomes,

$$\boldsymbol{M}_B = \boldsymbol{I}_B\dot{\boldsymbol{\omega}}_B + \tilde{\boldsymbol{\omega}}_B\boldsymbol{I}_B\boldsymbol{\omega}_B \tag{63}$$

Where $\boldsymbol{I}_B$ is the moment of inertia matrix for the rigid body quadrotor. The off

diagonal terms of the inertia tensor due to imbalances in the mass distribution are negligible,

$$\boldsymbol{I}_B = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{64}$$

The sum of the moments on the quadrotor are a function of the angular velocity from the motor's thrust and the gyroscopic torque induced on the airframe from the rotating blades such that $M_B = M_{T_B} + M_{G_B}$. However, the moment induced by the gyroscopic effect is negligible as it is an order of magnitude lower then the moment induced by the motor thrust. The moment generated by the ith motor is proportional to the thrust force and arm length between the motor and the center of gravity. It is illustrated in Figures 34 and 36 how each pair of motors contributes to a pitching, rolling or yawing moment. Formally this takes the form of,

$$\boldsymbol{M}_{T_B} = \begin{bmatrix} \ell\left(T_1 - T_3\right) \\ \ell\left(T_4 - T_2\right) \\ T_1 + T_3 - T_2 - T_4 \end{bmatrix} \tag{65}$$

Collecting Equations (63), (64), and (65), results in the system of equations for angular acceleration of the ideal plus configuration quadrotor in the body frame,

$$\dot{p} = \frac{1}{I_{xx}}\left[\ell\left(T_1 - T_3\right) + qr\left(I_{yy} - I_{zz}\right)\right] \tag{66a}$$

$$\dot{q} = \frac{1}{I_{yy}}\left[\ell\left(T_4 - T_2\right) + pr\left(I_{zz} - I_{xx}\right)\right] \tag{66b}$$

$$\dot{r} = \frac{1}{I_{zz}}\left[T_1 + T_3 - T_2 - T_4 + pq\left(I_{xx} - I_{yy}\right)\right] \tag{66c}$$

With the quadrotor's dynamic equations defined the kinematics are considered

and their equations outlined. By definition kinematics is used to describe the geometrically possible motion of a body without considering the forces and moments causing the motion. In the classic sense, kinematics provides the body velocity, both linear and angular, through the time derivative of the positions.

From the previous force and moment equations and for position control, it is evident a relation to calculate the inertial position and velocity from known body frame variables is required. For angular velocity this is achieved since Euler angles are not constant with time and therefore a relationship between the Euler angle rates and the body frame rates is formed. Using the definition of unit vectors a rotation matrix for the angular velocity is found,

$$\boldsymbol{T} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \tag{67}$$

Which produces the angular velocity kinematic equations,

$$\dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \tag{68a}$$

$$\dot{\theta} = q\cos\phi - r\sin\phi \tag{68b}$$

$$\dot{\psi} = q\sin\phi\sec\theta + r\cos\phi\sec\theta \tag{68c}$$

The final set of equations required are the linear velocity of the quadrotor in the inertial frame and this is where the effect of the urban wind is incorporated on the quadrotor's flight. The wind velocity vector is rotated into the body frame and it's

components are summed with the current body frame velocities to a create a new velocity vector such that,

$$u = u + W_{x_B} = \dot{X}_B \tag{69a}$$

$$v = w + W_{y_B} = \dot{Y}_B \tag{69b}$$

$$w = w + W_{z_B} = \dot{Z}_B \tag{69c}$$

where a positive value of wind results in an increase in the vehicle airspeed (hence the wind velocity is taken positive when it acts along the negative direction of the body frame axes). This updated velocity is then also used to find the quadrotor's linear velocity in the inertial frame by applying the body to Earth rotation matrix Equation (50),

$$\dot{X}_E = \dot{X}_B \cos\theta \cos\psi + \dot{Y}_B(\sin\phi \sin\theta \cos\psi - \cos\phi \sin\psi) + \dot{Z}_B(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi) \tag{70a}$$

$$\dot{Y}_E = \dot{X}_B \cos\theta \sin\psi + \dot{Y}_B(\sin\phi \sin\theta \sin\psi + \cos\phi \cos\psi) + \dot{Z}_B(\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi) \tag{70b}$$

$$\dot{Z}_E = -\dot{X}_B \sin\theta + \dot{Y}_B \sin\phi \cos\theta + \dot{Z}_B \cos\phi \cos\theta \tag{70c}$$

To close the system of twelve equations quadrotor specific constants and experimentally found values are used, shown below in Table 5. The quadrotor has dimensions of approximately 430 mm by 430 mm by 125 mm and a mass of 2.2 kg. The central structure of the body/fuselage is cubic prism for a simple drag coefficient estimation and convenient construction techniques.

Table 5: Quadrotor flight simulator parameters

Mass	(m)	2.2 kg
Arm Length	(ℓ)	0.215 m
Gravity	(g)	9.81 m/s^2
Air Density	(ρ)	1.225 kg/m^3
Moment of Inertia	(I_{xx})	1.3894×10^{-2} $kg \cdot m^2$
	(I_{yy})	1.2621×10^{-2} $kg \cdot m^2$
	(I_{zz})	2.0518×10^{-2} $kg \cdot m^2$
Drag Coefficient	(C_{D_x})	1.05
	(C_{D_y})	1.05
	(C_{D_z})	1.05
Body Area	(S_x)	0.0136 m^2
	(S_y)	0.0136 m^2
	(S_z)	0.0256 m^2

To achieve autonomous attitude and position control a cascaded control scheme using a PID controller for the attitude and PD controller for the position is implemented as illustrated in Figure 37,

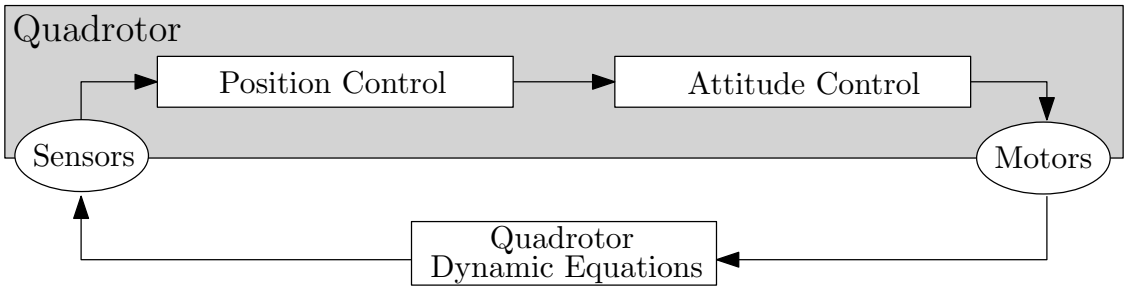


Figure 37: Quadrotor cascade attitude and position control method

Due to the varied number of methods available for a quadrotor to determine its position (GPS, sonar, LIDAR, etc.) no particular sensor error is modelled in that the controller accepts its position directly from the quadrotor dynamics as its true position in space. Based on this position information and the desired location, a commanded attitude is determined using a PD controller,

$$\theta_{des} = K_{P_X}(X_{des} - X) + K_{D_X}(\dot{X}_{des} - \dot{X}) \tag{71a}$$

$$\phi_{des} = K_{P_Y}(Y_{des} - Y) + K_{D_Y}(\dot{Y}_{des} - \dot{Y}) \tag{71b}$$

This desired attitude is then compared to the current attitude as determined from the data fusion of a magnetometer and a 6-axis inertial measurement unit (with modelled sensor error) from the Simulink Aerospace Blockset. A PID controller is then used to minimize the error between the desired angles from the position controller and the estimated orientation,

$$U_\theta = K_{P_\theta}(\theta_{des} - \theta) + K_{D_\theta}(\dot{\theta}_{des} - \dot{\theta}) + K_{I_\theta}\int_0^t (\theta_{des} - \theta) \tag{72a}$$

$$U_\phi = K_{P_\phi}(\phi_{des} - Z) + K_{D_\phi}(\dot{\phi}_{des} - \dot{\phi}) + K_{I_\psi}\int_0^t (\psi_{des} - \psi) \tag{72b}$$

$$U_\psi = K_{P_\psi}(\psi_{des} - \psi) + K_{D_\psi}(\dot{\psi}_{des} - \dot{\psi}) + K_{I_\psi}\int_0^t (\psi_{des} - \psi) \tag{72c}$$

The pitch, roll, and yaw control actions (U_θ, U_ϕ, U_ϕ) are then split among the appropriate motors according to,

$$PWM_1 = U_\theta - U_\psi \tag{73a}$$

$$PWM_2 = -U_\phi + U_\psi \tag{73b}$$

$$PWM_3 = -U_\theta - U_\psi \tag{73c}$$

$$PWM_4 = U_\phi + U_\psi \tag{73d}$$

This results in a motor specific pulse width modulation (PWM) signal which is converted into a T_i value using an experimentally determined relationship between the PWM signal and force generated by an APC 10x4.7 SF propeller and Great Planes Rimfire 400 28-30-950 out-runner motor. The gains for each of the constants shown in the two control schemes are listed in Tables 6 and 7 and remain constant for all the simulation results presented.

Table 6: Position controller PD gains

Gains:	K_{P_X}	K_{D_X}	K_{P_Y}	K_{D_Y}	K_{P_Z}	K_{D_Z}
Value:	0.2	4.0	0.2	4.0	0.4	1.0

Table 7: Attitude controller PID gains

Gains:	K_{P_ϕ}	K_{D_ϕ}	K_{I_ϕ}	K_{P_θ}	K_{D_θ}	K_{I_θ}	K_{P_ψ}	K_{D_ψ}	K_{I_ψ}
Value:	11.0	-20	0.02	11.0	-20	0.02	11.0	15	0.5

A simplified visual of the preceding control loops is shown in Figure 38. On the left side are the quadrotor's state in inertial space and the desired inertial space location (and yaw angle). After the control signal is cascaded through the controllers the resulting control actions are split and summed for the four motors. On a quadrotor an electronic speed controller (ESC) converts this signal to a desired motor RPM, producing a certain amount of thrust. The simulator directly uses an experimentally found relationship between the PWM signal and the amount of thrust force, thereby closing the loop between the governing equations of motion and simulating autonomous position control.

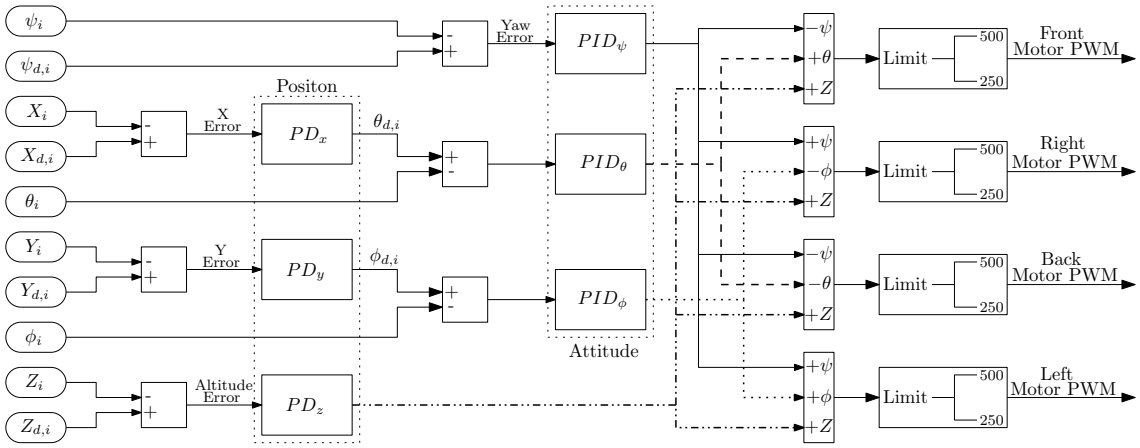
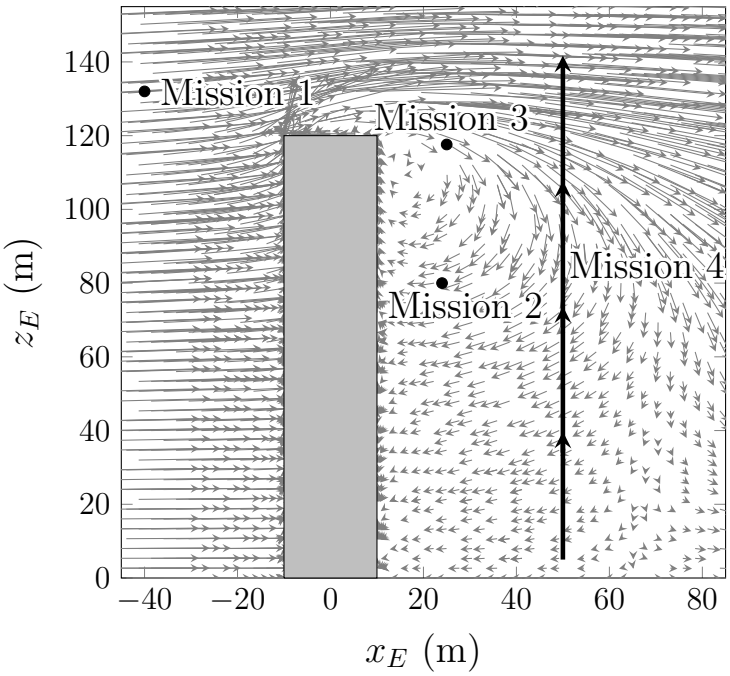


Figure 38: Quadrotor cascade PD-PID control block diagram

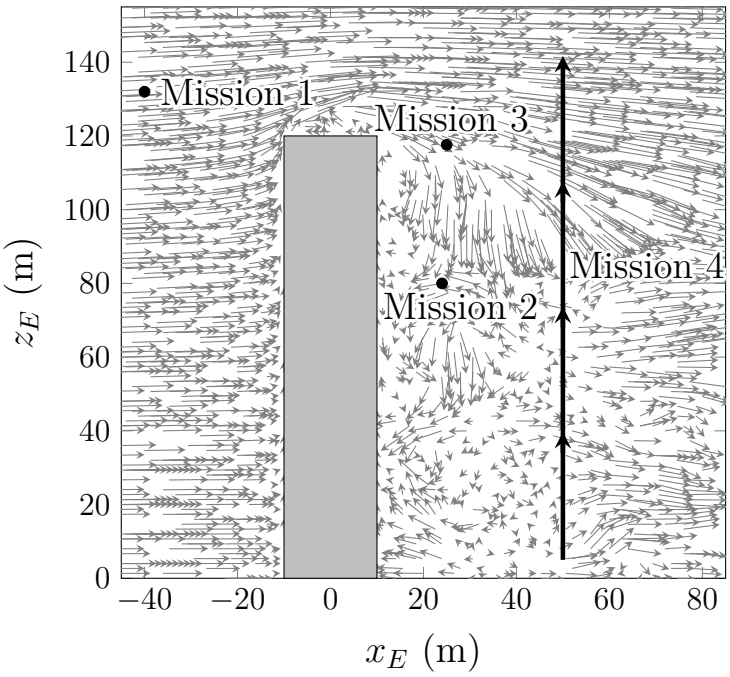
Chapter 5

Results

To evaluate the effects of a RANS versus a LES simulated wake field five flight missions are performed. Three of the missions are to hold position at specified locations. Mission 1 is in the undisturbed freestream ahead of the building to establish a vehicle reference performance. Mission 2 is fully within the building wake and represents a location that contains significant regions of re-circulating flow. Mission 3 is located along the wake boundary near the building rooftop, where deviations from this position will cause the vehicle to experience significantly varying wind patterns as a mix of the freestream and wake flow conditions. Mission 4 is a vertical ascent through the building wake with a commanded velocity of approximately 1 m/s, starting five meters off the ground in the building wake and ending in the accelerated velocity region above the roof. Finally the last mission is a horizontal translation along the y_E from freestream conditions, through the building wake, and back out to freestream flow. All these missions are shown in the x_E-y_E plane (Figure 39) and x_E-z_E plane (Figure 40) superimposed on the RANS and LES velocity field vectors.

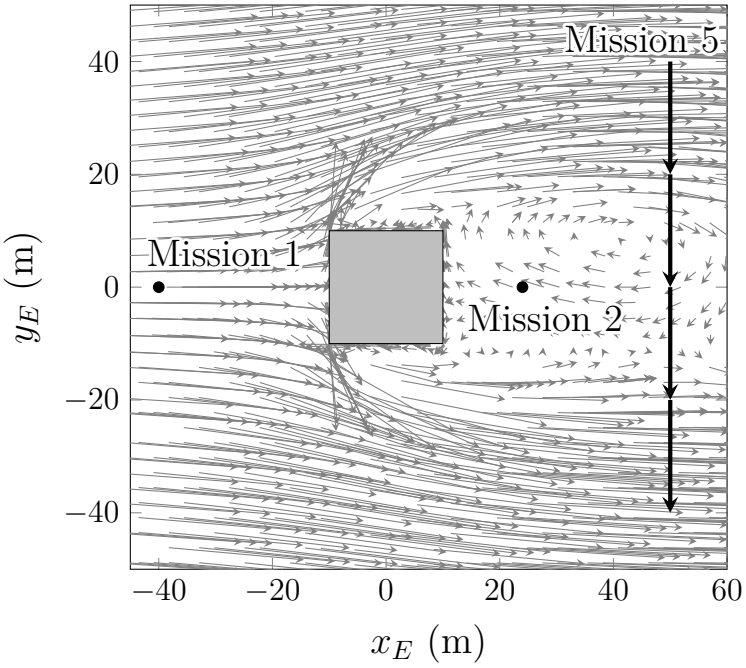


(a) RANS wake field at t=38.20s

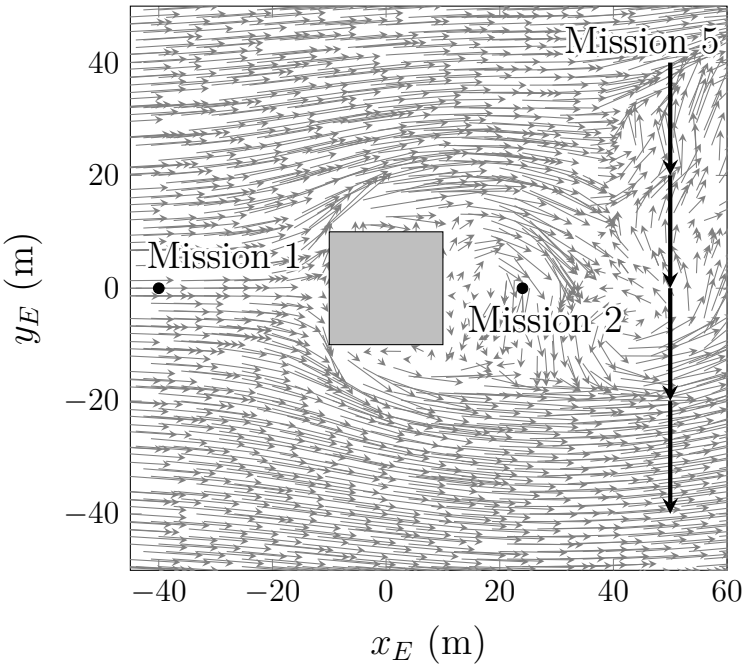


(b) LES wake field at t=26.85s

Figure 39: Flight mission locations and velocity field samples. The RANS vectors are scaled 2.2 times larger than the LES vectors for illustration.



(a) RANS wake field at t=38.20s



(b) LES wake field at t=26.85s

Figure 40: Flight mission locations and velocity field samples. The RANS vectors are scaled 2.2 times larger than the LES vectors for illustration.

From the vectors in Figures 39 and 40, not only does LES produce velocities, on average, 2 times higher than RANS, but the resulting wake field is mored random. It is evident Mission 2 should have to handle not only the turbulent motions in the wake but the movement of the bulk flow coming back down over the building roof and horseshoe vortices around the side. The crosswind motions are not as prevalent in Mission 3 at roof height, however as shown by the velocity vectors Mission 3 should experience streamwise velocities higher than freestream values as the flow accelerates over the building roof.

Mission 4 should experience a range of changing velocity components as the quadrotor ascends from the wake with all three components changing rapidly (with small magnitudes compared to the freestream) to very directional flow in the accelerated and freestream areas. Mission 5 should experience a similar change as Mission 4 but in the streamwise and crosswind directions as the quadrotor goes from the freestream, through the oscillating wake, and back into the freestream.

5.1 Mission 1 - Freestream Wind Position Hold

Mission 1 is in the freestream wind, outside of any wake affects, to compare the flight performance between: (i) a constant specified value, (ii) RANS wake field, (iii) LES wake field background wind conditions. The velocity components for the three background wind conditions are shown in Figure 41, where it is seen both RANS and LES produces a wind parallel velocity appropriately 6% less than the 4 m/s freestream due to the missions proximity to the building and energy draining effects the turbulence modeling introduces. The sharp increase in velocity from 0 m/s to 4 m/s between 5 seconds and 10 seconds is a wind ramp unintentionally simulating a large wind gust. This is done to give the controller a 5 second window to initialize and stabilize the quadrotor without any wind influence. The wind is then quickly ramped up to its full value over the next 5 seconds, which from various tests seemed to provide more realistic results in comparison to longer, shallower ramps. Furthermore, there very small oscillations in the crosswind direction and a small positive normal velocity component due to the wind starting to accelerate over the building roof.

The quadrotor's position over a 125 second hover at the desired location for Mission 1 is shown in Figure 42, separated into the three axis components. The small drop in altitude at the start of the simulation is due to the controller and motor's initializing to place the quadrotor in a hover. The sharp increase in the streamwise direction X_E is due to the wind ramp blowing the quadrotor downstream towards the building. As expected all three background wake fields produce similar position deviations of ± 0.5m from the setpoint.

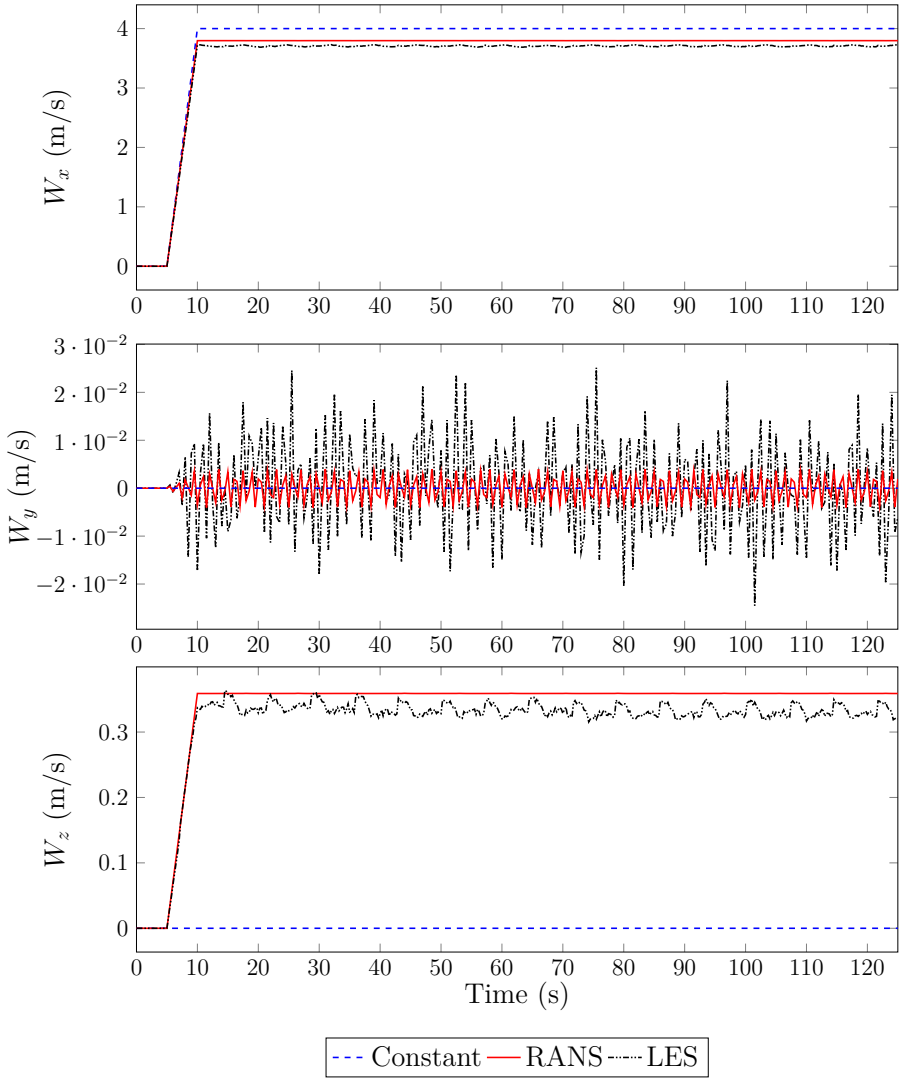


Figure 41: Mission 1 - wind velocity components

To further illustrate and quantify the position hold flight performance, deviation cuboids are formed representing the maximum quadrotor deviations from the set-point. Figure 43 shows the bounds within which the quadrotor is able to hold its position for the 125 second flight duration, where the length of the shown coordinate vectors represents a quadrotor body length along that axis. As can be seen, all three

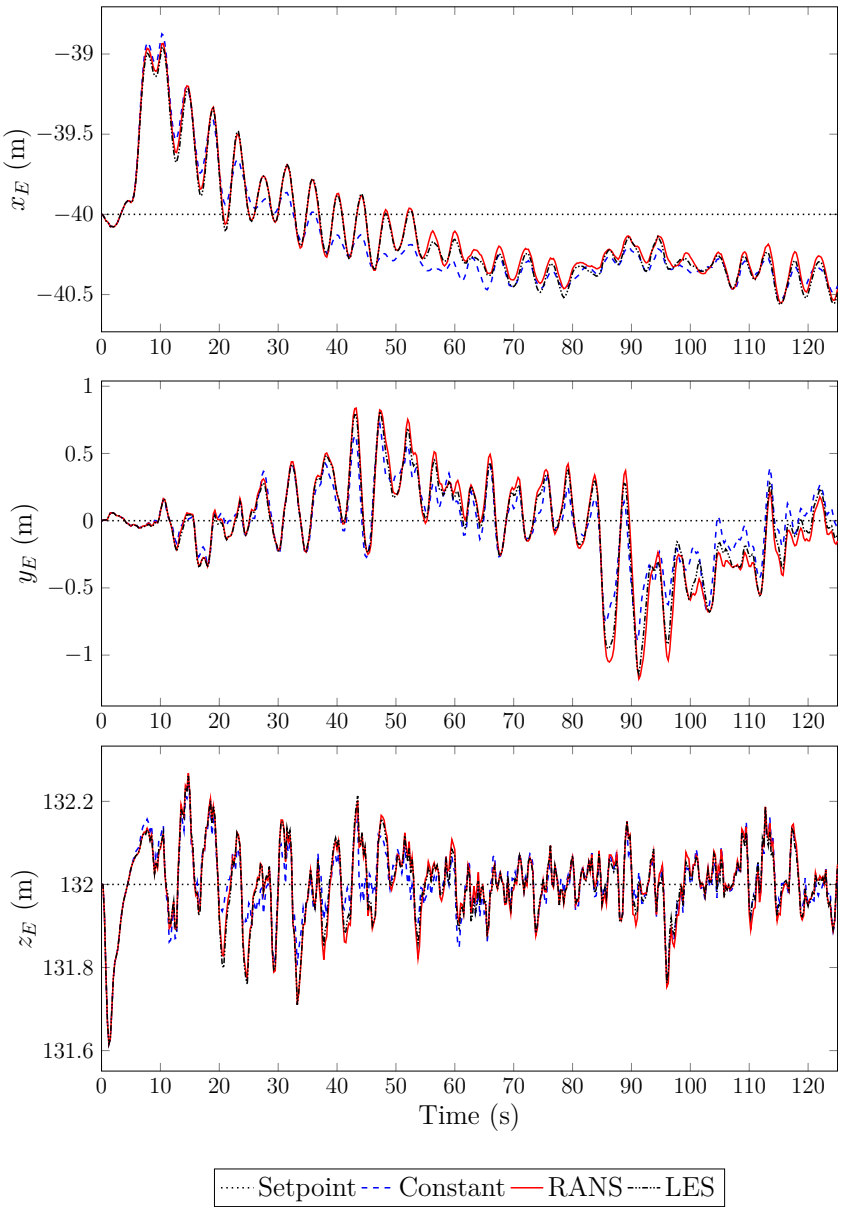


Figure 42: Mission 1 - quadrotor position

boxes are of comparable size illustrating the expected result that when flying in a freestream wind condition the simplification of a constant background is reasonably accurate.

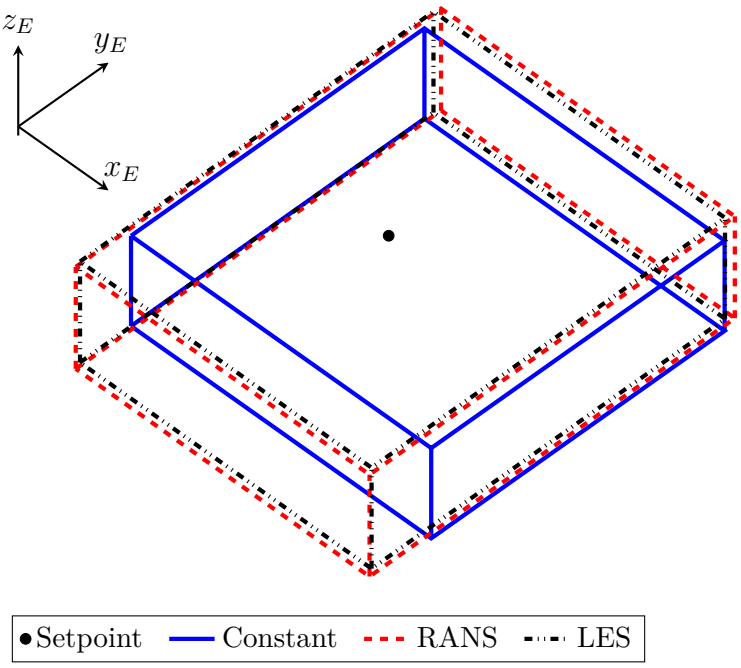


Figure 43: Mission 1 - maximum path deviation bounding boxes. The length of the shown coordinate axis represents a quadrotor body length along that axis.

The use of a RANS simulation increases the deviation volume by approximately 37% from the deviation volume observed when only a constant wind velocity is assumed, while the LES wind database increases this volume by approximately 42%. The larger increase in the LES bounding box volume is expected in that even though the position of mission one is upstream of the building wake, by virtue of the manner in which the LES upstream boundary conditions are applied, a greater degree of freestream turbulence is present. With a constant background wind the controller is able to hold the quadrotor to within $+2/-1$, ± 1.25, and ± 0.5 body lengths in the x_E, y_E, and z_E directions receptively of the desired location. This illustrates the best position hold scenario with a constant but significant wind force in combination with the simulated IMU noise.

5.2 Mission 2 - Building Wake Position Hold

Given that Mission 2 is located fully within the wake region, the temporal and spatial turbulent motions continually demand corrections from the quadrotor's controller. Fluctuations are generated by multiple vortices which begin along the sides and the roof of the building, get shed downstream, and meet within the wake. However, only the LES wake field resolves these large scale vortices while the RANS wind database averages the fluctuations as shown by the wake velocity components in Figure 44. The RANS components are smoothed from the averaging and have an overall smaller magnitude than the LES wake field velocities.

This variation in wake velocities results in a significant difference in the ability of the quadrotor to hold its position as can be seen in Figure 45. The application of the LES wake velocities generates a substantial decrease in the controllers ability to hold the quadrotor's position. At this location there is a 0.8 m/s and 1.0 m/s bias in the average velocity acting in the negative x_E and z_E directions respectively (see Figure 44) which causes the large position deviations seen in Figure 45.

From the position plots of Figure 45 it can be seen how the quadrotor is initially pushed upstream towards the building and slowly returns to a location downstream of the setpoint. Since there is a local wind bias in the negative x_E direction (towards the building) this steady state offset is produced to the lack of an integral term in PD position controller (similarity seen in Figures 42 and 48).

Generating the maximum bounding cubiods from the position components clearly

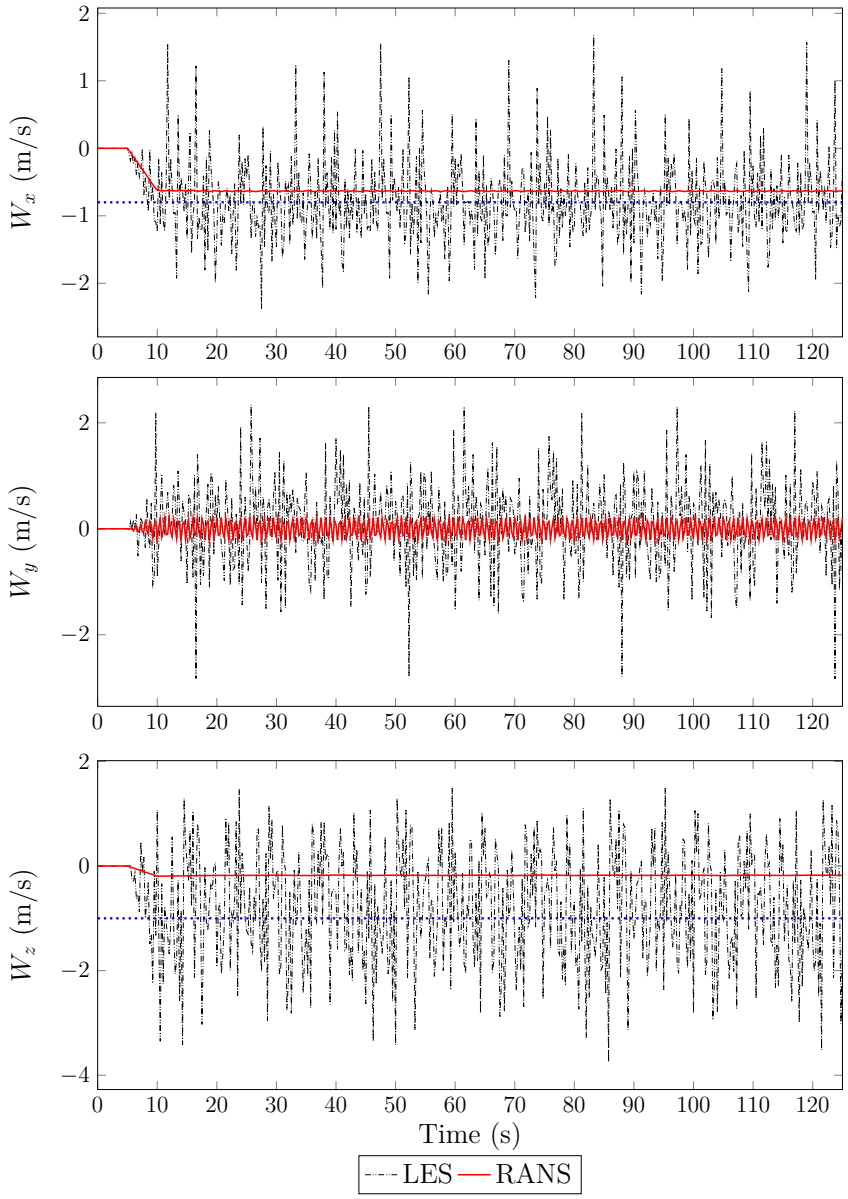


Figure 44: Mission 2 - wind velocity components

shows the resulting increase in setpoint deviation from the LES background wind, shown in Figure 46. The RANS results show a relatively small region within which the same controller is able to hold the quadrotor. Due to the small time averaged

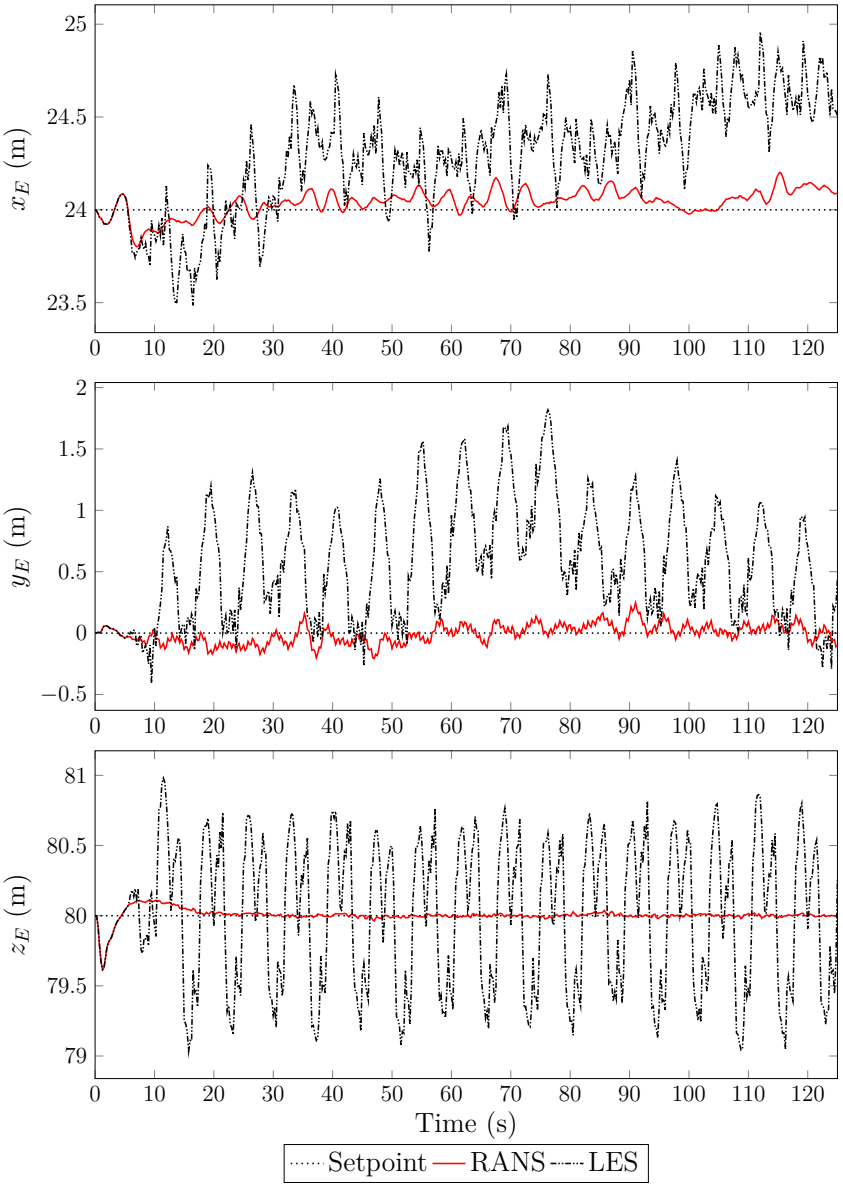


Figure 45: Mission 2 - quadrotor position

velocity in this region the controller is able to hold the quadrotor within a bounding box with a volume of approximately 0.03 m^3. This compares to a volume of 7.2 m^3 (over a two hundred fold increase) for the same vehicle/controller configuration

but with the wake field velocities now being supplied from the LES database. If the wake field field is calculated using a RANS simulation the predicted performance is that the quadrotor can be held to within ± 0.5 body lengths in all directions at this location. This is in contrast to when the wake field field is obtained from the LES database where then the simulations predict both a loss in isotropy (in that the deviations are not evenly distributed about the setpoint as shown with the relative setpoint locations on the edges of the bounding boxes) and that the absolute value of the deviations are larger. The LES simulations predict a variation of $+2/-1$, $+1.5/-0.5$, and ± 2 body lengths in the x_E, y_E and z_E directions respectively.

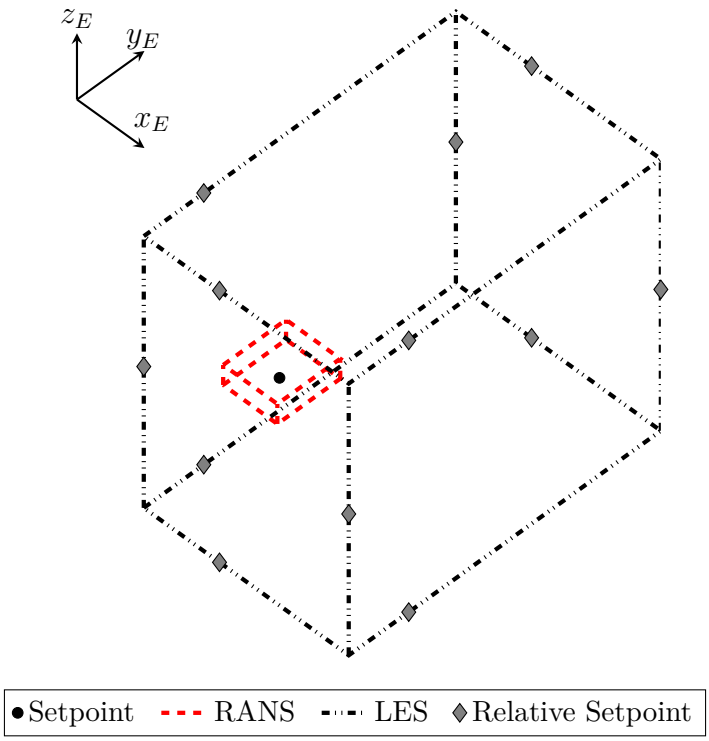


Figure 46: Mission 2 - maximum path deviation bounding boxes. The length of the shown coordinate axis represents a quadrotor body length along that axis.

5.3 Mission 3 - Top Wake Boundary Position Hold

Mission 3 is located near the top wake boundary, in the gradient area studied in Figure 22a. The RANS wake field has velocities of approximately 1 m/s, small oscillations about 0 m/s, and approximately -1 m/s in the x_E, y_E, and z_E directions respectively (Figure 47). The LES wake field velocities have double the average magnitude of the RANS wake field and have average fluctuations of ± 1 m/s on all three components, also shown in Figure 47.

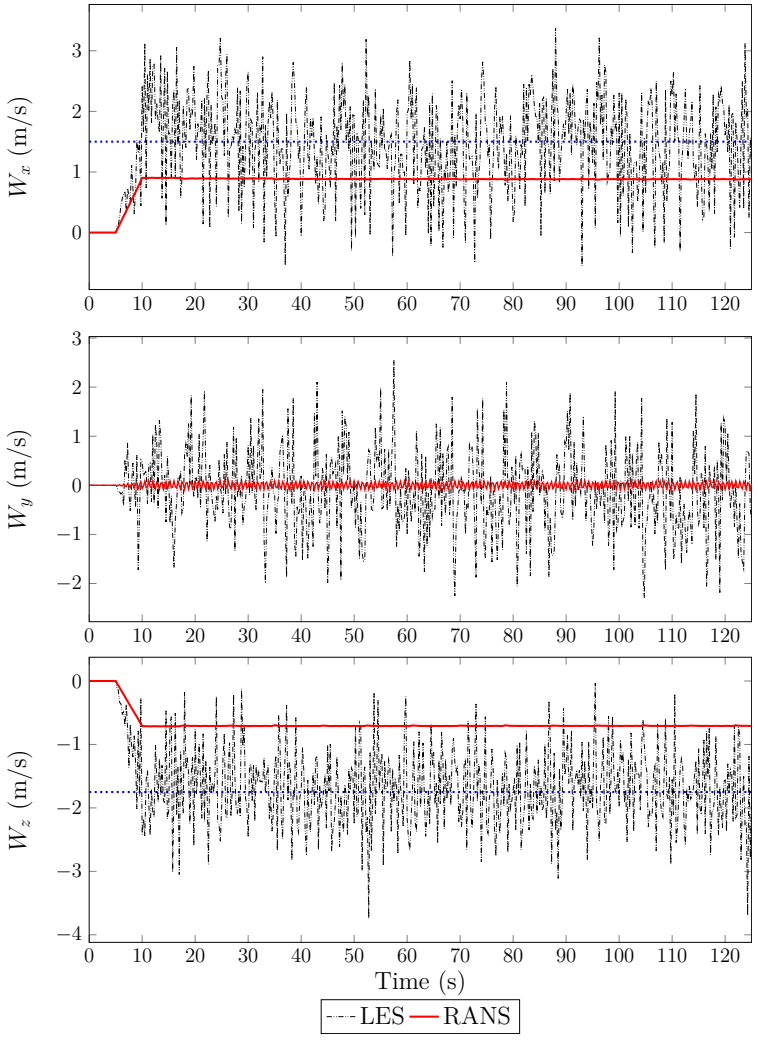


Figure 47: Mission 3 - wind velocity components

Similar to Mission 2 in the wake, the LES wake field produces much larger position deviations as illustrated in Figure 48. Additionally, the strong wind in the x_E direction results in a steady state offset upstream of the setpoint previously seen in Figures 42 and 45. While the altitude controller is able to maintain ±0.5m about the setpoint it comes at the cost of reduced y_E axis performace due to the quadrotor's dynamics coupling.

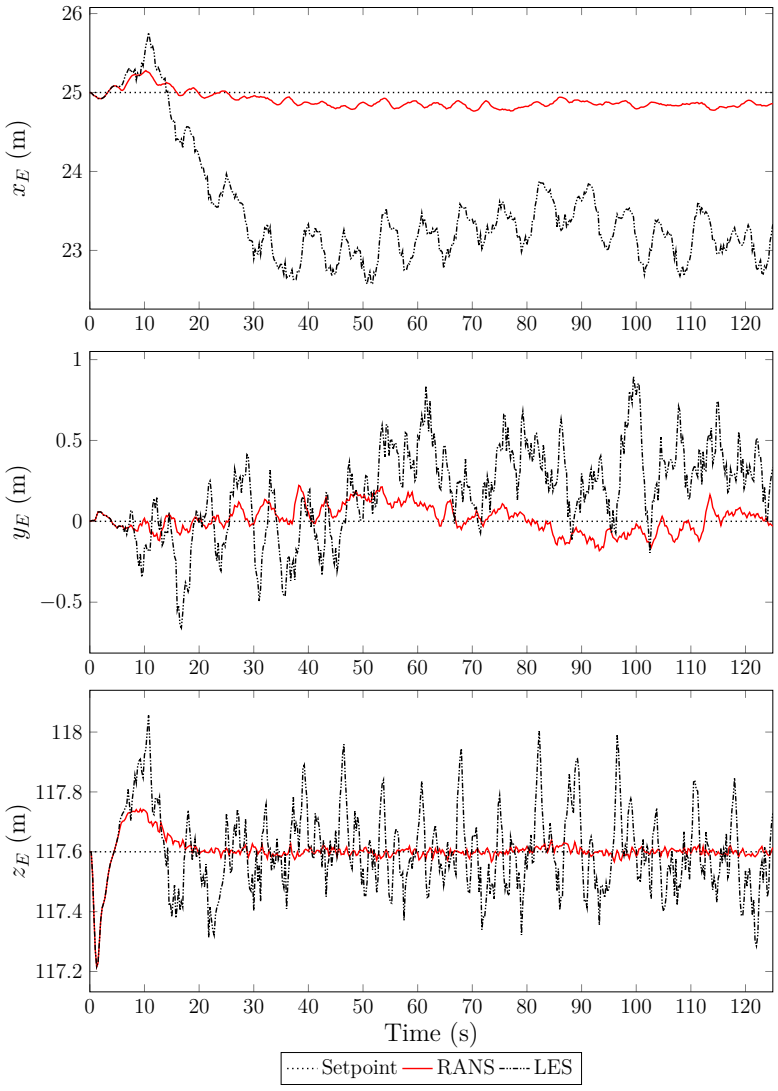


Figure 48: Mission 3 - quadrotor position

The constant offset wind in the x_E and z_E directions, coupled with the small oscillations in the y_E direction of the RANS database produces variations of only ± 0.25 body length in the x_E and y_E directions, and ± 0.1 body lengths in the z_E direction, illustrated in Figure 49. With the higher velocity magnitudes and larger turbulent fluctuations the LES database wind results in position deviations of $+2/-5$, $+0.5/-0.5$, and ± 0.5 body lengths in the x_E, y_E, and z_E directions respectively. The large steady state error in the x_E direction is clearly visible here and is the main contributor to the LES deviation volume being over 100 times larger than the RANS volume.

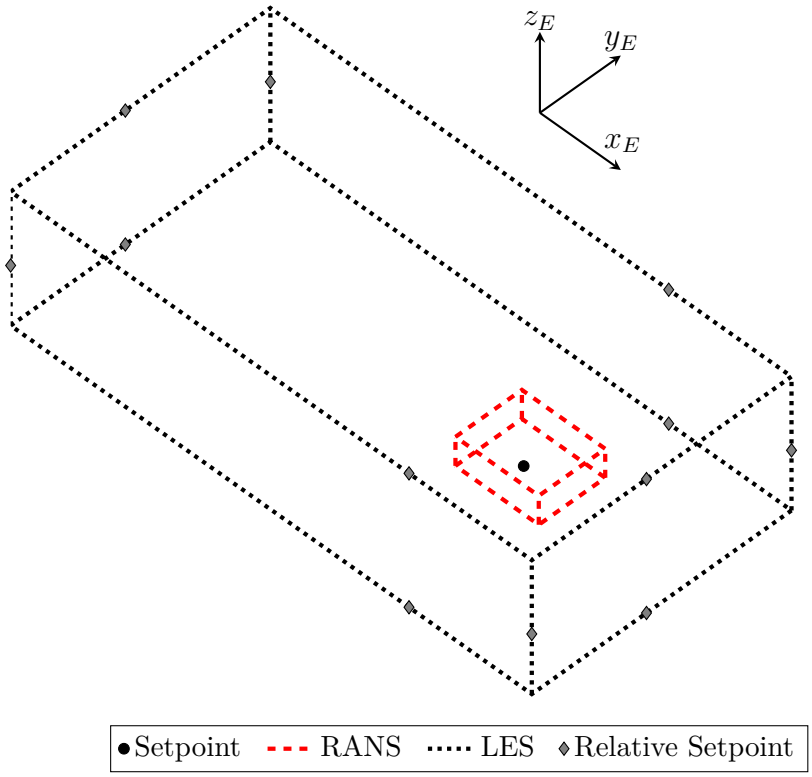


Figure 49: Mission 3 - maximum path deviation bounding boxes. The length of the shown coordinate axis represents a quadrotor body length along that axis.

5.4 Mission 4 - Ascent Though Wake

The ascent flight of Mission 4 is designed to test the controller's ability to hold its lateral position while increasing the altitude, simulating a desired situation such as building inspection. The quadrotor is initialized at a height of 5 meters and holds position for 5 seconds without background wind influence. The wind is then quickly ramped up over the next 5 seconds as the quadrotor increases altitude at approximately 1 m/s up to 135 meters in the freestream. The results for the fourth mission are shown in Figure 50 with the predicted RANS and LES flight paths shown in scale to the building size and location.

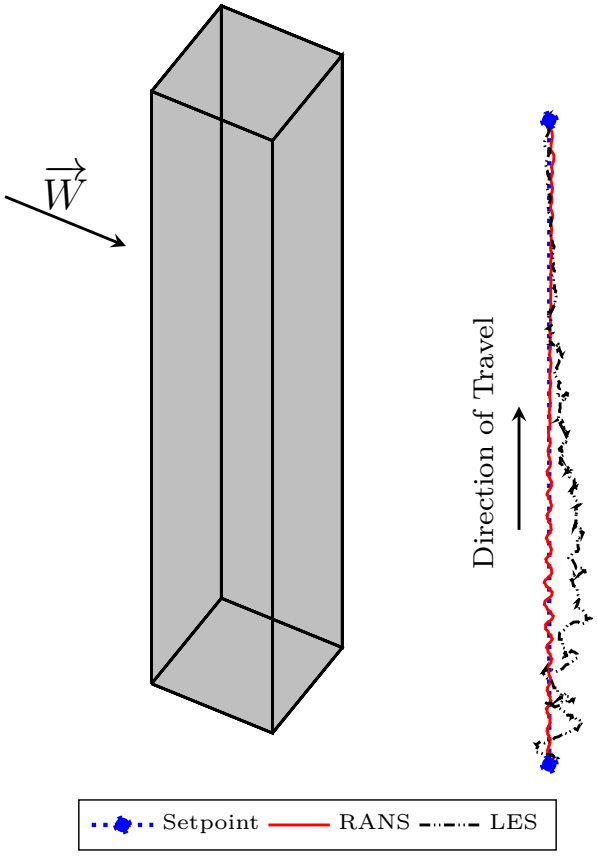


Figure 50: Mission 4 - LES deviations and building

The wake velocity components the quadrotor experiences over the flight are shown in Figure 51. The winds are now a function of both time and location as the quadrotor's position changes as the mission is flown. In the wake the RANS and LES have the same oscillations observed in the loop interval probe plots (Figures 33c and 33d). As the quadrotor ascends these crosswind component oscillations reduce to almost zero (W_y in Figure 51) while the streamwise velocity component increases close to freestream value. The vertical wind component slowly decreases as the quadrotor flies through the roof down wash as shown in Figure 39 until it returns to near zero in the freestream.

The two flight paths are also shown separated into two planes, wind parallel (x_E-z_E) and crosswind parallel (y_E-z_E) in Figures 52a and 52b to allow for a clearer visualization of the quadrotor's performance. The wind velocities deep in the building wake coupled with the constantly increasing altitude setpoint produce maximum variations of ± 1 and $+0.25/-1$ body length in the x_E and y_E directions respectively when the quadrotor is subjected to the RANS database velocities. Along the same section of flight time the LES wake velocities generate variations of $+12/-4$ and $+8/-4.5$ body lengths in the x_E and y_E directions respectively.

While the velocity is large in the streamwise direction at the end of the mission flight path, the turbulent fluctuations in all three directions reduce to near zero values. Since it is these large scale changes in velocity which have the largest impact on the quadrotor's position, comparable position performance is achieved for both RANS and LES database wind at the end of mission 4. After reaching the desired

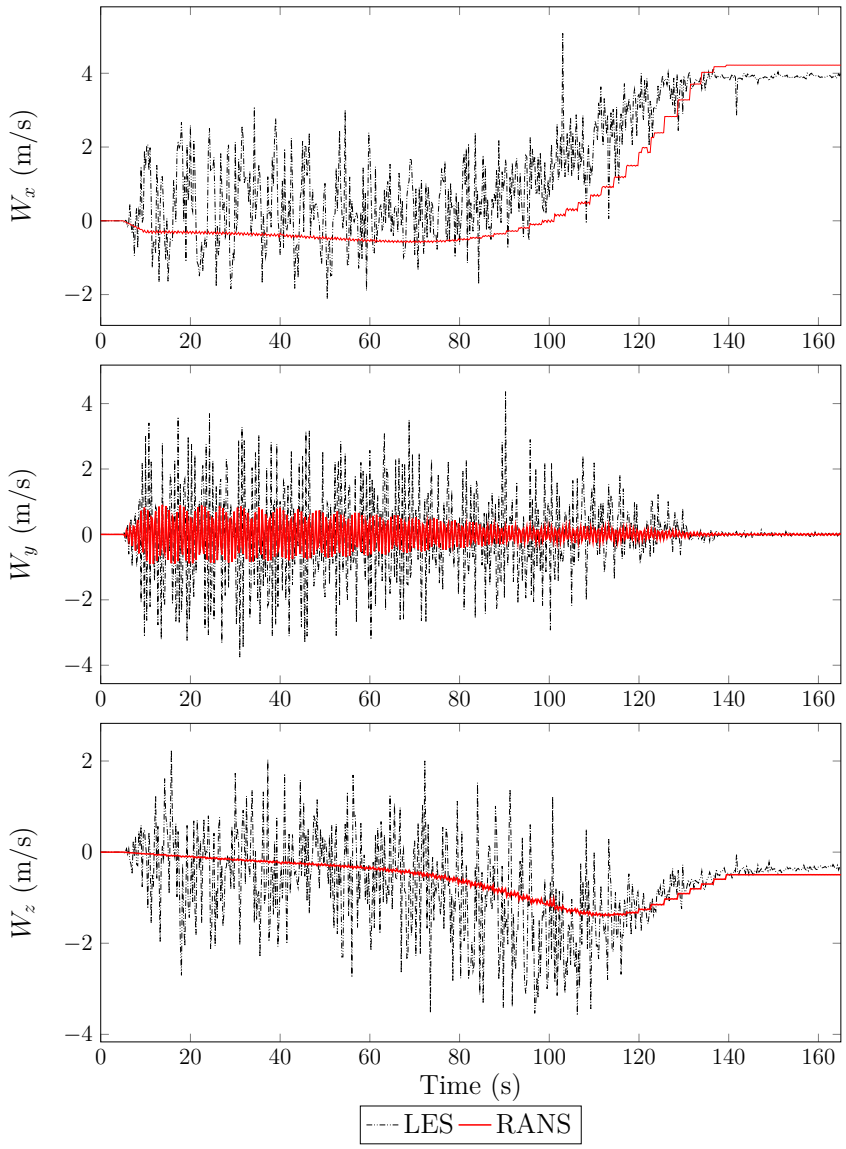
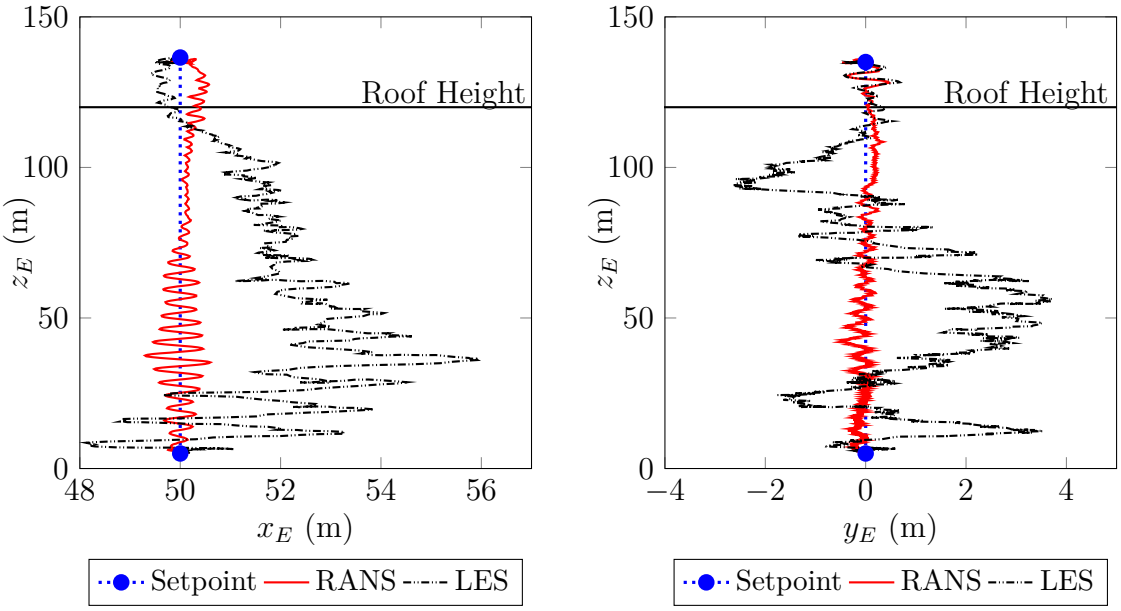


Figure 51: Mission 4 - wind velocity components

135 m altitude the quadrotor attempts to maintain it's position for 20 seconds. It is found that the positions for both the RANS and LES background wind conditions converge to produce deviation performance similar to mission 1 of ±1 body length in both the x_E and y_E directions, and ±0.25 body lengths in both the z_E.



(a) Freestream parallel plane **(b)** Crosswind parallel plane

Figure 52: Mission 4 - quadrotor position

5.5 Mission 5 - Crosswind Wake Translation

The cross wake flight of Mission 5 is designed to test the controller's ability to hold its streamwise position and altitude while traveling in the corsswind direction. This situation is an initial step to move towards the previous work of Galway et al. [19, 51, 52] which ultimately consisted of flying through multiple building wakes for realistic flight testing of missions in an urban environment. To study how traveling in and out of the wake impacts the quadrotor's autonomous flight performance, the single building geometry is used as illustrated in Figure 53.

The wake velocity components the quadrotor experiences over the flight are shown in Figure 54. From the streamwise component is is clear when the quadrotor

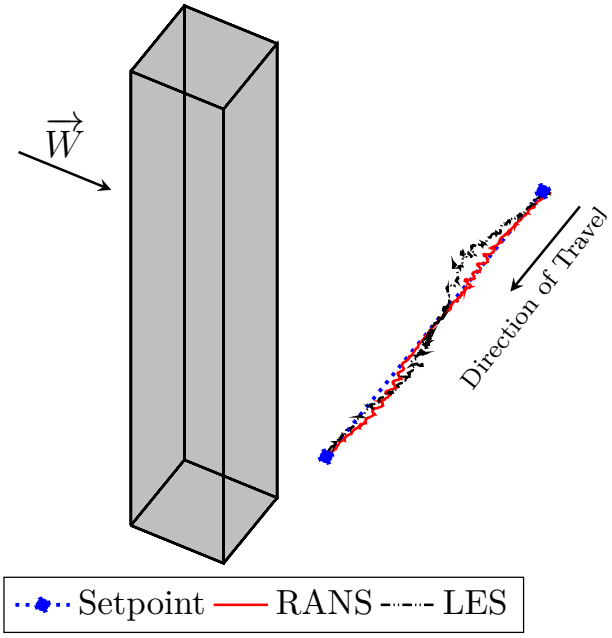


Figure 53: Mission 5 - LES deviations and building

leaves the freestream (with u=4 m/s), at the center of the lower velocity wake (u≈1 m/s), and back to the freesteam. As experienced in Mission 2, the crosswind component exhibits high frequency oscillations and there is a negative ground normal component of w≈1.5 m/s near the center of the building.

The two flight paths are again separated into two planes, wind parallel (x_E-y_E) and crosswind parallel (y_E-z_E) in Figures 55a and 55b to allow for a clearer visualization of the quadrotor's performance. As the quadrotor enters the wake, the freestream velocity causes the controller to over compensate as it decreases causing a large streamwise deviation upwind of the setpoint. As expected this divergence coupled with the demand of a moving setpoint in the crosswind direction results in deviations in the quadrotor's altitude. The RANS wake field produces

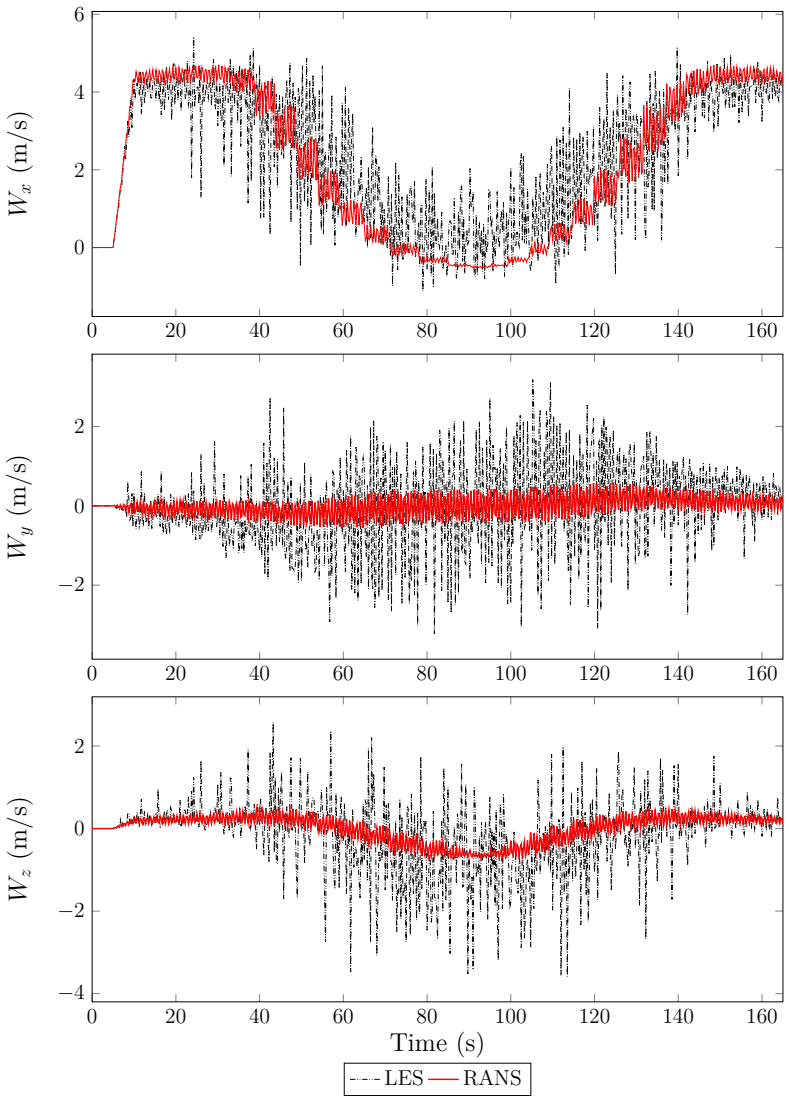
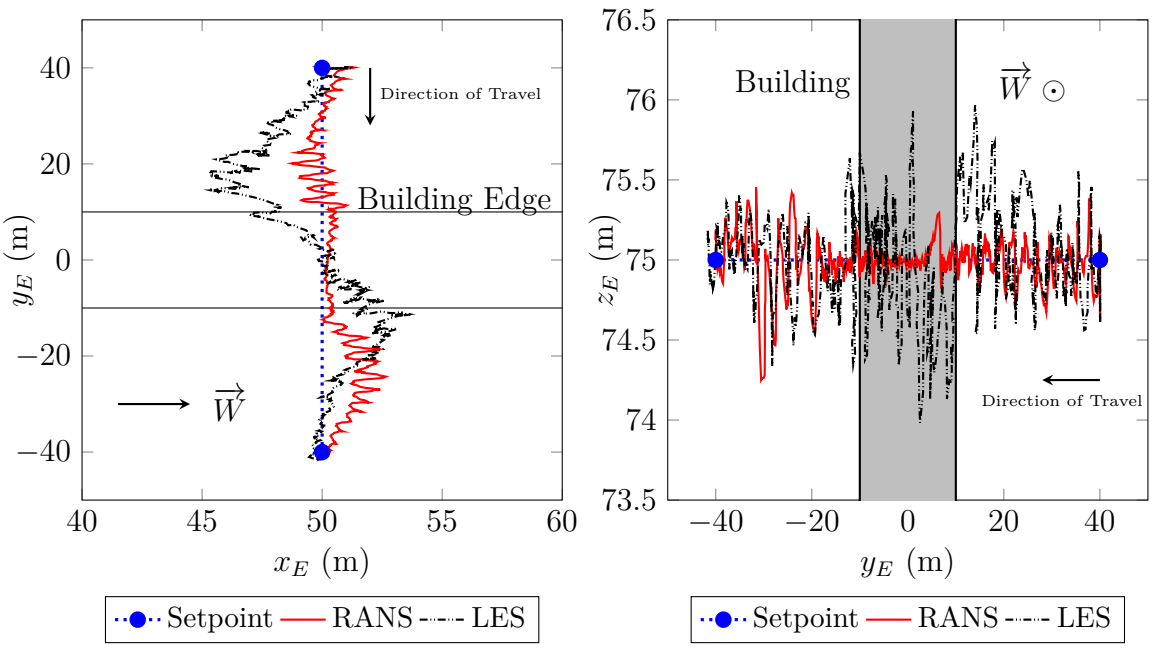


Figure 54: Mission 5 - wind velocity components

deviations of approximately -2 and ± 0.25 body lengths in the x_E and z_E directions respectively. When using the LES wake field the quadrotor reaches up to ± 8 body lengths in the x_E as it enters the wake and ± 2 body lengths in the z_E direction.

All three wake velocity components have smaller magnitudes in the wake (see t=90s

in Figure 54), and with a smaller position deviation upon entering the wake the RANS wake field produces a deviation of approximately +0.5 body lengths in the x_E direction and on average almost 0 in the z_E direction. When using the LES wake field velocities, the quadrotor does not have time to recover from the extreme position deviation before going through the wake. As the controller fights to reach the setpoint it overshoots, producing in z_E deviations of $\pm$ 2 body lengths as it begins to experience the increasing streamwise velocity. As the quadrotor leaves the wake this increasing velocity results in an x_E deviation of +6 body lengths for the LES wake field. A similar trend is seen with the RANS results but has a lower magnitude with an x_E deviation of +4 body lengths downstream of the setpoint.



(a) Freestream parallel plane **(b)** Crosswind parallel plane

Figure 55: Mission 5 - quadrotor position

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

Motived by the many uses, increasing popularity, and growing urban populations a method is presented for using LES to generate urban wake velocity fields for use in studying wake effects on autonomous quadrotor's flight performance. The flow field is generated around a single building in isolation, representing the simplest building block geometry to build up a full urban environment. The wake velocities are stored in a database and accessed by a MATLAB/Simulink flight simulator based on a custom quadrotor platform. This simulator and database is used to study the difference in flight performance between wake fields generated by RANS methods and LES with five flight missions. The mission types and their locations are chosen to study a specific performance ability such as position holding in a turbulent wake or vertical ascent in the wake.

Results of holding position in a constant freestream show both CFD methods produce similar results and can hold position in all three directions within approximately ±1.5 body lengths. When the quadrotor is in or on the boundary of the building wake the maximum deviation volumes, as calculated when using a RANS or LES background wind, can differ by 2 orders of magnitude. This is a result of the resolved turbulent fluctuations in the LES wake field causing a greater degree of non-isotropic flow in comparison to RANS. Additionally the LES wake field causes skewed deviations by as much as 5 to 1 in a given direction for both holding position and moving along a desired flight path. Since the LES wake database more accurately reflects the wake fields present behind real world structures, using a wake field replicated by a RANS simulation will significantly over estimate the performance for position hold or slow moving flight paths for multirotor UAVs on the order of 0.5m in size and 2kg in mass.

6.2 Recommendations

The first step in future development is to continue the replication of Galway et al. [19, 51, 52] methods for application to the SUA platform, like a quadrotor, by expanding the urban wake database with buildings at different freestream orientations and more complex geometries such as urban canyons. This can then be expanded further by applying the selection algorithm of Galway to build up an urban environment using the CFD building blocks to simulate flight with multiple buildings. In parallel, the physical quadrotor can be finalized and a mission planned with the goal of collecting experimental data of the quadrotor's autonomous performance with a single building geometry.

The capabilities of the flight simulator can be expanded in the form of testing the effects of 3D interpolation between the known database points (mesh spacing). Initial testing indicates interpolating between the current mesh density does not produce a discernible difference in the either the wake velocity components or the quadrotor's position. And while interpolating cannot produce eddy motion, as this is limited implicitly by the mesh spacing, it may provide useful information for implementing and testing a multi-point quadrotor model. Currently the wind forces are applied to the center of gravity. A multi-point model would allow for the study of flight performance with aerodynamic moments induced by the wind effects acting on other parts of the quadrotor, such as the arms/propellers. Finally the flight simulator can be further developed to include multiple quadrotors for the design and testing of vehicle swarming in urban environments.

List of References

[1] International Civil Aviation Organization. *Unmanned aircraft systems (UAS).* International Civil Aviation Organization, Montréal. ISBN 9789292317515 9292317512 (2011).

[2] Pratt. "-1 Predator unmanned aircraft." Http://commons.wikimedia.org/wiki/File:MQ-1_Predator_unmanned_aircraft.jpg (2008).

[3] C. Slattery. "Northrop Grumman MQ-4c Triton." Http://en.wikipedia.org/w/index.php?title=Northrop_Grumman_MQ-4C_Triton&oldid=642760198 (2015).

[4] Jrfreeland. "Northrop Grumman MQ-8 Fire Scout." Http://en.wikipedia.org/w/index.php?title=Northrop_Grumman_MQ-8_Fire_Scout&oldid=640305274 (2015).

[5] MilborneOne. "Schiebel Camcopter S-100." Http://en.wikipedia.org/w/index.php?title=Schiebel_Camcopter_S-100&oldid=643663322 (2015).

[6] KrisfromGermany. " Aladin." Http://en.wikipedia.org/w/index.php?title=EMT_Aladin&oldid=542404626 (2015).

[7] Dkroetsch. “Miniature UAV.” Http://en.wikipedia.org/w/index.php?title=Miniature_UAV&oldid=644858733 (2015).

[8] TFCforever. “de Bothezat helicopter.” Http://en.wikipedia.org/w/index.php?title=De_Bothezat_helicopter&oldid=609142829 (2015).

[9] Halftermeyer. “.Drone.” Http://pl.wikipedia.org/w/index.php?title=AR.Drone&oldid=41440599 (2015).

[10] M. J. Allen. “Autonomous soaring for improved endurance of a small uninhabited air vehicle.” In “Proceedings of the 43rd aerospace sciences meeting, AIAA,” (2005).

[11] J. W. Langelaan. “Long distance/duration trajectory optimization for small uavs.” In “ Guidance, Navigation and Control Conference and Exhibit, Hilton Head, SC,” (2007).

[12] A. Chapman and M. Mesbahi. “ flocking with wind gusts: Adaptive topology and model reduction.” In “American Control Conference (ACC), 2011,” pages 1045–1050 (2011).

[13] J. Guerrero, J. Escareno, and Y. Bestaoui. “Quad-rotor MAV trajectory planning in wind fields.” In “2013 IEEE International Conference on Robotics and Automation (ICRA),” pages 778–783 (2013).

[14] S. Waslander and C. Wang. “Wind Disturbance Estimation and Rejection for Quadrotor Position Control.” American Institute of Aeronautics and Astronautics. ISBN 978-1-60086-979-2 (2009).

[15] J. Escareno, S. Salazar, H. Romero, and R. Lozano. “Trajectory Control of a Quadrotor Subject to 2d Wind Disturbances.” *Journal of Intelligent & Robotic Systems* **70**(1-4), 51–63. ISSN 0921-0296, 1573-0409 (2012).

[16] Y. Chen, Y. He, and M. Zhou. “Modeling and Control of a Quadrotor Helicopter System under Impact of Wind Field.” (2013).

[17] M. Kothari, I. Postlethwaite, and D.-W. Gu. " Path Following in Windy Urban Environments." *Journal of Intelligent & Robotic Systems* ISSN 0921-0296, 1573-0409 (2013).

[18] M. Orr, S. Rasmussen, E. Karni, and W. Blake. "Framework for developing and evaluating MAV control algorithms in a realistic urban setting." In "American Control Conference, 2005. Proceedings of the 2005," pages 4096–4101 vol. 6 (2005).

[19] D. Galway, J. Etele, and G. Fusina. *Urban Wind Modeling with Application to Autonomous Flight*. Masters Thesis, Carleton University (2009).

[20] "Urban Development." Http://data.worldbank.org/topic/urban-development (2014).

[21] D. Murphy and J. Cycon. *Applications for mini VTOL UAV for law enforcement*.

[22] T. Hegazy, B. Ludington, and G. Vachtsevanos. "Reconnaissance and surveillance in urban terrain with unmanned aerial vehicles." In "Proceedings of 16 th IFAC World Congress," pages 4–8 (2005).

[23] R. L. Mota, L. F. Felizardo, E. H. Shiguemori, A. C. Ramos, and F. Mora-Camino. "Expanding Small UAV Capabilities with ANN: A Case Study for Urban Areas Inspection." *British Journal of Applied Science & Technology* **4**(2), 387–398 (2014).

[24] AeroVironment. "Qube Public Safety UAS - AeroVironment, Inc." Http://www.avinc.com/public-safety/qube (2015).

[25] T. Coyle and . a. p. S. M. V. Blog. "Comparing traditional CSI procedures with UAV Mapping." Http://diydrones.com/profiles/blogs/comparing-traditional-csi-procedures-with-uav-mapping (2014).

[26] "Amazon Prime Air." Http://www.amazon.com/b?node=8037720011 (2014).

[27] T. A. Press. " begins drone delivery in Germany." Http://www.cbc.ca/1.2777647 (2014).

[28] L. Kelion. "Alibaba begins drone delivery trials." Http://www.bbc.com/news/technology-31129804 (2015).

[29] F. P. Staff. "Pizza-delivering drones? Domino's shows off unmanned aircraft that delivers customer orders." *Financial Post* Http://business.financialpost.com/2013/06/06/dominos-drone-pizza-delivery/ (2013).

[30] J. Etele. "Overview of Wind Gust Modelling with Application to Autonomous Low-Level UAV Control." Contract Report (2006).

[31] S. A. Raza and J. Etele. "Simulation tool for testing and validating uav autopilots in wind gust environments." *AIAA Atmospheric Flight Mechanics Conference and Exhibit* (2012).

[32] F. White. *Fluid Mechanics*. McGraw-Hill, 7 edition edition. ISBN 9780077422417 (2010).

[33] R. B. Stull. *An Introduction to Boundary Layer Meteorology*. Springer, New York, softcover reprint of the original 1st ed. 1988 edition edition. ISBN 9789027727695 (1988).

[34] X. Tapia. *Modeling wind flow over complex terrain using OpenFoam*. Masters Thesis, University of Gavle (2009).

[35] R. E. Britter and S. R. Hanna. "Flow and Dispersion in Urban Areas." *Annual Review of Fluid Mechanics* **35**(1), 469–496 (2003).

[36] J. Wieringa, A. Davenport, S. S. Grimmond, and T. Oke. "New revision of Davenport Roughness Classification." *Proc., 3EACWE, Eindhoven, The Netherlands* (2001).

[37] P. Gousseau, B. Blocken, and G. van Heijst. " simulation of pollutant dispersion around isolated buildings: On the role of convective and turbulent mass fluxes in the prediction accuracy." *Journal of Hazardous Materials* **194**, 422–434. ISSN 03043894 (2011).

[38] A. Walton and A. Y. S. Cheng. "Large-eddy simulation of pollution dispersion in an urban street canyonPart II: idealised canyon simulation." *Atmospheric Environment* **36**(22), 3615–3627. ISSN 1352-2310 (2002).

[39] S. M. Salim, R. Buccolieri, A. Chan, and S. Di Sabatino. "Numerical simulation of atmospheric pollutant dispersion in an urban street canyon: Comparison between RANS and LES." *Journal of Wind Engineering and Industrial Aerodynamics* **99**(2-3), 103–113. ISSN 01676105 (2011).

[40] Y.-H. Tseng, C. Meneveau, and M. B. Parlange. "Modeling flow around bluff bodies and predicting urban dispersion using large eddy simulation." *Environmental science & technology* **40**(8), 2653–2662. ISSN 0013-936X (2006).

[41] T. Tamura. "Large eddy simulation on building aerodynamics." In "Proceedings of the seventh Asia-Pacific Conference on Wind Engineering, pp131-157," (2009).

[42] J. He and C. C. S. Song. "Evaluation of pedestrian winds in urban area by numerical approach." *Journal of Wind Engineering and Industrial Aerodynamics* **81**(1-3), 295–309. ISSN 0167-6105 (1999).

[43] M. Bottema. "Urban roughness modelling in relation to pollutant dispersion." *Atmospheric Environment* **31**(18), 3059–3075. ISSN 1352-2310 (1997).

[44] F. M. Hoblit. *Gust Loads on Aircraft: Concepts and Applications*. Education Series. American Institute of Aeronautics and Astronautics, Washington, D.C. ISBN 0-930403-45-2 (1988).

[45] R. Martinuzzi and C. Tropea. "The flow around surface-mounted, prismatic obstacles placed in a fully developed channel flow (Data Bank Contribution)." *Journal of Fluids Engineering* **115**(1), 85–92 (1993).

[46] W. Rodi. "Comparison of LES and RANS calculations of the flow around bluff bodies." *Journal of Wind Engineering and Industrial Aerodynamics* **69-71**, 55–75. ISSN 0167-6105 (1997).

[47] S. Dutta, P. K. Panigrahi, and K. Muralidhar. "Experimental investigation of flow past a square cylinder at an angle of incidence." *Journal of engineering mechanics* **134**(9), 788–803 (2008).

[48] K. Elshorbagy, E. Wahba, and R. Afify. "Visualization versus CFD Simulation of Laminar Flow past Bluff Body." (2010).

[49] S. Houda, N. Zemmouri, A. Hasseine, R. Athmani, and R. Belarbi. "A CFD Model for Simulating Ubran Flow in Complex Morphological Street Network." *The Online Journal of Science and Technology* **2**(1), 1–10 (2012).

[50] J. Labovský and u. Jelemenský. "-based atmospheric dispersion modeling in real urban environments." *Chemical Papers* **67**(12), 1495–1503. ISSN 0366-6352, 1336-9075 (2013).

[51] D. Galway, J. Etele, and G. Fusina. "Modeling of Urban Wind Field Effects on Unmanned Rotorcraft Flight." *Journal of Aircraft* **48**(5), 1613–1620. ISSN 0021-8669, 1533-3868 (2011).

[52] D. Galway, J. Etele, and G. Fusina. "Development and implementation of an urban wind field database for aircraft flight simulation." *Journal of Wind Engineering and Industrial Aerodynamics* **103**, 73–85. ISSN 0167-6105 (2012).

[53] O. Foundation. " User Guide." (2014).

[54] H. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Prentice Hall, Harlow, England ; New York, 2 edition edition. ISBN 9780131274983 (2007).

[55] E. de Villers. *The Potential of Large Eddy Simulation for the Modeling of Wall Bounded Flows*. Ph.D. thesis, Imperial College of Science, Technology and Medicine (2006).

[56] O. Reynolds. "An Experimental Investigation of the Circumstances Which Determine Whether the Motion of Water Shall Be Direct or Sinuous, and of the Law of Resistance in Parallel Channels." *Philosophical Transactions of the Royal Society of London* **174**, 935–982. ISSN 0261-0523 (1883).

[57] J. M. Mcdonough. *LECTURES on TURBULENCE Physics, Mathematics and Modeling* (2004).

[58] H. Tennekes and J. L. Lumley. *A First Course in Turbulence*. MIT Press. ISBN 9780262200196 (1972).

[59] B. E. Launder and D. B. Spalding. "The numerical computation of turbulent flows." *Computer Methods in Applied Mechanics and Engineering* **3**(2), 269–289. ISSN 0045-7825 (1974).

[60] S. B. Pope. *Turbulent Flows*. Cambridge University Press. ISBN 9780521598866 (2000).

[61] A. Leonard. *Energy cascade in large-eddy simulations of turbulent fluid flows* (1974).

[62] H. Lu, C. J. Rutland, and L. M. Smith. "A priori tests of one-equation LES modeling of rotating turbulence." *Journal of Turbulence* page N37 (2007).

[63] Charlesreid1. "Filter (large eddy simulation)." Http://en.wikipedia.org/wiki/Filter_%28large_eddy_simulation%29 (2010).

[64] P. Sagaut. *Large Eddy Simulation for Incompressible Flows - An Introduction*. Scientific Computation. Springer, 3rd edition. ISBN 978-3-540-26344-9 (2006).

[65] A. Yoshizawa and K. Horiuti. "A statistically-derived subgrid-scale kinetic energy model for the large-eddy simulation of turbulent flows." *Journal of the Physical Society of Japan* **54**(8), 2834–2839 (1985).

[66] S. Krajnovic and L. Davidson. "Large-Eddy Simulation of the Flow Around a Bluff Body." *AIAA Journal* **40**(5), 927–936. ISSN 0001-1452 (2002).

[67] J. Franke, A. Hellsten, H. Schlunzen, and B. Carissimo, editors. *Proceedings / International Workshop on Quality Assurance of Microscale Meteorological Models Cost action 732 in combination with the European Science Foundation at Hamburg, Germany, July 28/29, 2005*. Univ., Meteorological Inst., Centre for Marine and Atmospheric Sciences, Hamburg. ISBN 3000183124 9783000183126 (2005).

[68] Y. Tominaga, A. Mochida, R. Yoshie, H. Kataoka, T. Nozu, M. Yoshikawa, and T. Shirasawa. " guidelines for practical applications of CFD to pedestrian wind environment around buildings." *Journal of Wind Engineering and Industrial Aerodynamics* **96**(10-11), 1749–1761. ISSN 01676105 (2008).

[69] I. B. Celik, Z. N. Cehreli, and I. Yavuz. "Index of Resolution Quality for Large Eddy Simulations." *Journal of Fluids Engineering* **127**(5), 949–958. ISSN 0098-2202 (2005).

[70] P. Gousseau, B. Blocken, and G. van Heijst. "Quality assessment of Large-Eddy Simulation of wind flow around a high-rise building: Validation and solution verification." *Computers & Fluids* **79**, 120–133. ISSN 00457930 (2013).

[71] T. Meng and K. Hibi. "Turbulent measurements of the flow field around a high-rise building." *Journal of Wind Engineering Japan* (76), 55–64. (in Japanese) (1998).

[72] J. Jeong and F. Hussain. "On the identification of a vortex." *Journal of Fluid Mechanics* **285**, 69–94. ISSN 1469-7645 (1995).

[73] J. C. R. Hunt, A. A. Wray, and P. Moin. "Eddies, streams, and convergence zones in turbulent flows." (1988).

[74] V. Kolar. "Vortex identification: New requirements and limitations." *International Journal of Heat and Fluid Flow* **28**(4), 638–652. ISSN 0142727X (2007).

[75] G. R. Meneely and N. L. Kaltreider. "The volume of the lung determined by helium dilution. Description of the method and comparison with other procedures." *Journal of Clinical Investigation* **28**(1), 129 (1949).

[76] B. Etkin and L. D. Reid. *Dynamics of Flight: Stability and Control*. Wiley, New York, 3 edition edition. ISBN 9780471034186 (1995).

Appendix A

LES Standard Working Directory

This appendix details the setup and required file structure to perform a LES with OpenFOAM as outlined in Chapter 2. A standard working directory (SWD) consisting of files and scripts is used to simplify the user work flow for running future simulations to expand the urban wind database. Figure 56 illustrates the simplicity and few steps required by the user to run the LES and Figure 57 shows the full required SWD.

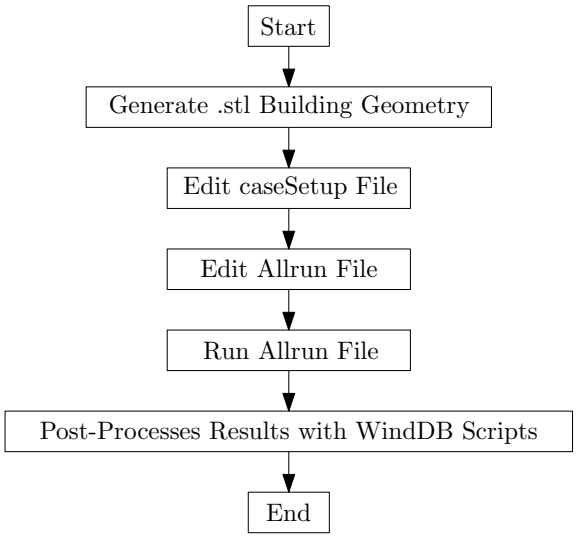


Figure 56: OpenFOAM LES user workflow

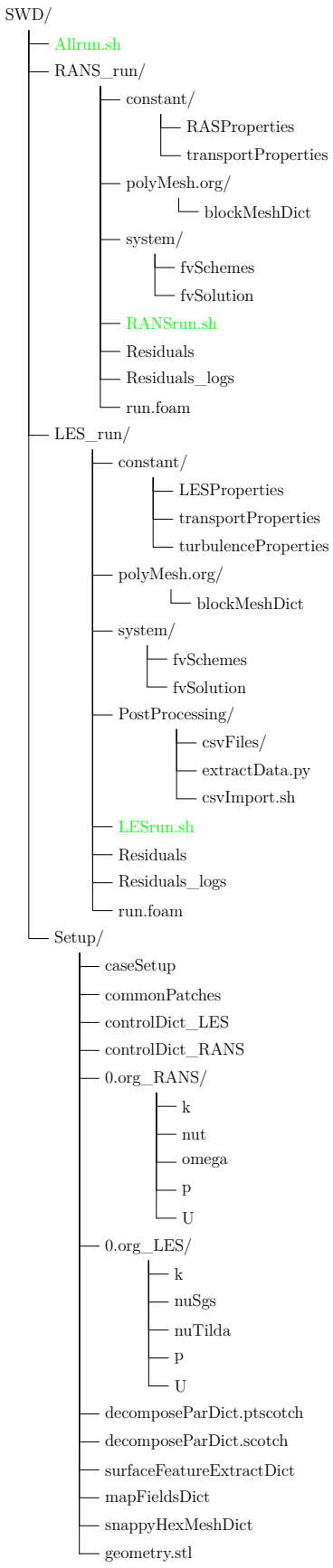


Figure 57: Outline of the standard OpenFOAM working directory

A zip and tar containing the latest version of the SWD can be downloaded with the following URL: https://github.com/sutherlandm/tara_les-swd/releases. The Allrun script is used to control which parts of the RANS and LES cases are performed, shown in Listing A.1.

Listing A.1: Allrun.sh

```
#!/bin/sh
cd ${0%/*} || exit 1

################################################################################
# Allrun.sh
#
# Mark Sutherland
# December 18, 2014
#
# Overarching script to run a standard OpenFOAM LES. Will run a parallel
# steady k-w SST RANS CFD simulation, map the final results, and run a parallel
# large-eddy CFD simulation.

#Ensure variables below are set, the standard working directory files are present, such
#as the geometry .stl file, and all variables in Setup/caseSetup are set.

CORENUM=16          #Number of parallel cores (must be local, non-network)
DORANS_MESH=1       #Do the RANS case meshing
DORANS=1            #Run the potential and k-e sims after meshing
DOMAP=1             #Reconstruct the RANS results and map after LES meshing
DOLES_MESH=1        #Do the LES case meshing
DOLES=1             #Run the LES sim after meshing and mapping
RECON=0             #Reconstruct LES results (not required for .csv generation)
################################################################################

#Export the variables so the RANS and LES run scripts can use them
export CORENUM DORANS_MESH DORANS DOMAP DOLES_MESH DOLES RECON

#Do the RANS steady-state case
cd RANS_run
chmod +x RANSrun
./RANSrun
cd ..

#Do the LES case
cd LES_run
chmod +x LESrun
./LESrun
cd ..
```

From the Allrun script and the SWD it is seen a RANSrun script is first called to control the RANS simulation based on the flags set in the Allrun, such as meshing and mapping. The RANSrun script is shown below in Listing A.2 with detailed comments explaining the clean up, setup, meshing, running, and reconstruction of the steady RANS case.

Listing A.2: RANSrun.sh

```
#!/bin/sh
cd ${0%/*} || exit 1

#############################################################################################
# RANSrun.sh
#
# Mark Sutherland
# December 18, 2014
#
# Script to run the cleaning, meshing and running of the RANS case
#############################################################################################

#Run the RANS meshing section which includes directory clean up
if [ $DORANS_MESH -eq 1 ]; then
  #Clean up the logs from last run if you need to do a full restart
  rm -rf logs
  rm -rf constant/polyMesh
  rm -rf constant/triSurface
  rm -r 0
  rm -r 0.org
  rm -rf processor*
  rm -rf constant/extendedFeatureEdgeMesh
  echo "----------Cleaned Last Run----------\n"
  sleep 1

  #Copy files from Setup folder
  mkdir logs
  mkdir 0
  mkdir ./constant/triSurface
  cp -r polyMesh.org constant/polyMesh
  cp ../Setup/*.stl ./constant/triSurface
  cp ../Setup/surfaceFeatureExtractDict ./system
  cp ../Setup/snappyHexMeshDict ./system
  cp -r ../Setup/0.org_RANS ./0.org
  cp ../Setup/controlDict_RANS ./system/controlDict
  cp ../Setup/decomposeParDict.ptscotch ./system/decomposeParDict.ptscotch
  cp ../Setup/decomposeParDict.scotch ./system/decomposeParDict.scotch
```

```
38
39  #Run blockMesh to make general background mesh
40  blockMesh | tee log_blockMesh
41  mv -i log_blockMesh logs
42  echo "----------blockMesh Done----------\n"
43  sleep 1
44
45  #Move the blockMesh around to center it about the .stl body if needed
46  transformPoints -translate '(0 0 0)' | tee log_transformPoints
47  mv -i log_transformPoints logs
48  sleep 1
49
50  #This is very important, it extracts the edges of the stl file to allow sHM to snap to them
51  surfaceFeatureExtract | tee log_surfaceFeatureExtract
52  mv -i log_surfaceFeatureExtract logs
53  sleep 1
54
55  #We need to copy over some files to allow for auto processor splitting
56  cp system/decomposeParDict.scotch system/decomposeParDict
57
58  #Lets breakup the domain to allow for parallel processing
59  decomposePar | tee log_decomposePar
60  mv -i log_decomposePar logs
61  echo "----------decomposePar Done----------\n"
62  sleep 1
63
64  #Again copy come files over, difference bettwen scotch and ptscotch
65  cp system/decomposeParDict.ptscotch system/decomposeParDict
66
67  #Now the actual fun part, lets run sHM to carve out our stl from the blockMesh
68  mpirun -np $CORENUM snappyHexMesh -parallel -overwrite | tee log_snappyHexMesh
69  mv -i log_snappyHexMesh logs
70  echo "----------snappyHexMesh Done----------\n"
71  sleep 2
72
73  #For parallel running
74  ls -d processor* | xargs -i rm -rf ./{}/0 $1
75  ls -d processor* | xargs -i cp -r 0.org/ ./{}/0/ $1
76
77  #Run renumberMesh to clean a few things up
78  mpirun -np $CORENUM renumberMesh -parallel -overwrite -latestTime | tee log_renumberMesh
79  mv -i log_renumberMesh logs
80  echo "----------renumberMesh Done----------\n"
81
82  #Run checkMesh to ensure accetable quality mesh
83  mpirun -np $CORENUM checkMesh -parallel -latestTime | tee log_checkMesh
84  mv -i log_checkMesh logs
85  echo "----------checkMesh Done----------\n"
86  sleep 1
```

```
87
88    #Run patchSummary to check all boundary patches are correct
89    mpirun -np $CORENUM patchSummary -parallel -latestTime | tee log_patchSummary
90    mv -i log_patchSummary logs
91    echo "----------patchSummary Done----------\n"
92    sleep 1
93  fi
94
95  #Run the potential and RANS cases if true flag. Both are run in parallel. The potential log is
96  #outputed to both the termial screen and a log file with the tee command. Only a log file is
97  #generated for the simpleFoam case. Follow the simulation with tail -f log_simpleFoam and
98  #ctl+C to stop following
99  if [ $DORANS -eq 1 ]; then
100   mpirun -np $CORENUM potentialFoam -parallel -initialiseUBCs -noFunctionObjects | tee
        log_potentialFoam
101   PID4=$!         #Save the process PID number
102   wait $PID4      #Wait for this PID to finish
103   mv -i log_potentialFoam logs
104
105   mpirun -np $CORENUM simpleFoam -parallel > log_simpleFoam &
106   PID1=$!
107   wait $PID1
108   mv -i log_simpleFoam logs
109 fi
110
111 #Reconstruct the mesh and finial simulation varaibles for mapping later. Currently the source
112 #must not be parallel but the desintiation can be parallel
113 if [ $DOMAP -eq 1 ]; then
114   reconstructParMesh -mergeTol 1e-06 -constant | tee log_reconstructParMesh #>
        log_reconstructParMesh &
115   PID2=$!
116   wait $PID2
117   mv -i log_reconstructParMesh logs
118
119   reconstructPar -latestTime | tee log_reconstructPar
120   mv -i log_reconstructPar logs
121   echo "----------reconstructPar Done----------\n"
122 fi
```

A similar script is then called for the LES case, as detailed in Listings A.3. Again the file is commented to outline the various steps used to ultimately generate the urban wake velocity fields.

Listing A.3: LESrun.sh

```
#!/bin/sh
cd ${0%/*} || exit 1

#############################################################################################
# LESrun.sh
#
# Mark Sutherland
# December 18, 2014
#
# Script to run the cleaning, meshing, mapping, and running of the LES case
#############################################################################################

#Run the LES meshing section which includes directory clean up
if [ $DOLES_MESH -eq 1 ]; then
  #Clean up the logs from last run if you need to do a full restart
  rm -rf logs
  rm -rf constant/polyMesh
  rm -rf constant/triSurface
  rm -r 0
  rm -r 0.org
  rm -rf processor*
  rm -rf constant/extendedFeatureEdgeMesh
  rm -rf system/mapFieldsDict
  rm -rf postProcessing/forceCoeffs
  rm -rf postProcessing/forces
  echo "----------Cleaned Last Run----------\n"
  sleep 1

  #Copy files from Setup folder
  mkdir logs
  mkdir 0
  mkdir ./constant/triSurface
  cp -r polyMesh.org constant/polyMesh
  cp ../Setup/*.stl ./constant/triSurface
  cp ../Setup/surfaceFeatureExtractDict ./system
  cp ../Setup/snappyHexMeshDict ./system
  cp -r ../Setup/0.org_LES ./0.org
  cp -r polyMesh.org constant/polyMesh
  cp ../Setup/mapFieldsDict ./system
  cp ../Setup/controlDict_LES ./system/controlDict
  cp ../Setup/decomposeParDict.ptscotch ./system/decomposeParDict.ptscotch
  cp ../Setup/decomposeParDict.scotch ./system/decomposeParDict.scotch

  #Run blockMesh to make general background mesh
  blockMesh | tee log_blockMesh
  mv -i log_blockMesh logs
  echo "----------blockMesh Done----------\n"
```

```
48  sleep 1
49
50  #Move the blockMesh around to center it about the .stl body if needed
51  transformPoints -translate '(0 0 0)' | tee log_transformPoints
52  mv -i log_transformPoints logs
53  sleep 1
54
55  #This is very important, it extracts the edges of the stl file to allow sHM to snap to them
56  surfaceFeatureExtract | tee log_surfaceFeatureExtract
57  mv -i log_surfaceFeatureExtract logs
58  sleep 1
59
60  #We need to copy over some files to allow for auto processor splitting
61  cp system/decomposeParDict.scotch system/decomposeParDict
62
63  #Lets breakup the domain to allow for parallel processing
64  decomposePar | tee log_decomposePar
65  mv -i log_decomposePar logs
66  echo "----------decomposePar Done----------\n"
67  sleep 1
68
69  #Again copy come files over, difference bettwen scotch and ptscotch
70  cp system/decomposeParDict.ptscotch system/decomposeParDict
71
72  #Now the actual fun part, lets run sHM to carve out our stl from the blockMesh
73  mpirun -np $CORENUM snappyHexMesh -overwrite -parallel | tee log_snappyHexMesh
74  mv -i log_snappyHexMesh logs
75  echo "----------snappyHexMesh Done----------\n"
76  sleep 2
77
78  # For parallel running
79  ls -d processor* | xargs -i rm -rf ./{}/0 $1
80  ls -d processor* | xargs -i cp -r 0.org/ ./{}/0/ $1
81
82  #Run renumberMesh to clean a few things up
83  mpirun -np $CORENUM renumberMesh -parallel -overwrite -latestTime | tee log_renumberMesh
84  mv -i log_renumberMesh logs
85  echo "----------renumberMesh Done----------\n"
86
87  #Run checkMesh to ensure accetable quality mesh
88  mpirun -np $CORENUM checkMesh -parallel -latestTime | tee log_checkMesh
89  mv -i log_checkMesh logs
90  echo "----------checkMesh Done----------\n"
91  sleep 1
92
93  #Run patchSummary to check all boundary patches are correct
94  mpirun -np $CORENUM patchSummary -parallel -latestTime | tee log_patchSummary
95  mv -i log_patchSummary logs
96  echo "----------patchSummary Done----------\n"
```

```
97    sleep 1
98  fi
99
100 #Map the final RANS results as the intial conditions and run the LES solver. Only a log file is
101 #generated for the pisoFoam case. Follow the simulation with tail -f log_simpleFoam and
102 #ctl+C to stop following
103 if [ $DOLES -eq 1 ]; then
104   mapFields ../RANS_run -consistent -sourceTime 'latestTime' -parallelTarget  > log_mapFields &
105   PID4=$!
106   wait $PID4
107   mv -i log_mapFields logs
108
109   #Run LES!
110   mpirun -np $CORENUM pisoFoam -parallel > log_pisoFoam &
111   PID1=$!        #Save the process PID number
112   wait $PID1     #Wait for this PID to finish
113   mv -i log_pisoFoam logs
114 fi
115
116 #Reconstruct the mesh and any new parallel simulation results if flag is true.
117 if [ $RECON -eq 1 ]; then
118   reconstructParMesh -mergeTol 1e-06 -constant | tee log_reconstructParMesh
119   PID2=$!
120   wait $PID2
121   mv -i log_reconstructParMesh logs
122
123   reconstructPar -newTimes | tee log_reconstructPar
124   PID3=$!
125   wait $PID3
126   mv -i log_reconstructPar logs
127   echo "----------reconstructPar Done----------\n"
128 fi
```

Appendix B

Using OpenFOAM on Kumomotojo

This appendix details how to use the outlined standard working directory (SWD) and remotely interface with the OpenFOAM/Paraview installation on the Kumomotojo workstation.

B.1 Using OpenFOAM

As outlined in Chapter 2 version 2.2.x of OpenFOAM is currently used to generate the urban wake fields, however Kumomotojo has both version 2.2.x and 2.3.x installed if the added features are required in the future[1]. To select a version, you must enter one of the following two alias commands whenever you open a *new* terminal.

```
of22x
```

```
of23x
```

This allows multiple versions of OpenFOAM to work along side one another and provide backwards compatibility for users with existing scripts. To test which version of OpenFOAM is active, if any, call for the help output of the transient laminar

[1] See http://www.openfoam.com/ for the change logs and details.

incompressible flow solver with the following command,

```
icoFoam -help
```

If everything works as planned, the icoFoam usage output shown in Figure 58 should print to the terminal.

```
Usage: icoFoam [OPTIONS]
options:
  -case <dir>       specify alternate case directory, default is the cwd
  -noFunctionObjects
                    do not execute functionObjects
  -parallel         run in parallel
  -roots <(dir1 .. dirN)>
                    slave root directories for distributed running
  -srcDoc           display source code in browser
  -doc              display application documentation in browser
  -help             print the usage

Using: OpenFOAM-2.2.x (see www.OpenFOAM.org)
Build: 2.2.x-61b850bc107b
```

Figure 58: icoFoam usage output with OpenFOAM version

If an OpenFOAM function, such as icoFoam, is called or tab completion does not work a version has not been selected and the messages in Figure 59 will appear. This usually occurs in the haste of opening a new or second terminal window without entering a version selection command as previously outlined.

```
If 'icoFoam' is not a typo you can use command-not-found to lookup the package t
hat contains it, like this:
    cnf icoFoam
```

Figure 59: icoFoam function not found message

The SWD should be downloaded, placed, renamed, and setup in the OpenFOAM *run* directory. This subfolder is in the OpenFOAM folder in the user's home directory, along with the OpenFOAM tutorials, and can easily be accessed with the following change directory alias,

```
run
```

Therefore when the SWD is prepared and ready, the simulation process is started with the following sequence of commands from a new terminal screen,

```
of22x
run
cd renamedSWD
chmod +x Allrun
./Allrun &
disown
```

The & and *disown* commands are important as they place the simulation in the background and then detach it from the current terminal respectively. After the *./Allrun* & command is entered, text will start to flow as the initial outputs of the scripts are teed to both a log file and the terminal screen. Simply typing *disown* and pressing enter 'in the text' will disown and detached the current processes allowing the user to close the terminal and/or logout. The *top* command can then be used to see what application is currently running and the log can be viewed in real time using the following command (in the directory of the log file),

```
tail -f log_applicationName
```

This will produce a self scrolling output of the log file which can be exited at anytime pressing *ctl+c*. Similarly each case contain **Residuals** and **Residuals_logs** files which can be used to monitor the simulation residuals with gnuplot. The four files have been created to take into account the two solving algorithms, simpleFoam and pisoFoam, as well as the log file location, currently being written to or saved in the logs/ folder. Therefore a desired command would take the form of and requires X forwarding to view the resulting plot,

```
gnuplot Residuals
```

B.2 Using ParaView

ParaView is an open-source, multi-platform data analysis and visualization application based on the Visualization Took Kit (VTK) devopled by Kitware Inc, and the Los Alamos and Sandia National Laboratories[2]. ParaView is used in conjunction with OpenFOAM to view the 3D simulation results. A strength of ParaView is its ability to not only process very large data sets and decomposed simulations, but do so in a distributed server-client model to take advantage of remote computational rendering power. Kumomotojo has been setup with a server ability and the following section will detail how to install ParaView on a personal client and connect to the server.

[2] http://www.paraview.org/

Currently ParaView 3.12 (for OpenFOAM-2.2.x) and ParaView 4.1.0 (for OpenFOAM-2.3.x) are installed on the server and for remote viewing the ParaView version on both the server and client must be the same. The various release builds of ParaView for all operating systems can be found here, http://www.paraview.org/download/, and should be downloaded and installed on the client computer. Linux users can look into their repository versions but it could prove easier to download or build from source.

ParaView has been designed to connect a client instance to a host server, rather than simply X-forwarding the servers display. While X-forwarding is far simpler to implement, the resulting performance is quite low and unstable. Therefore a SSH tunnel is used to forward the data from the server to the client via port forwarding.

B.2.1 Configure Linux Client

For a Linux based client the following alias can be added to the .bashrc file to simplify the tunnel startup,

alias **pfTunnel**="ssh *usr*@XXX.XXX.XX.XX -L 11150:localhost:11115 -g -C"

Where *usr* is replaced with your Kumomotojo login user name, and XXX.XXX.XX.XX with Kumomotojo's IP address. Port 11115 is the address the ParaView server will be publishing data to, based on the setup described below. Port 11150 represents the mapped client port which is 'connected' to the server point. Either port numbers can be changed, and should be for multi-connections

and multi-users[3], depending on the users existing port maps.

B.2.2 Configure Windows Client

For Windows based clients, Putty can be used to setup the SSH tunnel and port forward, found at http://www.chiark.greenend.org.uk/~sgtatham/putty/. Open an instance of Putty, and click *Tunnels* under the *SSH* category, shown in Figure 60. Enter the source port, the IP address with ParaView server port, and click *Add*.

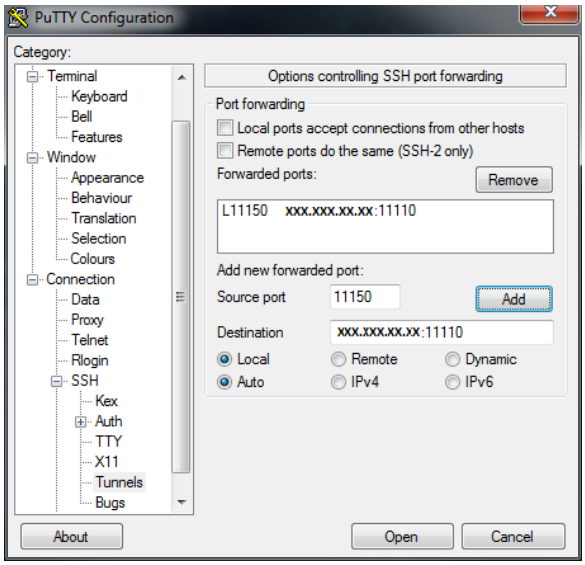


Figure 60: SSH tunnel Putty setup

Select the *Session* category, enter Kumomotojo's IP address, a desired name such as *Kumo*, and *Save* shown in Figure 61. This will save the IP address and SSH tunnel information for future connections by selecting the session name and selecting *Load*. Start the SSH session by clicking the *Open* button.

[3]Multi-connections and multi-users are untested and tests are recommended for future work

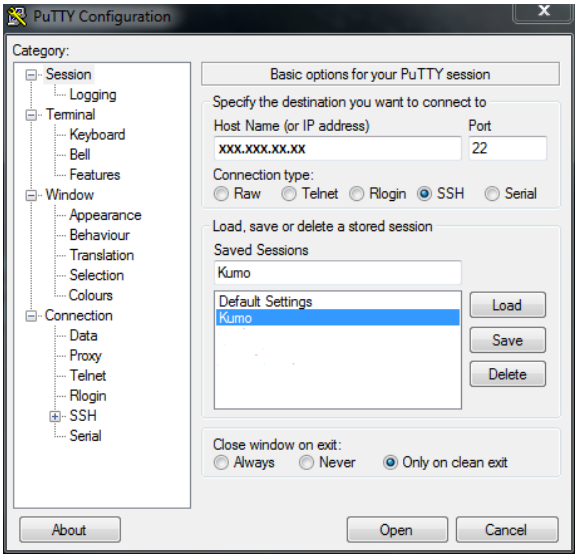


Figure 61: Putty connection setup

With the SSH tunnel setup between the server and client, the next step is to start the ParaViw Server. To facilitate easy startup times, the following aliases should be added to your users .bashrc file on the server,

alias **pfServer**="pvserver –server-port=11115"

alias **pfServer_OS**="pvserver –server-port=11115 –use-offscreen-rendering"

alias **pfServer_M2**="mpirun -np 2 pvserver –server-port=11115"

alias **pfServer_M4**="mpirun -np 4 pvserver –server-port=11115"

These sets of commands are used to start an instance of a ParaView server with different settings. There are 3 main options for the server:

- **pfServer**
  - Runs a 'standard' pvserver such that is uses the GPU to render into an X window before showing it in the GUI. This window will pop open on the client's desktop automatically when the system needs to render something, such as a coloured contour plane. This is a single serial server meaning the size of mesh/simulation is limited. The upper limit is not clearly defined but a rough estimate would be my course LES grid with 1.13×10^6 cells. If the server connection drops or takes an excessively long time to load, the server needs more power by running in one of the parallel modes outlined below.
- **pfServer_OS**
  - If the server does not have GPU rendering capabilities with OpenGL, the ParaView server uses CPU based rendering though off screen Messa (MessaOS) libraries. This option can also be used if there is an issue with the client opening X windows on the desktop. Since the rendering is being performed by the CPU's, which are typically already busy running simulations, the simulation size is even more limited than the pfServer option. While the response time is slightly quicker than rendering with an X window, only simple simulations of no more than $500,000$ cells should be opened with the server in this mode.
- **pfServer_M2** and **pfServer_M4**
  - For large simulations, such as the Medium and Fine LES simulations (2.43×10^6 and 7.67×10^6 cells) the server should be run in a parallel

mode to distribute the load. The parallelism of the ParaView server is only limited by the hardware of the server therefore these two commands serve only as shortcuts. Even 4 processors should be more than enough to run a server any simulations run on Kumo, however is the number of processes can easily be changed by entering the raw terminal command. While it does employ GPU rendering, running the server in parallel will make a thread for each server section thereby taxing the 'usually always in use' CPUs. *pfServer_M4* should only be used for short periods of post processing if all CPUs are busy to prevent automatic thermal shutdowns of the server.

With the SSH tunnel and ParaView server setup, the last item is to configure ParaView on the client side. After the appropriate version has been downloaded and installed, multiple ParaView versions can be installed on the client, open a new instance of ParaView and select the *connect to server button*, circled in Figure 62.

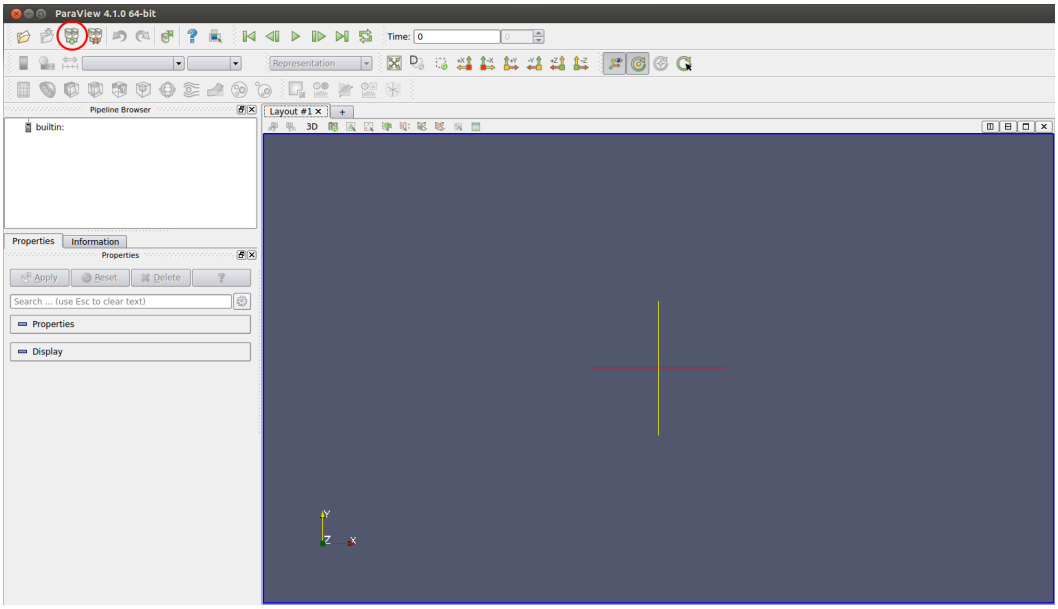


Figure 62: Connect to server selection

In the pop-up window select *Add Server*, shown in Figure 63.

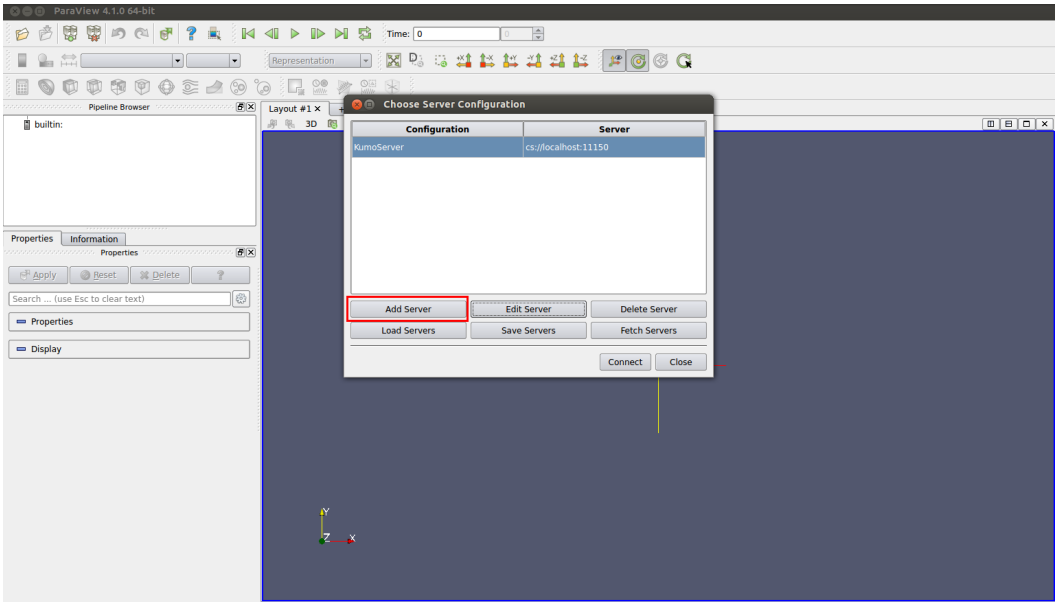


Figure 63: Add new server

Choose a name, leave *localhost* and enter local forwarded port.

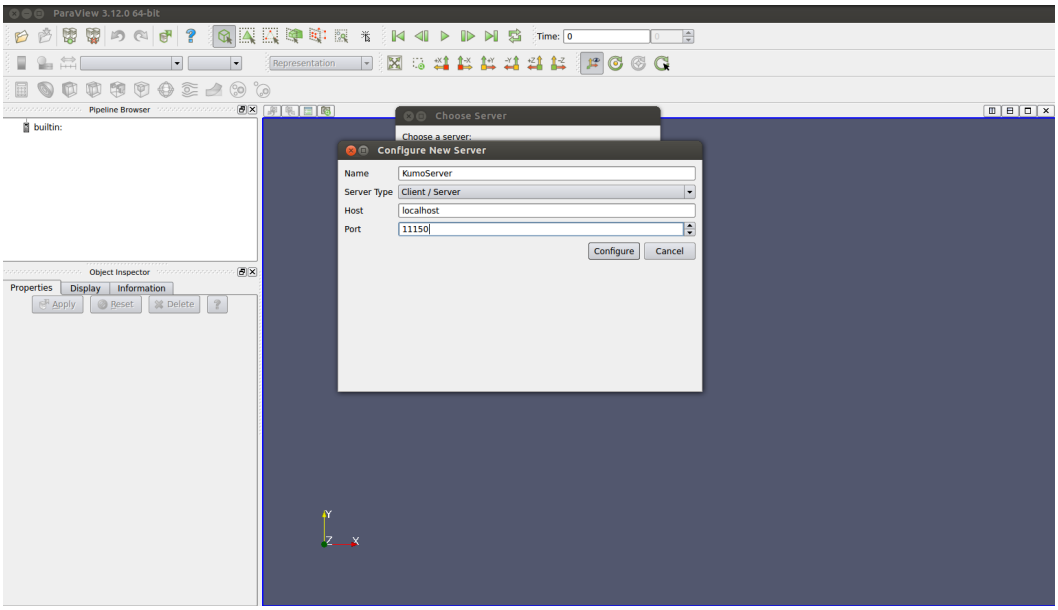


Figure 64: Enter the server details

Select Manual, meaning we will start the server ourselves each time, and Save.

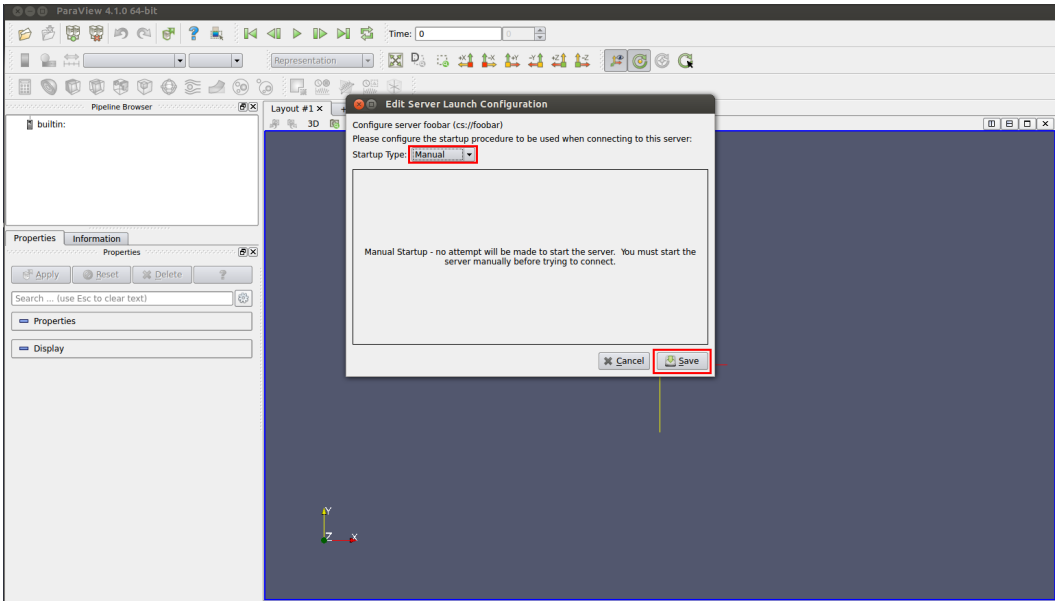


Figure 65: Select Manual and Save

B.2.3 Results Viewing

ParaView should now be setup and ready to post-process the OpenFOAM simulation results. To start viewing data remotely either; open a new terminal and enter *pfTunnel*, or open Putty and load the saved Kumo configuration, if the client is Linux or Windows respectively. You should be prompted for your Kumomotojo user name and/or password. Select the version of OpenFOAM you wish to use/used by entering one of the alias commands, *of22x* or *of23x*.

Navigate to the **foam.run** file in the desired simulation case, however this is optional as you can browse to your case through the client GUI later. Start a ParaView server by entering one of the server start aliases, *pfServer*, *pfServer_OS*, *pfServer_M2* or *pfServer_M4*, or a custom parallel server though the mpirun command. Open an instance of ParaView on the client and select the connect to server button on the top toolbar, Figure 66.

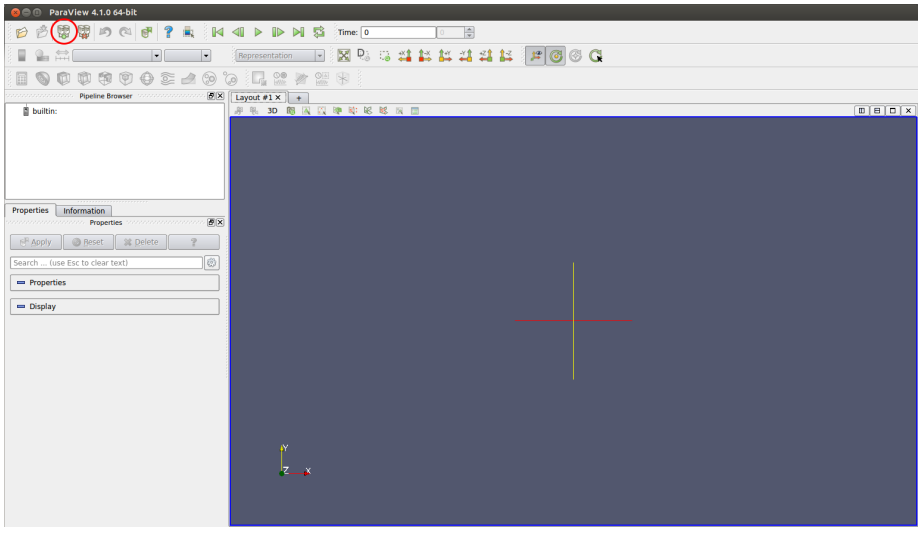


Figure 66: Connect to server selection

When the Choose Server dialog box pops up, click the name of the server from Figure 67 and select connect. You can check the connection status by looking at the SSH tunnel terminal window which should say “Client connected".

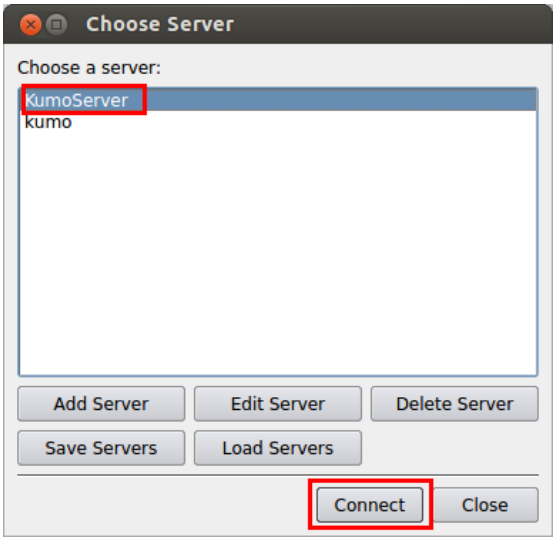


Figure 67: Server selection

If everything is connected correctly you should see **cs://localhost:11150** under the Pipeline Browser (of a different number if you forward to a different port). To open a case press the open button, left most on the top toolbar, shown in Figure 68. The directory it opens to will be the directory where you started the server. Navigate to your case's run.foam file and open. Select either *Reconstructed* or *Decomposed* under *case type* and press the green *Apply* button. It can take a minute or so to load the data and open the rendering X windows if needed. The simulation results can now be remotely viewed. A graphical illustration of this example workflow is shown in Figure 69.

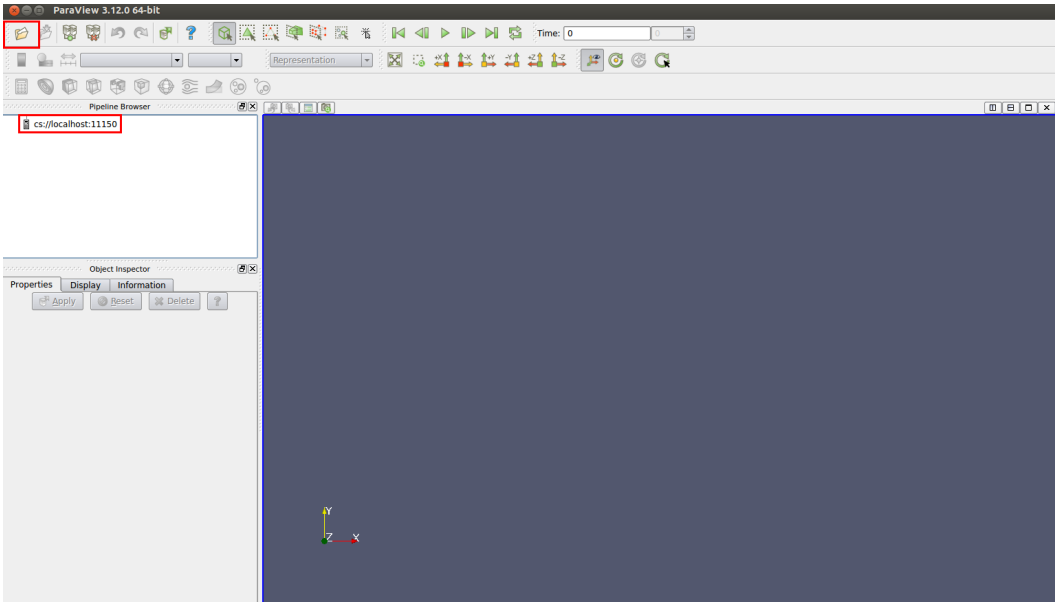


Figure 68: Server connection and open a case

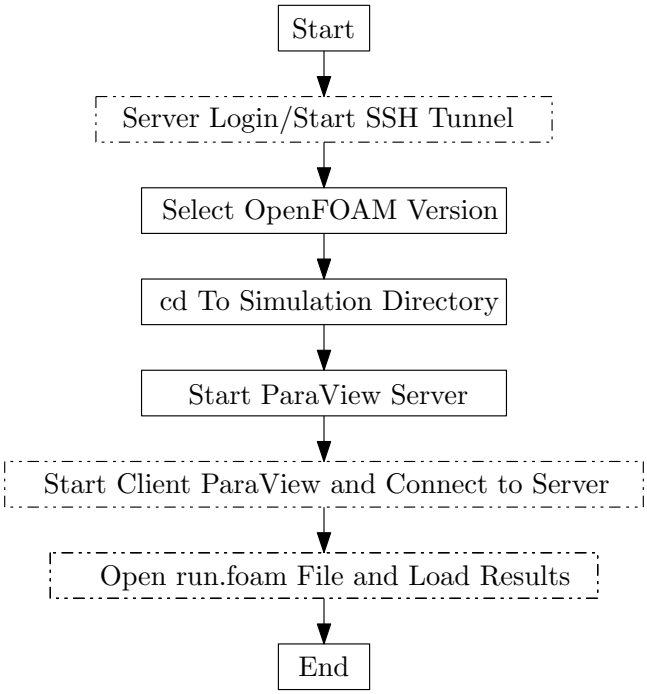


Figure 69: Remote ParaView workflow, dashed: client actions, solid: server actions

Appendix C

Urban Wake Database

This appendix details how the results of the OpenFOAM LES are post-processed into .csv files and how those files are further processed and uploaded to the urban wind database. To first export the .csv files, open the desired LES *run.foam* file with ParaView and ensure the *internalMesh* box is checked after the loading. Forward to the last timestep by selecting the top circled button in Figure 70. This is required as only the **U** and **p** fields should be saved to the .csv files and additional variables are generated which are not present at the start of the simulation, such as U_Mean, or Q. Therefore ensure only **U** and **p** are selected under Cell Arrays, circled in Figure 70, and make sure to click the green *Apply* button.

From the file menu select *Save Data* and the Save File dialog will open. Enter the desired prefix, following the database convention, and ensure to leave an underscore at end, as illustrated in Figure 71. Each timestep is saved a separate .csv file starting from zero and the number is appended to the entered prefix. This format is required for the renaming portion of the post-processing scripts introduced below.

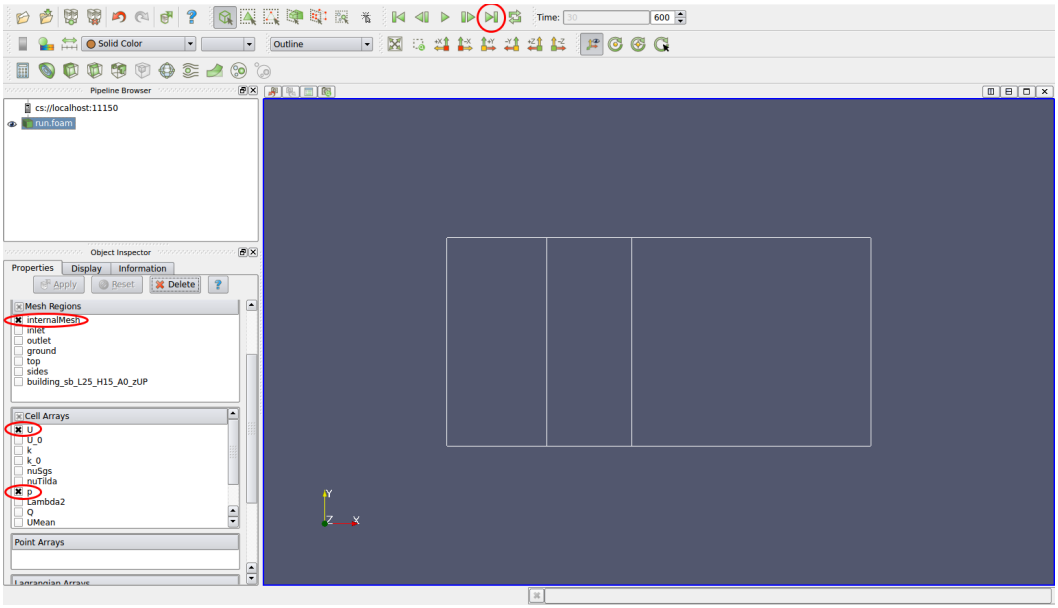


Figure 70: Load run.foam file, adjust timestep and select desired variables

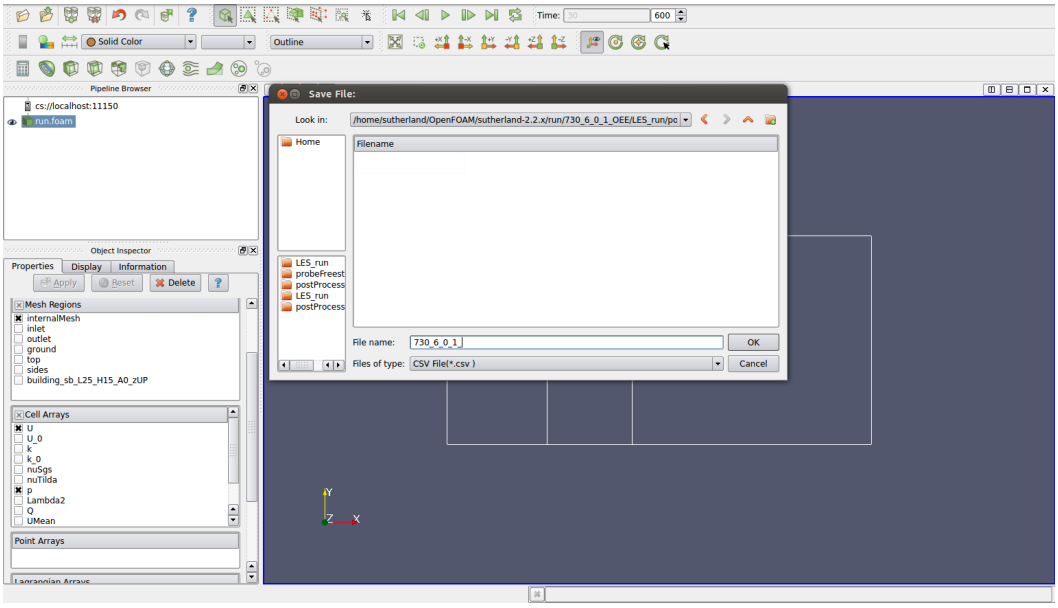


Figure 71: Save data and set file prefix

Configure the writer to save all the timesteps and ensure the field is association on the points, shown in Figure 72. After clicking *Ok* it will take some time to generate the .csv files, a function of the write interval and total simulation time. Similarly, the files can take on the order of 40 GB of hard drive space prior to trimming so ensure sufficient storage space is available.

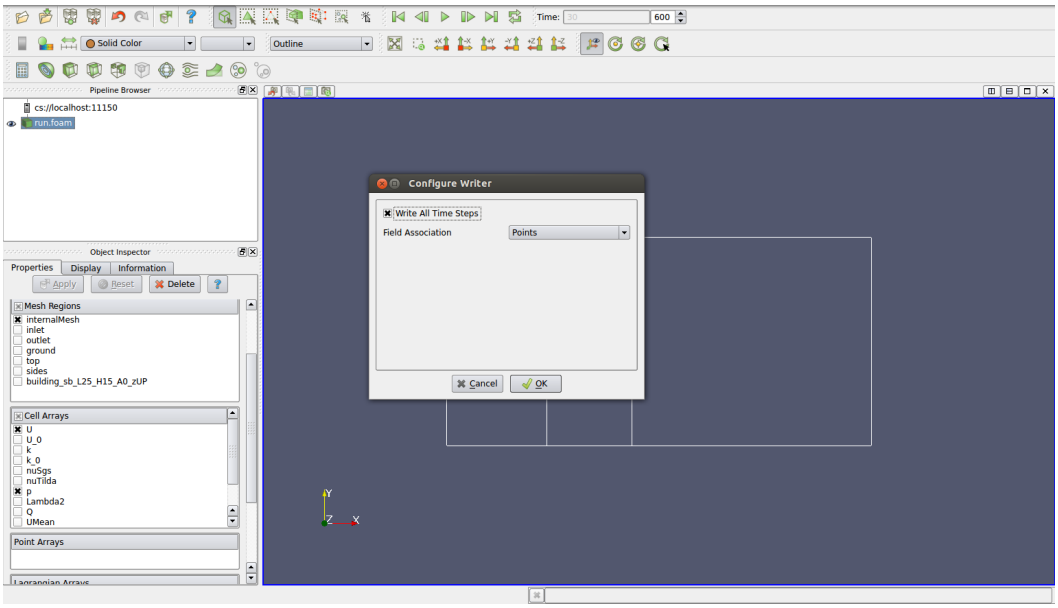


Figure 72: Save the data from the points and for all timesteps

After all .csv files have been generated they are trimmed and processed using the **extractData.py** and **csvImport.sh** scripts in the postProcessing sub folder. The python **extractData.py** script is used to trim each .csv file in space and time, and rename/renumber the remaining files. The commented script is shown in Listing C.1 with details on what should be mortified before execution. After modifying the script with; the fileName, start and end loop times, and desired wake region, the trimming is performed by entering *python extractData.py* form the command line.

Listing C.1: extractData.py

```
import csv
import os
import natsort
from os import listdir
from os.path import isfile, join

#############################################################################################
# extractData.py
#
# Mark Sutherland
# December 18, 2014
#
# Simple script to take the unprocessed .csv files from OpenFOAM/Paraview and trim them
# both in space and time. The spatial trimming is performed as a simple box by specifying
# the lower and upper bounds for x,y, and z. The temporal trimming is used for
# implementing the desired loop interval by specifying the start and end times.

#Ensure the exact selected times exsists, based on timestep save resolution. Also ensure
#the only saved varaibles in the .csv files are p,Ux,Uy,Uz,x,y,z and in that exact order.

fileName = "_____"          #.csv file names e.g. "730_6_0_1_"
dirPath = "csvFiles"        #Enter the directory where the csv files are

LES_simWriteInterval=0.05   #Save interval of CFD simulation
newStartTime=20.2           #Loop interval start time
LES_EndTime=28.9            #Loop interval end time

lower_x_bound = -6          #Define the box to keep the wind data. Usually taken 0.5m
upper_x_bound = 24.5        #inward of the wake refinment to ensure good wind data
lower_y_bound = -6
upper_y_bound = 6
lower_z_bound = 0
upper_z_bound = 21.5
#############################################################################################

#Setup the header of the csv file/database tables
newHeader = []
newHeader.append('p')
newHeader.append('Ux')
newHeader.append('Uy')
newHeader.append('Uz')
newHeader.append('x')
newHeader.append('y')
newHeader.append('z')

#Find the file numbers for the start and end loop inverval
timeStep=0
```

```
48 cutPoint =newStartTime/LES_simWriteInterval
49 lastTimeStep=LES_EndTime/LES_simWriteInterval
50 float(cutPoint)
51 float(lastTimeStep)
52
53 #Count the number of .csv files in the folder
54 onlyFiles = [f for f in listdir(dirPath) if isfile(join(dirPath, f))]
55 onlyFiles=natsort.natsorted(onlyFiles)
56
57 #Loop over all csv files for spatial and temportal trimming
58 for currentFile in onlyFiles:
59     #print "Working on "+currentFile      #Debug message
60     timeStep = currentFile.partition('.')[-1].rpartition('.')[0]
61     inFilePath=dirPath + "/" + currentFile
62
63     #Keep the file if in loop interval range and keep velocity data
64     #if in the defined box region
65     if float(timeStep) >= cutPoint and float(timeStep) <= lastTimeStep:
66       inFile = open(inFilePath, "rb")
67       reader = csv.reader(inFile)
68       outFile = open(dirPath + "/" + fileName + timeStep + ".csv", "wb")
69       writer = csv.writer(outFile)
70       rownum = 0
71       for row in reader:
72         if rownum == 0:
73             row = newHeader
74             writer.writerow(row)
75           #print ', '.jioin(row)      #Debug message
76         else:
77             if (float(row[4]) >= lower_x_bound and float(row[4]) <= upper_x_bound) and \
78             (float(row[5]) >= lower_y_bound and float(row[5]) <= upper_y_bound) and \
79             (float(row[6]) >= lower_z_bound and float(row[6]) <= upper_z_bound):
80                 writer.writerow(row)
81           #     print ', '.join(row)      #Debug message
82         rownum += 1
83       inFile.close()
84       outFile.close()
85     os.system("rm %s"%inFilePath)
86
87 #Recount the number of .csv files after temportal trimming
88 numFiles= [f for f in listdir(dirPath) if isfile(join(dirPath, f))]
89 numFiles=natsort.natsorted(numFiles)
90
91 #Renumber the remaining .csv files starting at one
92 count=1
93 for currentFile in numFiles:
94   os.rename(dirPath + "/"+currentFile,dirPath + "/"+fileName+str(count)+".csv")
95   count=count+1
96
```

```
97 #Script end message
98 print "Exraction Finished"
```

Once the .csv files have been trimmed they can be uploaded to the MySQL based Urban Wind Database using the **csvImport.sh** bash script, shown in Listing C.2. After editing the file for the database name, user name, user password[1], and base filename the script is run from the command line using *./csvImport*. If required, make the file executable by entering *chmod +x csvImport*. The script will go through each .csv file, make a new table for that timestep and load the data into the database. One important note are the saved variables and subsequently the database headers. The variables and their order are manual configured in both scripts. Ensure the number and order and uniform across the exported .csv files, **extractData.py** script and **csvImport.sh** script.

Listing C.2: csvImport.sh

```
1  #!/bin/bash
2
3  ##############################################################################################
4  # csvImport.sh
5  #
6  # Mark Sutherland
7  # December 18, 2014
8  #
9  # Simple script to take the processed .csv files from the trimming script and upload
10 # them to a MySQL database. The database must exist on the prior to upload and script
11 # must be run locally on the server.
12
13 #Ensure the variables below are filled.
14
15 #Define database connectivity
16 _db="------"              #Enter database name
17 _db_user="------"         #Enter user name
18 _db_password="------"     #Enter password
19
```

[1]The user names and passwords for the Kumomotojo database are removed due to the public availability of the scripts.

```
#Define directory containing CSV files
_csv_directory="./csvFiles"     #Enter the directory where the csv files are
_csv_BaseName="------"          #Enter csv base name, e.g. _csv_BaseName="730_6_0_1_*.csv"
#################################################################################################

#Go into directory
cd $_csv_directory

#Get a list of CSV files in directory
_csv_files=`ls -1 $_csv_BaseName`

#Loop through csv files
for _csv_file in ${_csv_files[@]}
do

  #Remove file extension
  _csv_file_extensionless=`echo $_csv_file | sed 's/\(.*\)\..*/\1/'`

  #Define table name
  _table_name="${_csv_file_extensionless}"

  #Make new table if it does not exist
    #Note: Not the best way to do this, the previous method tried to loop over the actual
    #headings in the file. Too much of pain, just change the variable and type if needed

  mysql -u $_db_user -p$_db_password $_db << eof
   CREATE TABLE IF NOT EXISTS \`$_table_name\` (
     p FLOAT, Ux FLOAT, Uy FLOAT, Uz FLOAT, x FLOAT, y FLOAT, z FLOAT
    ) ENGINE=MyISAM DEFAULT CHARSET=latin1
eof

 #Import csv into mysql
  mysqlimport --fields-enclosed-by='"' --fields-terminated-by=',' --ignore-lines=1 --local -u
    $_db_user -p$_db_password $_db $_csv_file

done
exit
```

Appendix D

Database-Simulator Integration

This appendix details how to remotely connect to the MySQL urban wind database for use with the Tara MATLAB flight simulator. Additionally, the naming convention of storing the CFD results and basic MySQL commands to interact with the database are provided. For the MATLAB/Simulink flight simulator to work the following toolboxes must be installed

- Aerospace Blockset
- Aerospace Toolbox
- Database Toolbox
- Fuzzy Logic Toolbox
- Model Predictive Control Toolbox
- Neural Network Toolbox

To see a complete list of the installed toolboxes, open MATLAB and use the *ver* command in the Command Window. Similar to the other Appendices, placeholders are used for sensitive details such as IP addresses, user names, and passwords which will be available as needed.

D.1 MySQL Database Structure

As outlined in Chapter 1, only a single building in isolation is used as an initial test of the quadrotor's performance with LES based wake fields. However, the urban wind database (UWD) currently contains both the RANS and LES generated wake field used in the results chapter. The database table naming follow a similar convention laid out by Galway [19] and are based on the defining parameters of the building, defined in Figure 8a and shown again in Figure 73.

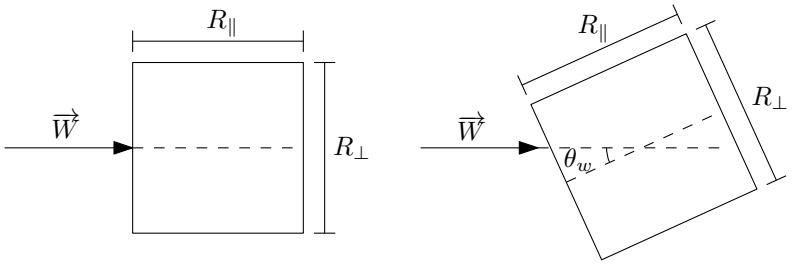


Figure 73: Single building characteristics

Therefore each database table takes the form of,

$$\mathrm{Re}_B\_\mathrm{Re}_E\_\theta_w\_R_\perp/R_\parallel\_\mathrm{T}$$

Where:

- Re_B is the base of the Reynolds Number
- Re_E is the exponent of the Reynolds Number
- θ_w is the wind angle
- $R_\perp/R_\parallel$ is the ratio of the building's side lengths
- T is the CFD simulation timestep number

For example, the 42nd timestep from the results used in this thesis would be *730_6_0_1_42* for the LES case and *730_6_0_1_RANS_42* for the RANS.

D.2 SSH Tunnel

Similar to the remote ParaView, a SSH tunnel is used to bridge the local flight simulator with the MySQL database[1]. This section will detail how to setup the SSH tunnel on Linux and Windows based clients.

D.2.1 Configure Linux Client

For a Linux based client the following alias can be added to the .bashrc file to simplify the tunnel startup,

alias **uwdTunnel**="ssh -f *usr*@XXX.XXX.XX.XX -L 7800:XXX.XXX.XX.XX:3306 -N"

Where *usr* is replaced with the UWD user name for the Kumomotojo server, and XXX.XXX.XX.XX os Kumomotojo's IP address. Port 3306 is the normal MySQL address and will be mapped to port 7800, the latter can be changed depending on the users existing port maps[2]. The flags -f, -N, and -L are used for; running ssh in the background, tells ssh to not enter any commands once the tunnel is open, and sets up an ssh tunnel that connects port 7800 on the localhost to port 3306 (default MySQL port) on the server respectively.

D.2.2 Configure Windows Client

For Windows based clients, Putty can be used to setup the SSH tunnel and port forward, found at http://www.chiark.greenend.org.uk/~sgtatham/putty/. Open

[1]Remote running of the simulator is future work
[2]Multi-connections, multi-users, and different port numbers are for future work

an instance of Putty, and click *Tunnels* under the *SSH* category, shown in Figure 74. Enter the source port, the IP address with MySQL server port, and click *Add*.

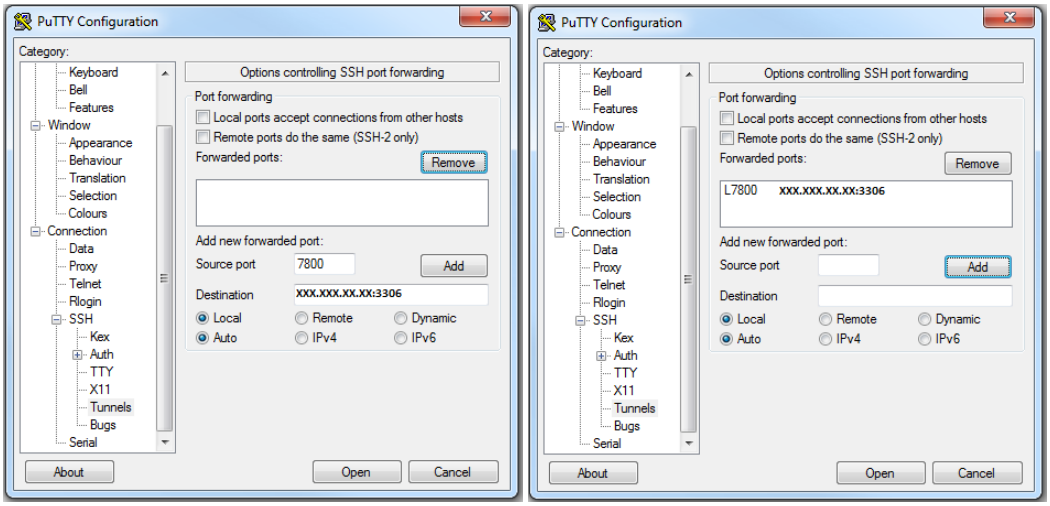


Figure 74: SSH tunnel Putty setup

Select the *Session* category, enter Kumomotojo's IP address, a desired name such as *windDB*, and *Save* shown in Figure 75. This will save the IP address and SSH tunnel information for future connections by selecting the session name and selecting *Load*. Start the SSH session by clicking the *Open* button[3].

[3]Both tunnels for the remote ParaView and the database can be added to the same session/name

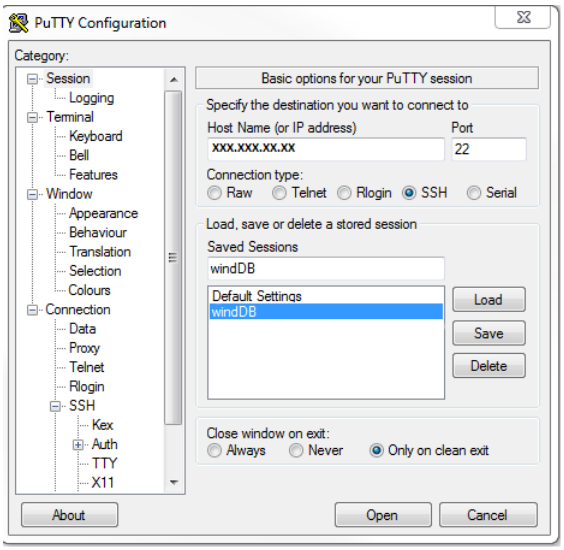


Figure 75: Putty connection setup

To start the tunnel either; open a new terminal and enter *uwdTunnel*, or open Putty and load the saved windDB configuration, if the client is Linux or Windows respectively. You should be prompted for a user name and/or password, enter your Kumomotojo user details or the provided user specific credentials. Open the tunnel before starting MATLAB and loading the Simulink model and leave the terminal open for the duration of using the simulator.

D.3 Driver Installation

The connection between MATLAB and MySQL is performed though a collection of java scripts bundled together into a java executable. Unfortunately this requires a driver installation and slight modifications to the MATLAB running on the client computer. The following steps outline downloading the java driver and how to install it, using the platform independent method.

1. The compressed files are available for download along side this tutorial. Alternatively the same compressed archive can be downloaded from https://dev.mysql.com/downloads/connector/j/ by selecting the *Platform Independent* option.
2. After opening the zip file, the *mysql-connector-java-5.1.30-bin.jar* must be moved to the java folder in root MATLAB directory. From the root MATLAB installation folder, where MATLAB was installed to on the client computer, the file path is **./java/jar/**.
3. To tell MATLAB where to look for the drive, a small edit is made in the Java class path. Open MATLAB and type *edit classpath.txt* in the Command Window. Ignore any autogenterator warnings that may appear.
4. At the very bottom of the *edit classpath.txt* add the following line: *$matlabroot/java/jar/mysql-connector-java-5.1.30-bin.jar*.

After a restart of MATLAB, the connector should allow the data in the MySQL database to be remotely accessed and applied in the flight simulator. Instructions on how to view the contents of the UWB and how to manually query it though MATLAB for testing purposes, followed by the details linking the actual Simulink simulator, are now presented.

D.4 Database Access and MATLAB Connection

The available database entries can be accessed by using the previously setup SSH connection to the host server. Next log into MySQL using the following command,

```
mysql -u userName -p
```

Where *userName* is the provided MySQL login name, not your Kumomotojo user name. After pressing Enter you'll be promoted for the provided MySQL password. You should get a welcome screen as shown in Figure 76 and a *mysql>* command prompt. From here you can view all of the databases stored on the server by entering **show databases;**, again illustrated in Figure 76. To actually access and view the wind data, enter a command such as **use windDB;** to change into that database.

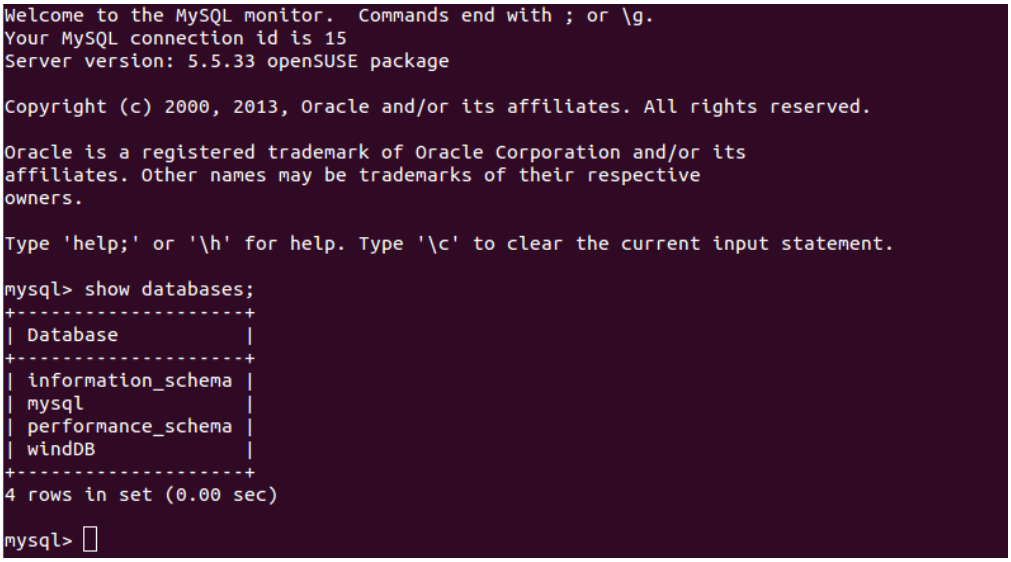


Figure 76: List of all the databases stored on the MySQL server

To list the tables in the windDB for example, use the **show tables;** command. This will return a huge list of all the tables in the database, a subset is shown in Figure 77. As previously outlined when producing the .csv files, each table represents a saved timestep of the CDF simulations. An appropriate save interval is found to be 0.05 seconds, Section 4.1.1, and loop intervals from the techniques in Section 4.1.2.

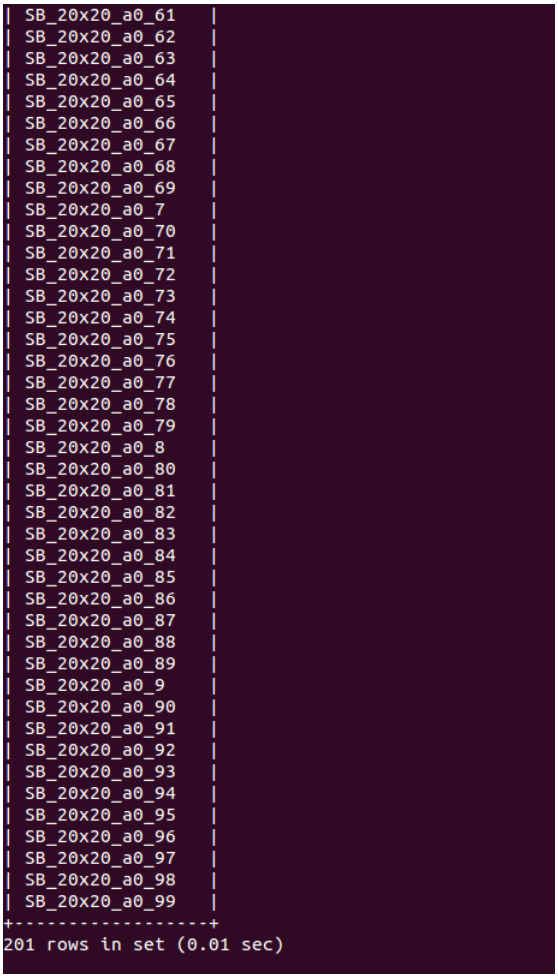


Figure 77: List of tables in windDB

To view the contents of a specific table enter a command such as,

```
select * from 20x20x120_a0_88 limit 10;
```

This command will select table 20x20x120_a0_88 and display all the columns and the first 10 rows of the table. The limit should always be used due to the huge volume of stored data in each table, this example has 1,322,025 rows. Figure 78 illustrates the results of the select command along with the structure of the stored data. Each

point in the flight area (x,y,z) has corresponding velocity components and a pressure value.

```
mysql> select * from SB_20x20_a0_88 limit 10;
+----------+---------+-------------+----------+--------+---------+--------+
| p        | Ux      | Uy          | Uz       | x      | y       | z      |
+----------+---------+-------------+----------+--------+---------+--------+
| -1.39088 | 13.8205 |   -0.118983 | -2.98557 | 34.375 |      15 |  4.375 |
| -1.58209 | 13.8236 | -0.00188551 | -3.29029 | 34.375 |      15 | 4.0625 |
| -1.82928 | 13.7887 |   -0.113196 | -3.64142 | 34.375 | 14.6875 | 4.0625 |
| -1.41211 | 13.8576 |   -0.184618 | -3.33488 | 34.375 | 14.6875 |  4.375 |
| -4.35744 | 14.1705 |    0.104261 | -4.21018 | 34.375 |      15 |    2.5 |
| -3.81308 | 14.3877 |    0.197164 | -3.17902 | 34.375 | 14.6875 |    2.5 |
| -2.72841 | 14.2313 |   0.0271365 | -3.25001 | 34.375 | 14.6875 | 2.8125 |
| -3.82949 |  14.794 |   0.0855556 | -3.87724 | 34.375 |      15 | 2.8125 |
| -2.70415 | 14.8412 |    -0.86827 | -3.77006 | 34.375 |      15 |  3.125 |
| -1.74024 | 13.7604 |   -0.771009 | -3.60255 | 34.375 | 14.6875 |  3.125 |
+----------+---------+-------------+----------+--------+---------+--------+
10 rows in set (0.00 sec)

mysql>
```

Figure 78: Query from the select and limit command ($t = 4.4s$)

To access this data in MATLAB as a test, start the SSH tunnel followed by MATLAB. Normally the *dbconnect.m* file in the flight simulator handles the database connection and will be setup to automatically connect when the simulator is started. The import parts of the file are used here to manually test the connection. Create a database connection object by entering the following command in the MATLAB Command Window.

conn = database('dbName', 'userName', 'passWord',
'com.mysql.jdbc.Driver', 'jdbc:mysql://localhost:7800/dbName')

Where dbName is the database name such as *windDB*, and *userName* and *passWord* are the authorized MySQL login details. Note the name of the database is required twice, at the beginning and end of the command. Then make an execution command by entering,

```
curs = exec(conn, 'select * from 20x20x120_a0_88 limit 10');
```

Actually query the data from the database,

```
curs = fetch(curs);
```

To visually see the pulled data,

```
curs.Data
```

If there are no error messages at any point and a 10 by 7 matrix should be returned and the SSH tunnel/MATLAB database integration is working. Similarly it should now be possible for a remote client to access the MySQL server at run time to return the urban wake velocities to the flight simulator.