OPTIMISATION OF COMPLEX GEOMETRY BUILDINGS BASED ON WIND LOAD ANALYSIS

Graduation Report

Erron Estrado
4716116

Optimisation of Complex Geometry Buildings Based on Wind load Analysis
Graduation Report

July 9, 2019

Author:
Erron J Estrado
4716116
MSc. Architecture, Urbanism, and Building Sciences
Building Technology Track

Mentors:
Dr Michela Turrin
AE+T | Design Informatics

Ir. Peter Eigenraam
AE+T | Structural Mechanics

Third Mentor:
Ir. Andreja Andrejevic
OMRT

External Examiner
Dr Ir. Andrej Radman
Architecture Theory

Abstract

One result of climate change is the increasing strength and frequency of wind events. This creates a problem for the also increasing number of high-rise buildings many of which are of unconventional shape. However, current methods for calculating wind response either do not account for these geometries, such as the Eurocode or are prohibitively expensive and time-consuming, such as physical wind tunnel tests. This thesis aims to address this issue by developing a computational method by which one can analyse the structural effects of wind on a building and optimise the external geometry to reduce those effects in the early design phase.

The method involves the combination of three main algorithms: Computational Fluid Dynamics (CFD) to simulate the wind and the pressure it exerts on a building, Finite Element Analysis (FEA) which calculates the structural effects such as deflection and stresses due to these forces, and an optimisation algorithm which can iteratively manipulate an input geometry to obtain better performance. For this thesis, a tool based on the method was developed in Grasshopper, the visual scripting plugin for Rhinoceros3D. Existing plugins were used for the main algorithms while custom scripting was used to combine them into a single tool that was made relatively easy to use and returned quick results.

The methodology involved extensive research into the various aspects of the method. This was followed by the development of the method throughout which testing and validation were performed to determine its accuracy and timeliness. Case study buildings were tested with the goal of reducing structural material use. In all tests, the mass of structural material needed was reduced by allowing the optimisation algorithm to manipulate only the external geometry of the building. This produced a tool within Grasshopper and a set of guidelines for developing such a method.

Keywords: Computational Fluid Dynamics; Optimisation; Finite Element Analysis; Wind engineering; Parametric design; Computational design

Acknowledgements

This document is the result of not only the final year of the Building Technology master's programme at the TU Delft but the culmination of a long academic journey.

I am very grateful to my mentors for their support. To Dr Michela Turrin for sharing her immense knowledge of optimisation and computational processes and for always pushing me to improve my overall research. To Peter Eigenraam for his guidance on the structural aspects of my work and showing me the importance of this topic. To Andreja Andrejevic and all my other colleagues at OMRT, thank you for giving me support in the development of the tool but also inspiring me to the potential of computational tools for the built environment. Special thanks to Adelya Doudart de la Grée for her invaluable guidance on CFD, to Christoph Waibel for his support in working with the GH Wind plugin, and Baojie He for providing wind tunnel results for the CAARC model. Thanks also to Hans Hoogenboom for setting up the computer to run the simulations.

I would also like to thank my parents whose hard work, love, and support has enabled me to have the opportunity to pursue my education and be what I am today. And to my sister, my loving girlfriend, and all other family and friends for their continuous encouragement and support of everything I do.

Finally, a great thanks to my beloved Biertje group: Agata, Alex, Valeria and Sofia. Thanks for being my second family in Delft, whom I could always count on to go through tough times and share the best moments of joy.

CONTENTS

Acronyms

ABL:	Atmospheric Boundary Layer
BF:	Butterfly
CAARC:	Commonwealth Advisory Aeronautical Research Council
CAD:	Computer-Aided Design
CDE:	Computational Design Exploration
CDO:	Computational Design Optimisation
CFD:	Computational Fluid Dynamics
CFD-O:	CFD-based Optimisation
CSO:	Cross-Section Optimisation
DES:	Detached Eddy Simulation
FDM:	Finite Difference Method
FEA:	Finite Element Analysis
FEM:	Finite Element Model/Method
FFD:	Fast Fluid Dynamics
FSI:	Fluid-Structure Interaction
FSIO:	Fluid-Structure Interaction based Optimisation
FVM:	Finite Volume Method
GA:	Genetic Algorithm
GH:	Grasshopper
LES:	Large Eddy Simulation
MOO:	Multi-Objective Optimisation
OPF:	Optimisation Problem Formulation
OPS:	Optimisation Problem Solution
RANS:	Reynold's Averaged Navier-Stokes
RBF:	Radial Basis Functions
SHM:	Snappy Hex Mesh

1 INTRODUCTION

When we think of loads on a building we commonly think of self-weight and the weight of people and fixtures as the main structural factors. However, wind can as well play a large role in structural design. Particularly for very tall and slender buildings which have inherent flexibility due to their form. The movements caused by wind can make buildings uncomfortable for occupants as well as cause damage to the building. This is particularly a problem for areas that are susceptible to strong winds, both on a regular basis and at certain times of the year such as hurricane-prone regions. This is only getting worse due to the effects of climate change. Since the 1980s the strength of North Atlantic hurricanes has been increasing. In the United States alone hurricanes and tropical storms have caused more damage than any other large-scale natural disaster since 1980 (Melillo et al., 2014).

Also on the rise, is technology and its impact on architecture and the building industry. Contemporary building aesthetics are becoming increasingly non-uniform as the growth in design and fabrication technologies enable us to create geometries that before were much more difficult to realise. Firms like UN Studio, MAD Architects, and Zaha Hadid Architects have been building more and more buildings of highly complex geometries. This has been made much easier to do with the rise of parametric design software and the integration of programming and scripting into the architectural design process. Programs like Rhinoceros 3D and Revit enable architects to bring their creative ideas to life. But what challenges do these designs bring for the engineering side of the equation? After all, these buildings have to meet the same code requirements as any other and wind loading is a big area of concern for these buildings.

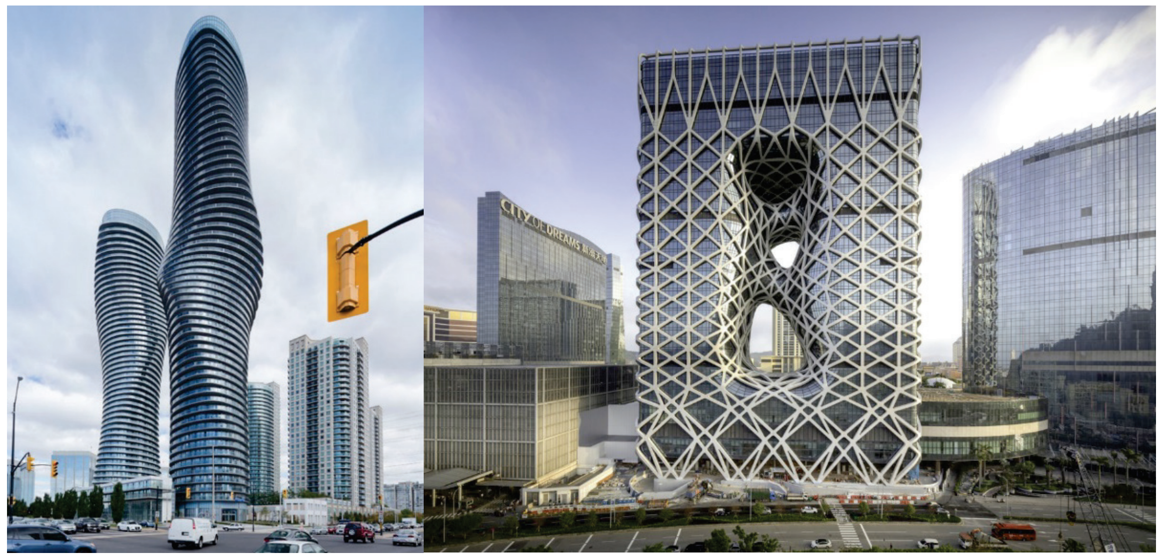


Figure 1-1: (Left) Absolute Towers by MAD Architects © Iwan Baan
Figure 1-2: (Right) Morpheus Hotel by Zaha Hadid Architects © Ivan Dupont

Many of the codes used today for wind loads are based on analytical methods developed in the 1960s. Since the flow of wind, like other fluids, is a highly complex dynamic condition these methods are based on simplifications and assumptions. In addition, many of these codes only provide guidelines for simple shapes like rectangles and circular cylinders and recommend physical wind tunnel tests for anything more complicated. Since technology has been applied so deeply into the design phase could we also use this increase in technology and computing power to solve this problem?

1.1 Problem Statement

The impact of climate change is being felt throughout the world and will continue to worsen in future. One way this is presenting itself is through stronger and more frequent windstorms. On the other hand, recent advances in technology have given rise to more buildings of atypical and very complex geometry, especially in high-rise buildings. However, current codes and calculation methods for wind loading are insufficient in that they do not adequately account for the geometry of these kinds of buildings and many of the equations are based on simplifications and approximations. This can lead to under/over design of the structure as well as uncomfortable building motions during wind events, particularly for slender high-rise buildings.

These calculations, particularly the Eurocode, can be quite tedious requiring many equations to be solved simply to arrive at a value of wind force for a single height and wind direction. This is very inefficient especially for the early stages of design where the building form is still being explored and may change many times in the process.

For this reason, wind load calculations tend to be done at a later stage of design by the structural engineers separate from the architects when the geometry of the building is already fixed. Thus, problems of wind load and wind-induced motions have to be dealt with at an element level rather than a building geometry level leading to compromises that can diminish the architectural intent.

For buildings of complex geometry, most codes recommend physical scaled wind tunnel testing. While this is very accurate it is often very expensive and takes a lot of time and effort to prepare the model and experiment setup.

Computational methods can be used to solve these problems however, differences in computer programs used necessitate time-consuming and complicated import/export of files where information can be lost and errors are prone to happen.

If the wind loading on a building massing can be deduced at an early stage in a simplified and integrated computational process, then the architect together with the engineers could analyse wind loads and use optimisation to generate options that minimise wind responses saving material in the structure and ensuring that neither the architectural nor structural design is compromised.

1.2 Objective

The main objective of this research is:

To develop a computational method for accurately analysing wind loads on a complex geometry building and optimising the geometry based on analysis results at an early design phase.

The idea is to have a method through which a tool could be created by which a user can gain insight into what kind of effects wind loads will have on a building of non-uniform complex shape and use that data to optimise the massing and generate options based on different building performance values. To do this, the tool should meet these requirements:

- **Ease of use:** This tool should be usable by architects and engineers who may have little background in Computational Fluid Dynamics (CFD) and Finite Element Analysis (FEA) but do have some experience with Computer Aided Design (CAD) and parametric design programs. It should be relatively simple to give inputs, run the procedure, and visualise results.
- **Single environment:** To aid in ease of use and save time, this tool should ideally accomplish all its tasks in a single environment. Users should not have to import and export models and other data to accomplish the task.
- **Rapid results:** This tool should be used at an early design phase where the geometry is still changing and evolving. Its results should help to inform those decisions. Thus, it should be able to perform its function in a relatively short time so as not to hold up the design process even with multiple iterations needing to be run.
- **Accuracy and precision:** With this simplification of use the tool should maintain an acceptable level of accuracy compared to more established methods like wind tunnel testing. More importantly, it should be precise so that there is a negligible deviation between the results of different iterations. This ensures the rapid reusability of the tool.

1.3 Research Questions

1.3.1 Main question

How can computational methods be used to accurately and efficiently calculate wind load on a complex geometry building and optimise the geometry to reduce wind responses in the early design phase?

1.3.2 Sub-questions

1. What are the existing methods for wind load analysis and how do they consider complex geometry buildings? Where do they fall short?
2. What kinds of geometries are more suitable for dealing with high wind loads?
3. What responses (deflections, vibrations, reaction forces) do building structures give to wind loading?
4. How can Computational Fluid Dynamics (CFD) be used to analyse the effects of wind on a building? How efficient is it compared to current calculation methods? How accurate is it compared to current calculation methods?
5. How can CFD, structural analysis, and optimisation be incorporated into a single, easy to use and efficient, computational process?
6. How can having accurate wind load analysis in an early design phase improve building performance?

1.4 Methodology

The methodology of this research is divided into 3 sections: Research, Develop, Verify. The goal is to develop a computational method in order to have a relatively fast and easy-to-use calculation of the effect of wind load on a building and optimise the geometry to improve structural performance. The way in which this will be done will be by combining three solvers namely a Computational Fluid Dynamics (CFD) algorithm, a Finite Element Analysis (FEA) algorithm, and an optimisation algorithm. CFD takes care of the simulation of the wind flow around the building and the pressure it exerts on the building surface and FEA calculates the structural effects such as deflections, moments, and forces. The combination of these two results in a Fluid-Structure Interaction (FSI) algorithm. Lastly, the optimisation algorithm will be responsible for manipulating the geometry to find the best performing option resulting in a Fluid-Structure Interaction based Optimisation (FSIO) method.

For this thesis, it was chosen to create and test this computational method by developing a tool based on it inside of Grasshopper, the visual scripting interface for Rhinoceros. Grasshopper was chosen because it is already well known to architects and engineers, it is relatively easy and quick to use, and can be supplemented with a vast array of plugins and self-written code in a variety of computer languages all in a single parametric environment. At the culmination of this thesis, the goal was to have a single Grasshopper script that incorporates CFD, structural analysis, and optimisation in a loop that can produce a more optimised form of an input massing as a proof for the viability of a computational FSIO method.

1.4.1 Research

This stage involves the literature review. Gaining knowledge in topics related to the problem to deduce what solutions exist, what has been done in this field, what existing methods can be used, and where current methods and technology is lacking.

It is divided into the following topics:

- Wind flow in the environment
- Wind actions on buildings
- Calculation of wind loading
- Computational Fluid Dynamics (CFD)
- Fluid-Structure Interaction (FSI)
- Optimisation methods

The conclusions drawn from this stage will allow for better setting of goals for the tools as well as boundary conditions for the development. A shortlist of software and methods that can be used and further evaluated will be chosen.

1.4.2 Develop

This stage focuses on the actual development of the method. It involves the joining of several smaller procedures to achieve the results desired from the method. These are as follows:

1. **Parametric building geometry**

   In order to facilitate easy modifications which are prevalent at the early stage of design this tool is aimed at, the input building should have a parametrically defined shape. A few variables should determine the overall building shape to be used further.

2. **Computational Fluid Dynamic (CFD) simulations**

   The impact of wind on a building is mainly due to the pressure exerted on the building surface. To obtain wind pressures on the façade of the building CFD will be used. This enables one to place a geometry in a virtual wind tunnel, simulate airflow, and calculate the effect that body has on the airflow and the effect the airflow has on the body namely the pressures exerted on the face.

3. **Evaluate the effect of wind pressure on the building structure**

   Results such as deflection at the top of the building and moment reactions at the base due to the wind need to be known. Thus, the wind pressures obtained from the CFD analysis as well as the building geometry will have to be converted to a form that can be analysed structurally to give these results.

4. **Incorporate optimisation**

   The optimisation portion involves changing the building geometry to improve wind response. The aim is to reduce wind pressures on the façade to lessen structural responses and properties like deflection, stresses, base moments, or mass of the structure. Thus, the optimisation will connect the resulting outputs from the structural analysis back to the building geometry parameters defined at the beginning to automatically manipulate the geometry, run the CFD analysis, the structural analysis, and evaluate the results in an iterative loop until an optimum shape is found.

As mentioned, it is desired for this thesis to develop this method within the Rhinoceros/Grasshopper environment. Thus, to achieve the above set out development goals plugins available for Grasshopper will be used combined with own scripting to connect the various parts. The imagined development procedure will be as follows:

1. Parametric building geometry
    a. Select test high-rise buildings of non-standard shape
    b. Create parametric models of the external massing
2. Computational Fluid Dynamics (CFD) simulations
    a. Select available CFD plugins and determine their setup, procedure, and results they can produce
    b. Evaluate them for precision, accuracy, and time by comparing to physical wind tunnel tests using a standard model
    c. Select a plugin based on that evaluation in addition to ease of use.
3. Evaluate the effect of wind pressure on the building structure
    a. Karamba3D, an FEA plugin for Grasshopper will be used to perform the structural analysis.
    b. Determine how the massing model and pressure loads from CFD will be transformed into a form that can be analysed by Karamba.
    c. Develop said translation procedure and connect all portions of the script to form a single FSI procedure.
    d. Test FSI procedure and verify results with hand calculations.
4. Incorporate optimisation
    a. Select optimisation plugins for Grasshopper based on algorithms researched.
    b. Incorporate into FSI procedure to form a Fluid-Structure Interaction and Optimisation procedure (FSIO).
    c. Test different combinations of input variables and objectives
    d. Compare to results from initial geometry

Since ease of use is greatly desired, efforts will be made throughout the development to make the script as seamless as possible. This includes minimising user input by automating parts of the script. Settings will be parametrised as much as possible so that they adapt automatically to any given building geometry. For those parameters that must be constant, testing will be done to determine the best option for precision, accuracy, and time.

1.4.3 Verify

In this stage the tool will be compared to existing methods to ascertain its comparativeness and if any improvement is made. Case study buildings of non-standard geometry will be used.

1. Calculate the wind loading using FSI method
2. Calculate the wind loading using Eurocode procedure
3. Compare results

Figure 1-3 shows the methodology for the research.

METHODOLOGY

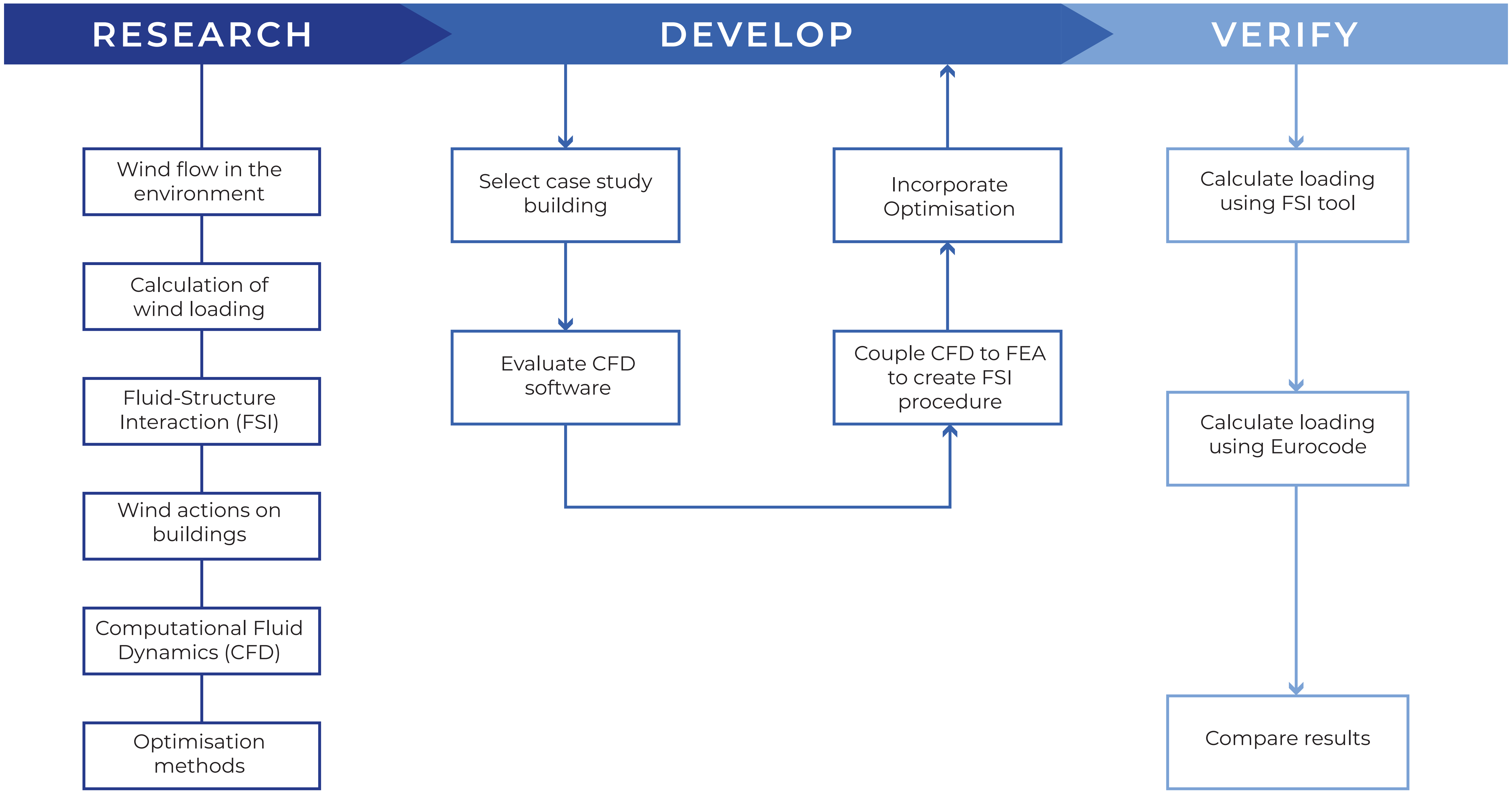

RESEARCH
DEVELOP
VERIFY
Wind flow in the environment
Calculation of wind loading
Fluid-Structure Interaction (FSI)
Wind actions on buildings
Computational Fluid Dynamics (CFD)
Optimisation methods
Select case study building
Evaluate CFD software
Couple CFD to FEA to create FSI procedure
Incorporate Optimisation
Calculate loading using FSI tool
Calculate loading using Eurocode
Compare results

Figure 1-3: Methodology diagram

2
RESEARCH

2.1 Wind Flow in the Environment

Wind in the environment is primarily due to temperature differences in the atmosphere which in turn cause pressure variations that cause air to flow from one part to another. Heat transfer from the equator to the cooler more northern latitudes and the forces of the earth's rotation are responsible for the major prevailing winds. On a smaller more local scale winds can vary widely in strength, direction, and frequency. Extreme events like storms and hurricanes can occur characterised by extremely high winds which can have a large impact on the built environment. Flowing wind exerts pressure on any surface it interacts with. Not only on perpendicular windward surfaces but those parallel and leeward as well eliciting a variety of structural responses, i.e. deflections, vibrations, and motions.

2.1.1 Boundary layer

Wind flow near the earth's surface is not smooth (laminar) but unsteady (turbulent). This is due to frictional effects of the earth's surface and its inherent roughness due to vegetation, orography (hills, cliffs, valleys, etc.), and buildings. This has the effect of slowing the flow of wind. This goes from zero at the earth's surface increasing logarithmically to a maximum value called the freestream. This layer of turbulent air is called the Atmospheric Boundary Layer (ABL). It is characterised by the Boundary Layer depth, the distance between the earth's surface and the beginning of the freestream where the friction of the earth no longer affects the flow of wind. Boundary layer depth depends on the roughness of the terrain below it. It can be very short for open countryside, to very high for cities (Figure 2-1).

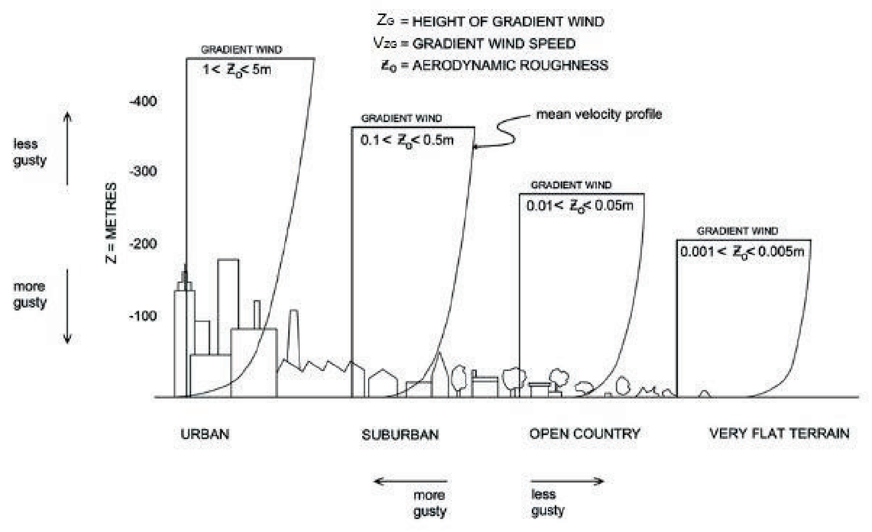


Figure 2-1: Boundary layer profiles (Cochran and ASCE. Committee on Structural Wind, 2012)

2.1.2 Roughness length

Roughness length, z_0, is a measure of the roughness of the earth's surface. It has a value equal to about 5-10% of the average height of the terrain roughness elements such as the trees, buildings, etc (Aynsley et al., 1977). This value is critical for accurate forming of the wind velocity profile for calculations as well as CFD and wind tunnel testing. It is usually given in table form based on the terrain by the relevant code as seen in Table 2-1 from the Netherlands National Annex to the Eurocode for wind actions on structures.

Terrain Category		z_0 m	z_{min} m
0	Sea or coastal area by the sea	0,005	1
II	Unbuilt area	0,2	4
III	Built area	0,5	7

Table 2-1: Terrain categories and parameters (NEN, 2011).

2.1.3 Turbulence

Wind, especially near the earth's surface is turbulent. It does not flow smoothly and steadily but its velocity constantly varies with time. This is due partly to thermal effects such as heated areas of air causing it to rise and collide with colder air but also due to mechanical turbulence. This occurs when airflow contacts obstacles such as trees, terrain, and buildings. The inertia of the wind causes its flow to deflect and often stumble over itself creating small circulating vortices called eddies. This turbulence is not locally contained but spreads to the rest of flow due to the collision of surrounding air molecules. Turbulent layers of air can be viewed as being governed by eddy viscosity which reflects the momentum transfer due to turbulence (Simiu and Scanlan, 1996). Turbulence is very important to wind applications for three reasons. Firstly, this causes structures to be subject to time-dependent rather than constant loads. Secondly, due to this fluctuating loading flexible structures may exhibit resonant amplification effects if the loading frequency matches its natural frequency. Lastly, The aerodynamic behaviour of structures depends strongly on the characteristics of the flow thus it is important that this is considered during physical and computational tests (Simiu and Scanlan, 1996).

Researchers found that the speed of the wind over time can be separated by subtracting out the steady component and then quantifying the fluctuating component which accounts for the short gusts above and below the average speed. Since these can be both positive and negative the root-mean-square (RMS) is found to give an absolute value (Cochran and ASCE. Committee on Structural Wind, 2012). This is the basis of the quasi-steady methods that many codes use for calculating wind load.

2.2 Wind Actions on Structures

2.2.1 Bluff and streamlined bodies

Buildings are considered bluff bodies. These are bodies which cause large separation of the airflow from windward to leeward side (Aynsley, 1999). As a result, the major proportion of drag comes from pressure drag, caused by the difference in pressure between the windward and leeward face, compared to streamlined bodies where the flow stays very close to the shape and causes mainly friction drag (Smits, 2018). Typically, bluff bodies are shapes with sharp corners, such as rectangular plan buildings, however, circular structures are also considered bluff since at high Reynold's numbers pressure drag dominates (Smits, 2018). Streamlined bodies, such as aeroplane wings, are made to allow the flow to smoothly re-join after separation. These shapes minimise pressure drag though friction drag is more of concern since the flow is in direct contact with the surface for an extended time. While it may seem that a streamlined shape may be always better it worth noting that streamlined profiles are optimised for a single or very small range of wind direction. Any small deviation in the angle of attack can cause a significant change in the magnitude and distribution of pressures across the surface (Smits, 2018). This is not ideal for buildings as they are stationary structures subjected to winds coming from varying directions. The geometry would need to be optimised in a way to allow for that.

2.2.2 Flow over a body

Consider a bluff body in the path of fluid flow, in this case, air. The air will flow around the body in a characteristic way depending on the shape. The flow can be divided into three parts:

1. Freestream flow – which is ahead and outside the influence of the surface where the flow is uniform.
2. Shear layers – the layer close to the body surface where velocity moves from zero at the surface to free stream velocity at the boundary. Also called the boundary layer.
3. Wake flow – the region behind a separated shear layer containing low-velocity eddy vortices. (Aynsley et al., 1977)

The free stream of air can be described using Bernoulli's equation:

$$p + \frac{1}{2}\rho V^2 = C$$

Equation 2-1: Bernoulli's equation

Where p = static pressure, ρ = density of air, V = velocity, and C is a constant. Bernoulli's equation is valid for steady (zero vorticity), inviscid (zero viscosity), and incompressible flows (Smits, 2018). Therefore, in the shear layer and wake flow, Bernoulli's equation is no longer valid.

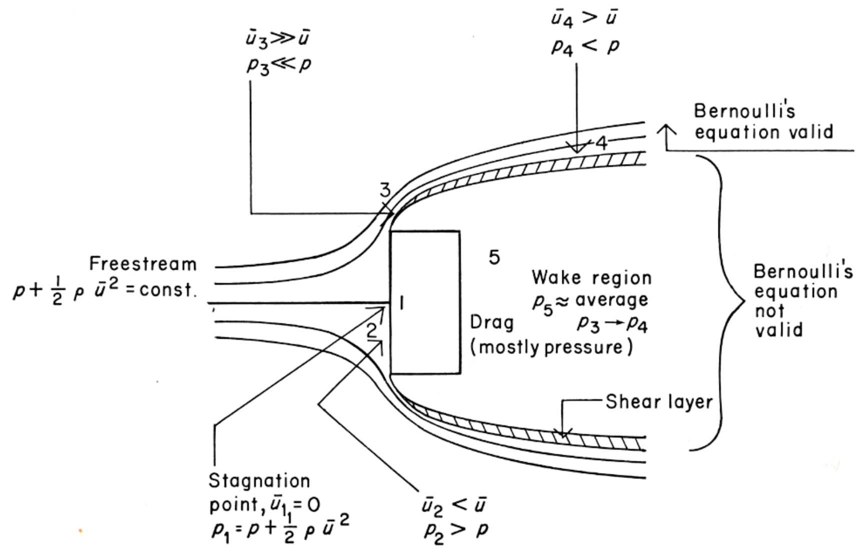

$\bar{u}_3 \gg \bar{u}$
$p_3 \ll p$
$\bar{u}_4 > \bar{u}$
$p_4 < p$
Bernoulli's equation valid
3
4
Freestream
$p + \frac{1}{2}\rho\,\bar{u}^2 = \text{const.}$
1
2
5
Wake region
$p_5 \approx$ average $p_3 \rightarrow p_4$
Drag (mostly pressure)
Bernoulli's equation not valid
Shear layer
Stagnation point, $\bar{u}_1 = 0$
$p_1 = p + \frac{1}{2}\rho\,\bar{u}^2$
$\bar{u}_2 < \bar{u}$
$p_2 > p$

Figure 2-2: Flow separation around a rectangular body in a free stream (Aynsley et al., 1977)

Flow separation occurs when the fluid particles near the surface are sufficiently decelerated by inertial forces from contact with the surface and the momentum of the flow above overcomes the cohesive viscous forces keeping the streamlines together (Simiu and Scanlan, 1996). This causes the flow at the surface to reverse forming eddy vortices that separate from the surface and form a free shear layer (Aynsley, 1999). This typically occurs at sharp corners or in the case of smoother shapes like a cylinder, the separation point as well as the characteristics of the flow, is dependent on Reynold's number. Reynold's number is the ratio of inertia forces to viscous forces in the flow (Holmes, 2007).

$$Re = \frac{\rho b V(z_e)}{\mu} = \frac{b V(z_e)}{\nu}$$

Equation 2-2: Reynold's number

Where b = streamwise chord length, $V(z_e)$ = fluid velocity at height z, and μ = dynamic viscosity, ρ = density, ν = kinematic viscosity = 1.5x10^{-6} for air (NEN, 2011). At low Reynold's numbers, viscous forces dominate, and the boundary layer is more laminar. There is smooth flow around the body. As Reynold's number increases the flow starts to separate and first forms large symmetrical vortices then as Re increases further, the vortices begin to alternate between each edge and are swept downstream forming what is called a Von Karman vortex trail. This phenomenon is called vortex shedding and can be particularly critical for tall buildings. At high Reynold's numbers like in most buildings, due to their large size and the low viscosity of air, a turbulent shear layer forms. This layer of rotating vortices separates the free stream laminar flow from the turbulent wake directly behind the building. The exact Reynold's numbers at which these different degrees of flow separation occur vary depending on the geometry of the building. For example, for a sharp-edged building shear layer formation happen at $Re >$ 1000 whereas for a circular cylinder this occurs at $Re >$ 5000 (Simiu and Scanlan, 1996).

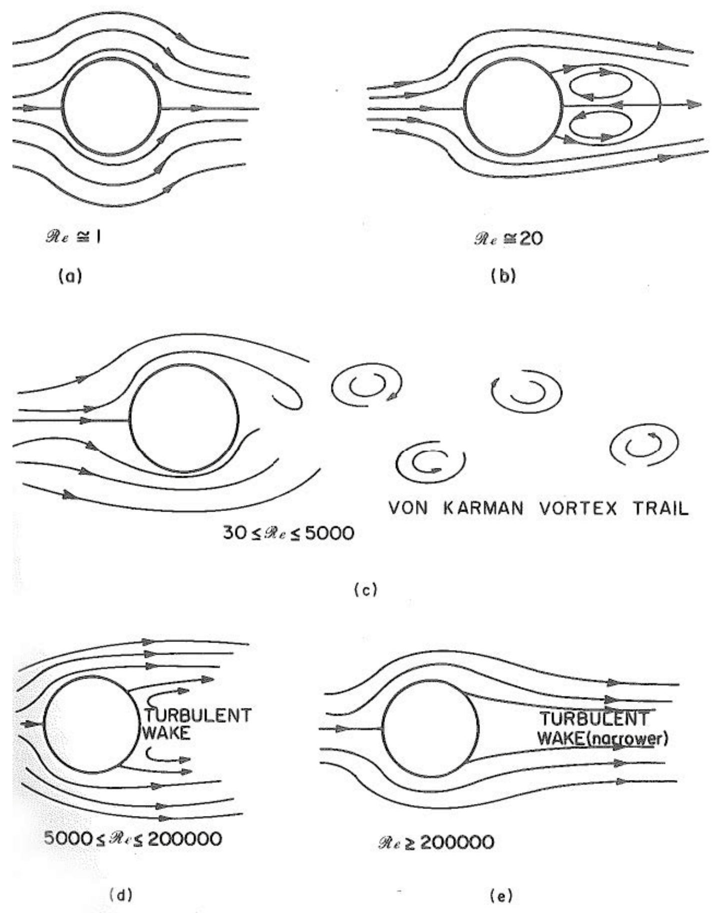


Figure 2-3: Flow separation at different values of Re (Simiu and Scanlan, 1996).

2.2.3 Wind forces

The loads imposed on a building in the flow of wind are divided into 3 categories (Cochran and ASCE. Committee on Structural Wind, 2012):

1. Along-wind loads
2. Cross-wind loads
3. Torsional loads

Along-wind

The along-wind force is caused by drag. Drag is the total force in the streamwise direction caused by the fluid flow on a body. It is made up of friction drag, caused by the viscosity of the fluid and its contact with the surface, and pressure drag, caused by the pressure distribution on the body. For bluff bodies such as buildings, only pressure drag is considered as the friction drag component is very small (Aynsley et al., 1977). This pressure is often expressed as a dimensionless coefficient, C_p.

$$C_p = \frac{P}{0.5\rho V^2}$$

Equation 2-3: Pressure coefficient

Where P = pressure, ρ = density, and V = the fluid velocity. Typically, a body experiences positive pressures (direction toward the body) at the windward face. The maximum positive pressure, around C_p = 0.9 for rectangular plan buildings, acts at the stagnation point seen in Figure 2-4. This is the point at the windward face where the velocity of the flow is brought to zero (Holmes, 2007).

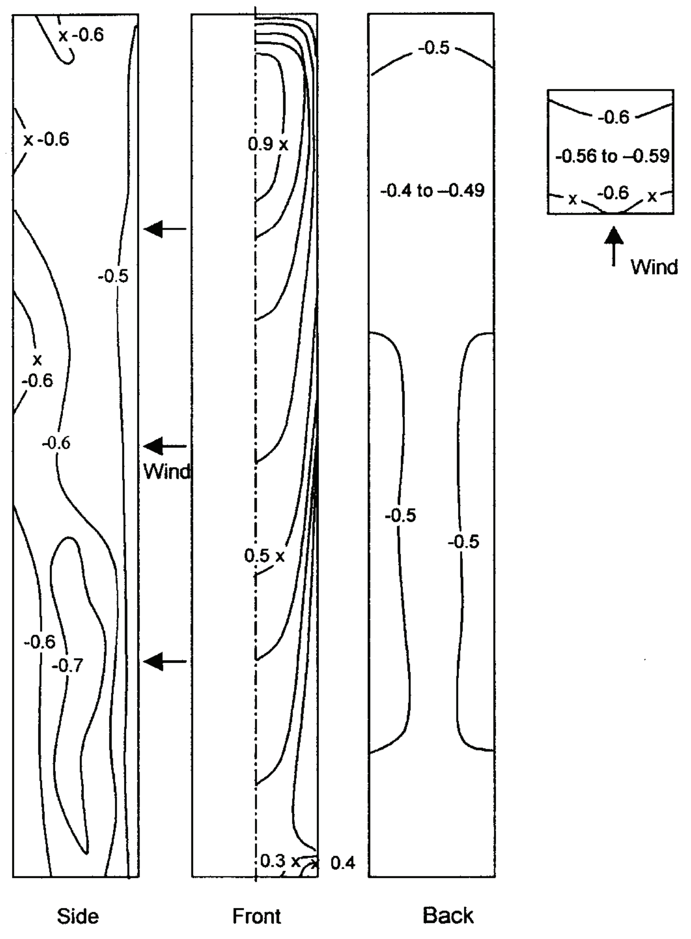


Figure 2-4: Pressure coefficient distribution on a rectangular prism (Holmes, 2007)

The leeward side of the building typically experiences negative pressures. However, there is no typical limit on the pressure coefficient for negative pressures (Aynsley, 1999). The drag force can also be represented as a dimensionless coefficient, C_d:

$$C_d = \frac{F_d}{0.5A\rho V^2}$$

Equation 2-4: Drag coefficient

Where F_d = Drag force and A = area normal to the flow. F_d can also be expressed per unit span with A being replaced by B = reference dimension. The drag coefficient, especially for curved shapes is dependent on Reynold's number since the position of flow separation is determined by the viscous forces as opposed to sharp-edged bodies where separation occurs at those points mostly regardless of Reynold's number (Holmes, 2007). As seen in Figure 2-5 below there is a sharp drop in drag in what is called the critical region, around Re = 2x10^5 to 5x10^5. At this range of Reynold's number, the flow transitions from laminar to turbulent at the boundary layer of the body. As Re further increases it comes to a maximum of about 1/3 of its original value (Simiu and Scanlan, 1996).

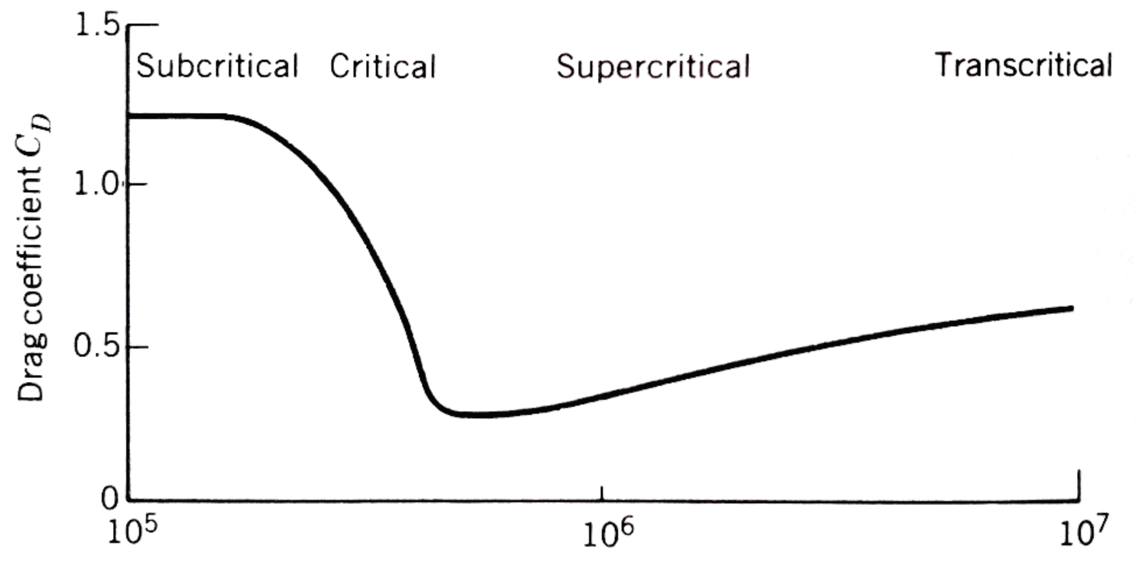


Figure 2-5: Variation of C_d with Re (Simiu and Scanlan, 1996)

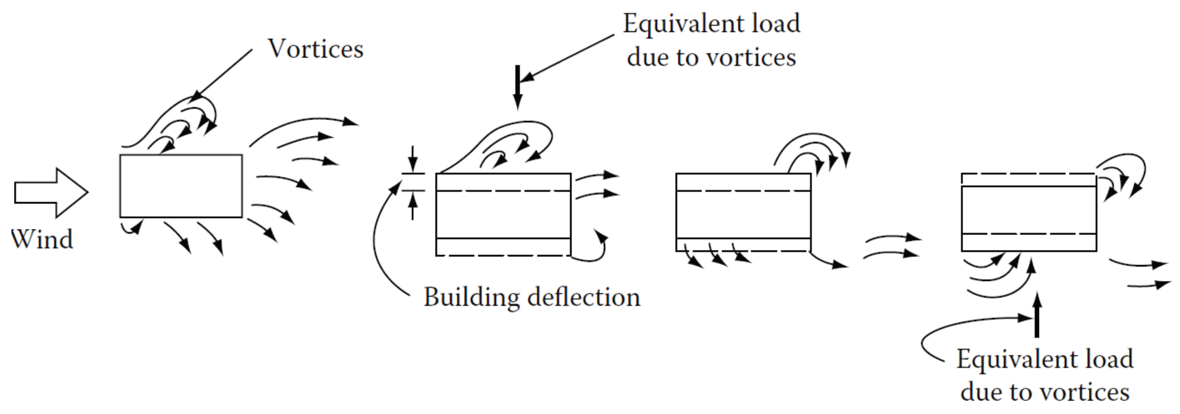
Cross-wind

The force experienced by a body in a direction normal to the wind flow is called lift (Simiu and Scanlan, 1996). While streamlining building geometry can reduce drag it may also increase lift forces which can be even more critical in buildings (Aynsley, 1999). For example, in low rise, large span buildings such as stadia and arenas the lift force in the vertical direction is usually the most critical loading as wind flowing over the roof can cause large negative pressures (suction) which can damage the roof structure and cladding (Simiu and Scanlan, 1996). Lift forces can also be represented by a dimensionless coefficient:

$$C_L = \frac{F_L}{0.5A\rho V^2}$$

Equation 2-5: Lift coefficient

In tall buildings, cross-wind lift forces in the horizontal direction are often more critical than along wind forces (Taranath, 2012). The most critical of these crosswind effects is vortex shedding which is the periodic shedding of vortices from alternating sides of the building (Figure 2-6).



Vortices
Equivalent load due to vortices
Wind
Building deflection
Equivalent load due to vortices

Figure 2-6: Vortex Shedding (Taranath, 2012)

Vortices are formed on the sides of the body parallel to the flow direction at the separation point. As shown previously, at low wind speed they are shed symmetrically but at high speed, the vortices break away from each side one after the other in a periodic way. As they break away they induce a force on the building normal to the surface which causes a vibration of the building (Taranath, 2012).

The frequency at which the vortices are shed is determined by the Strouhal number, S.

$$S = \frac{n_s D}{V_h}$$

Equation 2-6: Strouhal number

Where n_s = frequency of vortex shedding, D = body dimension normal to the flow, and V_h = flow velocity at height h. The Strouhal number is not constant but varies with wind speed and shape of the body and for circular and other smooth shapes, it varies with Reynold's number up to a limit of about 0.21 (Holmes, 2007). If the shedding frequency increases until it is within a range of about 10% of the natural frequency of the building, the building will resonate. Meaning, assuming low damping, it will vibrate intensely side to side as if it has zero stiffness. Further changes in speed will not affect the frequency as the vortex shedding is now determined by the displacement of the building and not the wind speed until the speed increases significantly. This is called lock-in and can cause extreme discomfort to occupants, and in some cases, structural damage to the building (Mendis et al., 2007).

Torsional Loads

In addition to horizontal displacements, wind loads can also cause buildings to twist around their axis causing torsional loads. Torsional loads and the resulting torsional dynamic response occur when there is a non-uniform pressure distribution over the face of a building, particularly in cases of unsymmetrical building geometry, and/or if the centre of mass and centre of rigidity do not coincide, for example, in a building with its core off to one side. Torsion can also occur in buildings that are partially shielded by another of similar height (Holmes, 2007). Torsional responses are not well studied or typically dealt with in building codes yet excessive torsion can particularly damage curtain walls and, just like cross-wind vibrations, cause great occupant discomfort (Cochran and ASCE. Committee on Structural Wind, 2012). The peak torque at the base of a building as a function of the wind speed V at height h, $T_{max}[V(h)]$, can be calculated by:

$$T_{max}[V(h)] = \psi\{\overline{T}[V(h)] + g_T T_{rms}[V(h)]\}$$

Equation 2-7: Peak torque at base

ψ is a reduction factor which accounts for the fact that the wind directions responsible for highest mean torque, $\overline{T}$, rms torque, T_{rms}, and most extreme conditions on the site will most likely never coincide. Thus in most cases $0.75 < \psi < 1$ (Simiu and Scanlan, 1996). Torsional peak factor, $g_T = 3.8$.

$$\overline{T}[V(h)] \simeq 0.038 \rho L^4 h n_T^2 V_r^2$$

Equation 2-8: Mean torque

$$T_{rms}[V(h)] \simeq 0.00167 \frac{1}{\zeta_T^{1/2}} \rho L^4 h n_T^2 V_r^{2.68}$$

Equation 2-9: Peak RMS torque

Where n_T = natural frequency and ζ_T = damping ratio. And:

$$V_r = \frac{V(h)}{n_T L}$$

Equation 2-10: Vr

$$L = \frac{\int |r| ds}{A^{1/2}}$$

Equation 2-11: L

A = cross-sectional building area, r = the torque arm of element ds, which is the perpendicular distance between the centre of rigidity and the centre of ds at the building boundary (Simiu and Scanlan, 1996).

2.2.4 Geometric strategies to reduce wind response

Wind loads are critical to tall buildings due to the array of static and dynamic responses that can occur in the structure. In fact, treating dynamic responses to keep a tall structure comfortable is often more difficult than ensuring structural strength (Irwin, 2009). The response of a building to wind depends not only on its shape but also its stiffness distribution, mass distribution and damping properties (Irwin, 2009). However, there are several strategies that designers can take to reduce these dynamic loads at the source, i.e. the geometry of the building.

Optimisation of the geometry can be done at an early design phase to reduce the occurrence and intensity of these loads.

Vortex shedding can be a big problem for very tall buildings. It causes discomfort, and in extreme cases damage to building elements. The source of vortex shedding is the building geometry and thus it can be severely reduced or eliminated by certain geometric strategies. Irwin (2009) gives the following design choices that can reduce vortex shedding:

- **Softened corners:** Eliminating sharp edges by rounding, chamfering, or stepping back the corners of buildings can greatly reduce vortex excitation. These should ideally extend about 10% of the building dimension.

- **Spoilers:** Façade elements such as vertical fins can help keep the flow attached to the building for longer. This shown in the thesis by Vongsingha (2015)

- **Porosity:** Placing openings through the building allowing air to flow through and disrupt or weaken vortices. An example of this can be seen in 432 Park Avenue tower in New York City by Rafael Viñoly Architects.

- **Tapering and setbacks:** Vortex shedding depends on Strouhal number which from Equation 2-6 can be seen varies with building width. If the building width varies with the height it causes vortices to be shed at different frequencies along the height thus causing incoherent shedding which greatly reduces the force compared to contiguous vortices along the height.

- **Varying cross-section shape:** Like tapering, this causes different vortex shedding frequencies along the height of the building.

The Burj Khalifa in Dubai, Figure 2-7, is a good example of these applications. It is a tapering tower made up of a collection of rounded tubes that step back along the height but also vary in height along the width of the building. This arrangement ensures that the vortices do not shed in an organised manner to induce a steady frequency of sway (Feblowitz, 2010)

Torsional loads can also be very uncomfortable for occupants. These can as well can be reduced by optimising the geometry by the following strategies:

- **Pressure distribution:** unsymmetrical geometries can cause pressure concentrations on areas away from the centre of stiffness forming a moment arm. The building geometry can be configured to ensure a more even pressure distribution (Cochran and ASCE. Committee on Structural Wind, 2012).

- **Alignment of centres:** Buildings have a centre of mass, where gravity acts, and the centre of stiffness, where lateral loads are mainly resisted. If the two centres are wide apart, for example in a building with an elevator core off to one side, this can cause torsional loads. The two centres should be kept close to concentric in order to minimise this (Holmes, 2007).

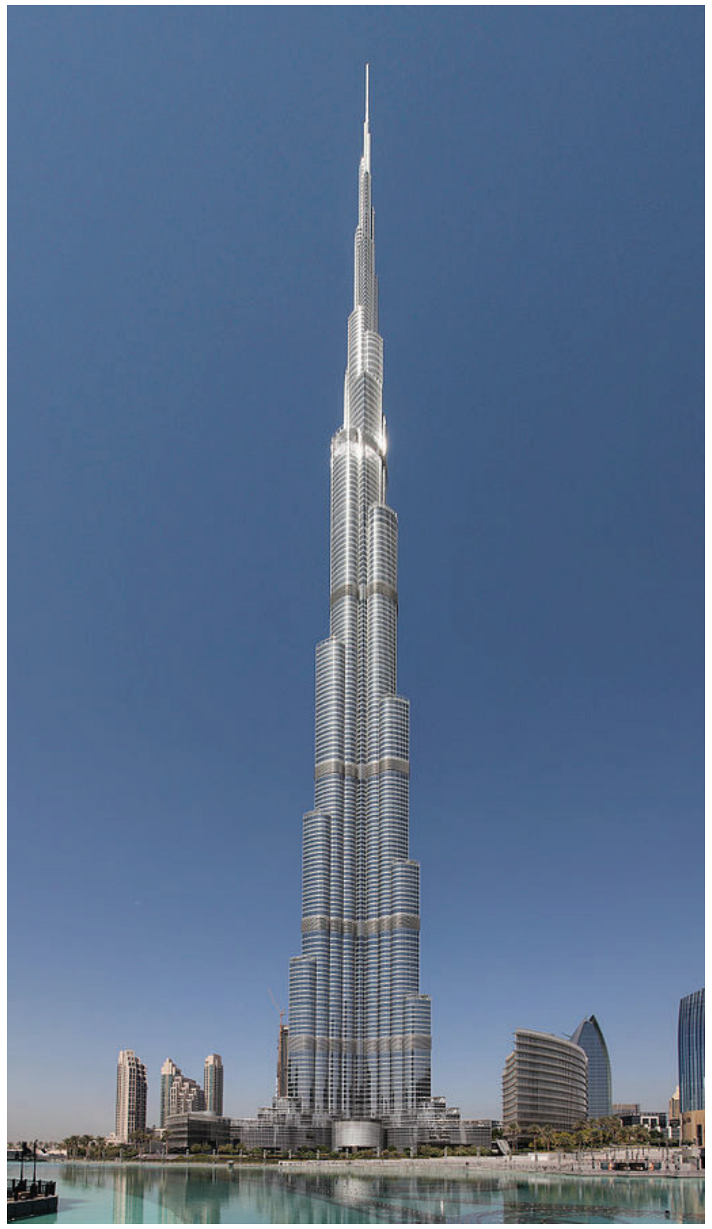


Figure 2-7: Burj Khalifa (Donaldytong, 2012)

2.3 Calculation of Wind Loading

Wind load calculations are inherently more complicated to deal with since, unlike most other building loads which are static, wind load is dynamic. It varies constantly with time due to the unsteady turbulent nature of wind. Alan G. Davenport and his work on using probability and statistics to develop an equivalent static function for calculation of wind loads in the 1960s led to the methods that we use today (Holmes, 2007). The 'quasi-steady' assumption is the basis of many modern wind load codes and standards such as the Eurocode. The quasi-steady method separates the dynamic wind speed into a steady, or mean, value and an unsteady, fluctuating, value. The proportion of this turbulent part is determined by the peak factor.

Maximum wind speed, $\hat{V}(z)$, is given by:

$$\hat{V}(z) = V_m(z) + g(t) \cdot \sigma_v(z)$$

Equation 2-12: Maximum wind speed

Where $V_m(z)$ = mean wind speed at height z, $g(t)$ = gust factor over duration t, and $\sigma_v(z)$ is the root mean square of the turbulence (Cook, 2007). This gives the turbulence intensity, $I_v(z)$ as

$$I_v(z) = \frac{\sigma_v(z)}{V_m(z)}$$

Equation 2-13: Turbulence intensity

This method, however, can be quite conservative especially for the design of tall, slender, flexible towers. Flexible buildings are defined as having a height to width ratio, $h/b > 4$, or a fundamental frequency of less than 1 Hz (Simiu and Scanlan, 1996).

2.3.1 Eurocode procedure

The Eurocode is based on this quasi-steady approach. Though some values like the pressure coefficients are based on the more complex Cook-Mayne methodology it is still brought back to the quasi-steady form for ease of calculation (Cook, 2007). EN1991-1-4:2005; Eurocode 1: Actions on structures - Part 1-4: General actions - Wind Actions, is the Eurocode section that sets out how wind loads on structures should be calculated, hereafter referred to as EN. In addition, many countries such as the Netherlands have a national annex that gives data and guidance to specific to the country such as wind speed maps and terrain

categories. These are called Non-Contradictory Complementary Information (NCCI) as it is not allowed to directly modify the Eurocode, only supplement it with information relative to the respective country. The EN also includes informative annexes with necessary information (Cook, 2007).

- **Annex A: Terrain Effects** gives illustrations of terrain categories, rules for transitions between roughness categories, rules of orography and effects of upwind buildings.

- **Annex B: Procedure 1 for determining the structural factor c_sc_d, Annex C: Procedure 2 for determining the structural factor c_sc_d, and Annex D: c_sc_d values for different types of structures** give two alternative calculation procedures and a graphical method for some types of structure, respectively, for determining the structural factor c_sc_d, the factor that describes the effects of structural size and dynamics on the wind actions.

- **Annex E: Vortex shedding and aeroelastic instabilities** gives rules for the vortex-induced response, including two alternative calculation procedures, and guidance on other aeroelastic effects.

- **Annex F: Dynamic characteristics of structures** gives guidance on the dynamic characteristics of linear structures – fundamental natural frequencies, mode shapes and damping.

Member states are required to either adopt the entire annex as normative or reject it. Thus, care must be taken to reference both the main Eurocode and the applicable national annex to ensure the correct data and procedure is used. For this thesis, the Netherlands National Annex to NEN-EN1991-1-4+A2+C1 (NEN, 2011) is used. Hereafter referred to as NA.

The EN describes wind loads as characteristic values which they define as "values with a characteristic annual risk of being exceeded of 0.02 in each and every year that the structure remains in service" (NEN, 2005). The basis of the calculations is the fundamental value of basic wind velocity, $v_{b,o}$, defined as the 10-minute mean wind velocity with a 0.02 annual risk of being exceeded, irrespective of direction and season, at 10m above the ground in terrain category II (NEN, 2005). Category II is defined as open country with low vegetation such as grass and isolated obstacles with separations of at least 20 obstacle heights (NEN, 2005). The NA gives the values for $v_{b,o}$ based on location on a map describing 3 different wind areas (Figure 2-8). Applying a directional factor and season factor gives basic wind velocity, v_b. However, the NA assigns a value of 1.0 to both of these factors, therefore, $v_b = v_{b,o}$.

The EN gives the main equations for wind pressure and wind force on a building surface. These equations have several unknowns that need to be calculated some of which are geometry dependent. EN section 7 has different subsections for different building geometries, roof types, and structure types which give methods of calculations for the coefficients.

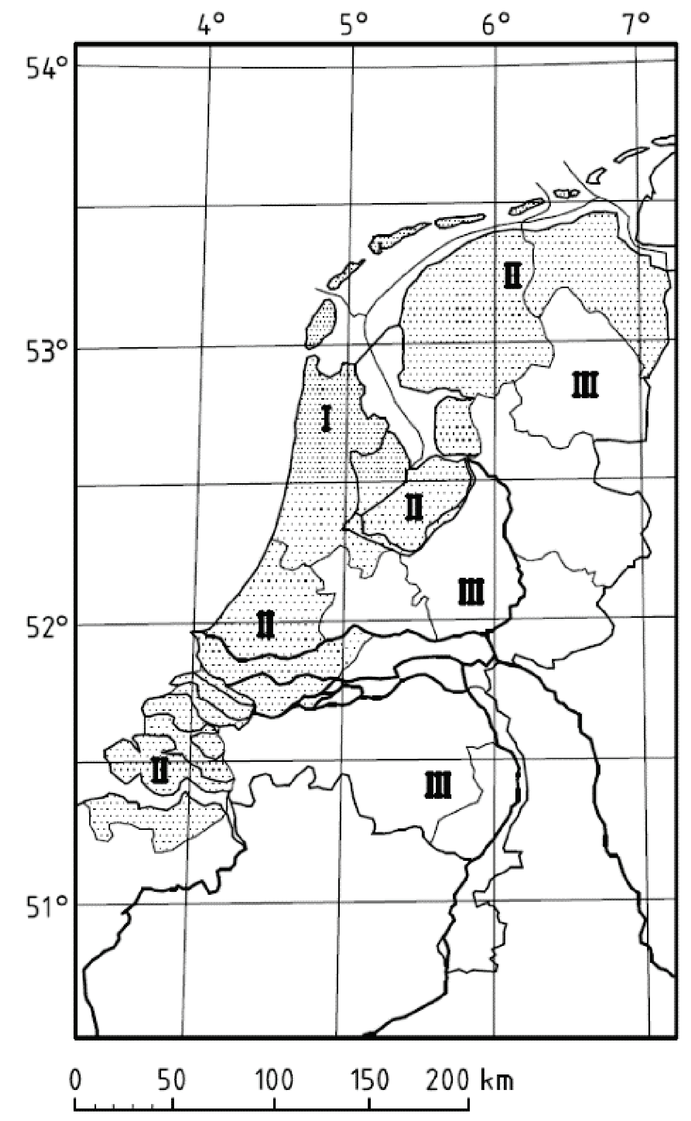


Figure 2-8: Classification of the Netherlands in wind areas (NEN, 2011)

To calculate the wind force, F_w, on a structure in the along-wind direction the EN gives the following equation:

$$F_w = c_s c_d \cdot c_f \cdot q_p(z_e) \cdot A_{ref}$$

Equation 2-14: Wind force on a structure

The force coefficient, c_f, is based on the shape of the building and given in section 7 of EN. $q_p(z_e)$ is the peak velocity pressure at reference height z_e. A_{ref} is the reference area on which the force is acting. This could be for a small element such as a cladding panel, or the whole building, in which case it should be the area normal to wind flow. One can also obtain the force per unit length by setting the desired dimension to 1, or the force per unit area by setting A_{ref} = 1. The structural factor, c_sc_d, accounts for the size effect and dynamic response described below. This factor may be split into two i.e. c_s and c_d, if c_d = 1.0 which occurs when if the building height is less than 50m and the ratio of height and width of the structure perpendicular to wind direction, h/b, is less than 5 according to the NA (NEN, 2011).

The size effect is the non-simultaneous action of peak wind pressures over faces of the structure and the dynamic response is the vibration of the structure in its fundamental mode due to the action of turbulence (Cook, 2007). Section 6 of the EN is dedicated to c_sc_d. Annex B and C give different methods for the calculation, while Annex D gives graphs from which you can determine structural factor based on building type and height, however, these values are very conservative (Cook, 2007).

The equation determines the dynamic response of a structure in the along-wind direction as the root-sum-square of a background component representing the quasi-steady response and a resonant component representing the dynamic oscillation at the natural frequency of the structure. This is known as the Davenport method and is implemented with some slight modifications in Annex B. Annex C uses a newer method from Dyrbye and Hansen which is somewhat simpler and gives values within 5% of Davenport's method (Cook, 2007). The NA requires the use of Annex C with the condition that the value of c_sc_d cannot be lower than 0.85. The equation for c_sc_d is as follows:

$$c_s c_d = \frac{1 + 2 \cdot k_p \cdot I_v(z_s)\sqrt{B^2 + R^2}}{1 + 7 \cdot I_v(z_s)}$$

Equation 2-15: Structural factor

Where k_p = the peak factor, B^2 is the background factor, R^2 is the resonant factor, and $I_v(z_s)$ is the turbulence intensity at reference height z_s. Note that the reference height $z_s = 0.6{\cdot}h$ is only valid for calculation of c_sc_d and is not to be used anywhere else in the wind force calculations.

In the EN turbulence intensity at height, z is given by:

$$I_v(z) = \frac{k_I}{c_0(z) \cdot \ln\left(\frac{z}{z_0}\right)}$$

Equation 2-16: Turbulence Intensity

k_I = the turbulence factor for which the recommended value by the EN and NA is 1.0. This assumes that RMS of k_I is constant throughout the building height, however, Cook (2007) shows that k_I should, in fact, be a function of height as RMS turbulence actually reduces with height. c_0 = orography factor which for flat terrain is equal to 1.0. In the case of terrain Annex A.3 of the EN should be used to calculate the value. z_0 = the roughness length given by table NB.3 - 4.1 of the NA.

The equations presented here so far are just a few of the equations needed to be solved in order to find the wind force on a building. An example calculation is shown in Appendix 5. About 20 equations need to be solved in order to obtain the wind force at a single height for one wind direction. Many of these are simplifications and approximations that lead to very conservative values (Cook, 2007). Furthermore, the EN procedure is only valid for buildings up to 200m in height (NEN, 2005). It only provides guidance for a limited number of building shapes; there is no procedure for unusually shaped buildings in the code. While the EN provides guidance on calculating forces due to vortex shedding in Annex E there is no provision for torsional loads. For these reasons, clause 1.5 of the EN states:

> In supplement to calculations, wind tunnel tests and proven and/or properly validated numerical methods may be used to obtain load and response information, using appropriate models of the structure and of the natural wind (NEN, 2005).

This statement leaves open the prospect for the use of CFD in wind engineering provided the numerical models are established and well validated.

2.3.2 Wind tunnel testing

Boundary layer wind tunnels remain the de facto testing method for buildings that do not conform to the restrictions of the EN. Testing scaled models in boundary layer wind tunnels can provide time-dependent surface pressures, including the complex cross-wind and torsional loadings crucial to tall buildings (Clannachan et al., 2009). However, they themselves have some inherent uncertainty in their results and care must be taken during the tests to ensure accuracy. Jensen (1958) showed that for scaled wind tunnel measurements the ratio of height, h, to

roughness length number, z_0, needs to be equivalent to ensure pressure measurements on the model in the tunnel would match those at full scale. Thus, h/z_0 came to be known as the Jensen number (Holmes, 2007). Scaling Reynold's number correctly is as well important for ensuring accurate values (Clannachan et al., 2009). This combined with the time and expense of wind tunnel testing makes them unsuitable for generating multiple optimal design iterations in an early project phase.

In a conversation on June 21, 2019, with Andy Mak, Bart Leclercq, and Josh Haigh, engineers at Aurecon in Dubai, UAE, it was discussed how wind tunnel testing is currently done in the context of high-rise and supertall buildings. In their projects wind tunnel testing is usually done. The longest time taken is waiting for a time slot to use the wind tunnel as they are usually very busy. Then comes the task of making the scaled model which today has been made a faster process with the advent of 3D printing. Then in all, the process of pre-processing, running the wind tunnel test, post-processing and getting the results can take an average of one to two weeks. What needs to be noted here is that this is only for a single building model. If the results turn out to be unsatisfactory, this process has to be repeated to test the performance of the new building model. The geometric strategies employed by engineers creating these buildings are usually based on general knowledge and rules-of-thumb acquired over the past 50 years of development in the field of wind engineering. While these can sometimes be sufficient at first there can be instances, especially with very unconventional designs, where even more needs to be done. What is commonly seen is that these wind tunnel tests are done at a later stage of design where the architect and client are very attached to the design and external geometric changes are not possible. This can result in having to increase the sizes or number of internal structural elements which then affect the architectural plan layout requiring changes.

This is where CFD can act as a complementary tool calculating wind loads at an early stage and generating optimal options which can then be verified later in the design by a scaled wind tunnel test and/or Eurocode calculations. Although wind tunnel testing is accurate it is not ideal for optimisation due to its physical nature relying on a trial and error approach. As Bernardini et al. (2015) state:

> Typically, wind tunnel tests are used to characterize the aerodynamic behaviour of the candidate shapes, selected a priori based on experience, therefore the number of configurations that can be considered is limited by the significant resources and time necessary to execute each test. As a consequence, a vast portion of the search space remains unexplored, and more conventional configurations are favoured over innovative solutions.

2.4 Computational Fluid Dynamics

A fluid is anything that flows to take the shape of its container such as water or air. It continuously deforms under the application of forces be that from gravity or external forces exerted on it. Fluid mechanics, or more specifically fluid dynamics, refers to the study of how these fluids move and the forces acting on them. Fluids are quite complex and can't be modelled as simply as solids can. Fluid dynamics is based on 3 principles:

1. Mass is conserved
2. $F = ma$ (Newton's second law)
3. Energy is conserved.

These, in turn, form the basis of the three governing equations of fluid dynamics - the continuity, momentum, and energy equations. These principles are generalised as a series of partial differential equations known as the Navier-Stokes equations (Wendt et al., 2009).

Computational Fluid Dynamics is the use of numerical methods to solve these governing equations (Mohotti et al., 2014). This involves subdividing the domain into a mesh of control volumes for which the solutions to the governing equations can be found. To enable this solution the continuous non-linear partial differentials, have to be replaced with an algebraic expression which gives a solution at a specific point. This process is called discretisation and can be done by either the Finite Difference Method (FDM), Finite Volume Method (FVM), or Finite Element Method (FEM). CFD software typically uses FDM (Anderson, 1995). CFD can describe many types of fluids and their flows. For the case of analysis of the effects of wind on a building, referred to as Computational Wind Engineering (CWE), we can refer to wind as an incompressible viscous flow. Though the viscosity is quite low it is necessary to take it into account to more accurately describe the flow separation and resulting turbulence at the boundary of the building. To analyse these flows a turbulence model is usually integrated into the solution. These use various algorithms to model the turbulent flow at boundary regions (Clannachan et al., 2009).

There has been a lot of research into the application of CFD for wind engineering problems over the last three decades which is accelerated by the continued advancements in computer technology and resources enabling faster and more detailed solutions (Clannachan et al., 2009). However, it is still not widely accepted by many codes as a method for wind load analysis. Most codes such as the Eurocode, United States' ASCE 7-10, and ISO 4354:2009 still do not explicitly mention the use of CFD for wind load. The Architectural Institute of Japan (AIJ) has published the AIJ guide for numerical prediction of wind loads on buildings (2008) which gives detailed advice on the use of CFD for wind engineering

purposes (Fransos and Lo Giudice, 2015). The fact that there is not a consensus on the use of CFD and CWE techniques shows that there are some shortcomings compared to traditional wind tunnel tests. Nonetheless, the benefits of CWE over time-consuming and expensive physical tests continue to inspire more research in the field.

To date, most of the research pertaining to CFD in the built environment has focused on natural ventilation, wind at pedestrian level, pollution dispersal, and other comfort aspects. Clannachan et al. (2009) state that "It has proven very difficult for CFD to acceptably model the complex flow interference phenomena induced from buildings." He further concludes that:

> This is the reason less work has been performed on predicting time-dependent surface pressures on these man-made bluff bodies. CFD has not developed enough to suggest it could replace wind tunnel testing in this respect. It does, however, offer encouraging potential to act as a complementary tool.

While verification should be performed by wind tunnel tests later in the design, CFD at this moment is very poised to be a useful tool at an early design phase when the geometry is still preliminary and a small amount of error in the results is acceptable. In the time since Clannachan's paper was published to the present day, it is known that computational power has increased immensely, thus it is reasonable to assume that CFD applications have also accelerated in their effectiveness.

2.4.1 Navier-Stokes equations

Mathematician Leonhard Euler was the first to develop equations to describe fluid flow in the 18th century (Hosch, 2018).

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = \frac{\nabla P}{\rho}$$

Equation 2-17: Euler equation describing fluid flow in modern notation

Where u is the fluid velocity vector, P is the fluid pressure, ρ is the fluid density, and ∇ indicates the gradient differential operator. However, Euler's equation defines inviscid flow; it completely neglects the effects of viscosity, mass diffusion, and thermal conductivity (Anderson, 1995). In the 19th century, French engineer Claude-Louis Navier and British physicist Sir George Gabriel Stokes independently developed the system of equations known today as the Navier-Stokes equations.

These equations expanded on Euler's work by including the effects of viscosity (friction). They can be represented in both conservation, based on an infinitesimal element fixed in space, and non-conservation form, based on an infinitesimal element moving with the flow (Anderson, 1995).

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = \frac{\nabla P}{\rho} + \upsilon \nabla^2 u$$

Equation 2-18: Navier-Stokes equation in modern notation

Where υ = the kinematic viscosity, and ∇^2 is the Laplace operator. See Appendix 1 for the full system of Euler and Navier-Stokes equations.

The Navier-Stokes equations are a coupled system of non-linear partial differential equations, and hence are very difficult to solve analytically. To date, there is no general closed-form solution to these equations that we know of (Anderson, 1995). This plays a big role in making fluid dynamics and by extension CFD a very complicated field.

2.4.2 Turbulence models

Wind flow, especially around bluff-bodies, constitutes of the free-stream and the turbulent regions of the boundary layer and wake. CFD can discretise and solve the Navier-Stokes equations for all these regions in what is called Direct Numerical Simulation (DNS). However, since turbulence is inherently very random and complex, DNS necessitates an excessively fine grid as each control volume for calculation must be smaller than the smallest eddy in order to fully capture the turbulent effect. This makes DNS a very computationally expensive and inefficient process (Clannachan et al., 2009). Thus, CFD calculations can be supplemented by a variety of turbulence models which more efficiently account for these conditions. However, at present no turbulence model is perfect and the selection of the most appropriate model depends on what is being analysed and forming a balance between accuracy and computational cost (Clannachan et al., 2009). Below we will examine some of these turbulence models and their methods and seek to determine which ones are most promising for the goals of this thesis.

The two leading turbulence models are Large Eddy Simulation (LES) and Reynolds Averaged Navier-Stokes (RANS), also known as Reynold's Average Stress (RAS) with each employing a different approach. LES is a transient method using a spatial filtering technique where all eddies larger than a certain size are calculated while those smaller are modelled in a sub-grid scale (SGS). Whereas, RANS is a steady-state time-averaging technique where, over a certain time period, the velocity vector is split into a mean and fluctuating part. Only the mean part is calculated but the effect of the fluctuating component is modelled on the flow (Clannachan et al., 2009). RANS gives only mean values whereas LES can give peak values and a more accurate picture of the transient phenomena of the flow at a particular time step. To achieve this LES needs a much finer grid and a higher number of time steps than RANS resulting in a much more computationally expensive process (Fransos and Lo Giudice, 2015).

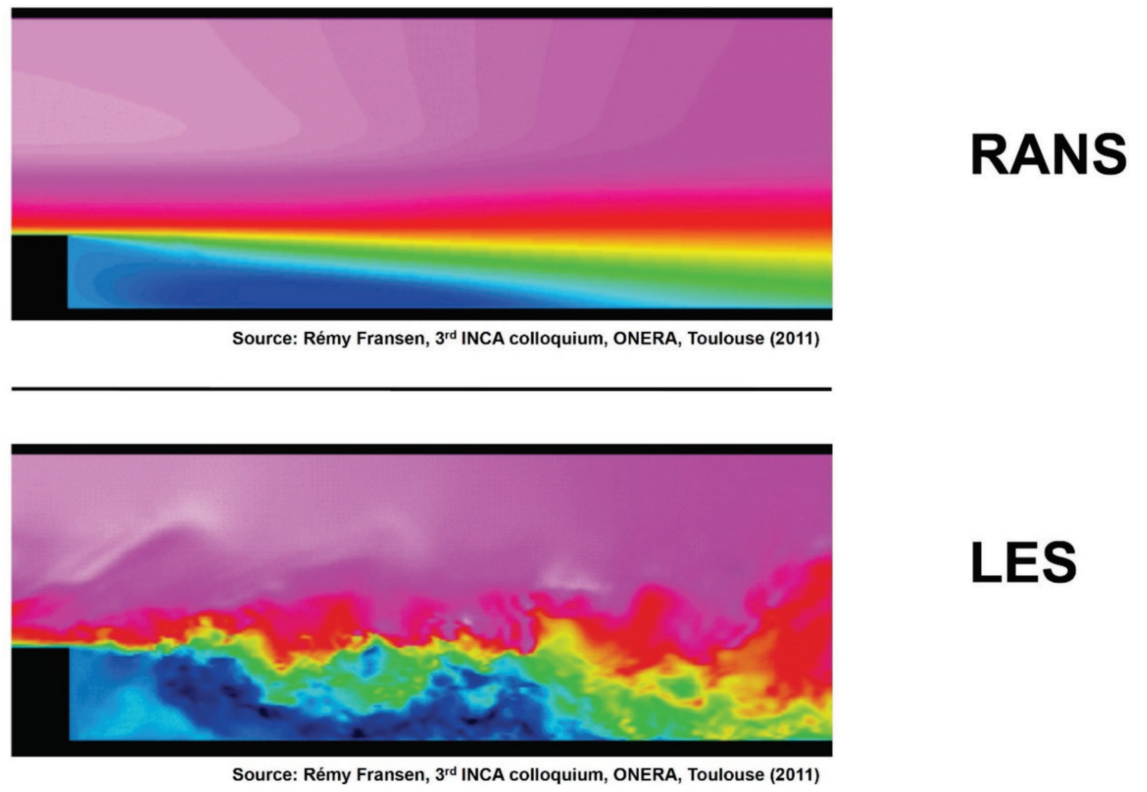

RANS
Source: Rémy Fransen, 3rd INCA colloquium, ONERA, Toulouse (2011)
LES
Source: Rémy Fransen, 3rd INCA colloquium, ONERA, Toulouse (2011)

Figure 2-9: Fluid flow calculated by RANS vs. LES turbulence models

The most common RANS methods are the $k-\varepsilon$ and k - ω models. These methods involve solving two additional equations to obtain the turbulent viscosity which involves solving for the kinetic energy, k, and either the turbulent dissipation rate, ε, or the specific dissipation rate, ω (Clannachan et al., 2009). The $k-\varepsilon$ is known to be unreliable in the turbulent regions around bluff bodies. Tamura et al (2008) performed CFD analyses using various turbulence models and compared the results to those from wind tunnel tests. They found that $k - \varepsilon$ overpredicted pressure coefficients at the front face of the building while underpredicting suction at the rear. Clannachan et al. (2009) also found that $k - \varepsilon$ consistently over-predicted the wake reattachment length and severely under-predicted the level of turbulent kinetic energy. Huang et al (2007) performed tests comparing the standard $k-\varepsilon$ model to ones with modifications proposed by Launder and Kato (LK) and Murakami et al. (MMK) as well as LES. They found that the standard $k-\varepsilon$ under-predicts the drag force coefficient C_d by about 20% with LK, MMK, then LES giving gradually more accurate results.

The $k-\omega$ model is known to perform better for near-wall flows, however, it suffered from inaccuracies in the freestream flow. Sun et al. (2009) used the $k-\omega$ model in their paper exploring Fluid-Structure Interaction (FSI) of airflow over a bridge deck. They concluded that the $k-\omega$ model is potentially well suited to simulating Vortex Induced Vibration (VIV) and flutter of bridges because of its superior performance in near-wall flow simulation compared with $k-\varepsilon$.

The Realisable $k-\varepsilon$ model was developed by Shih et al. (1994). Realisable means that, unlike the standard model where the ε is determined based on reasoning, the model satisfies mathematical constraints consistent with the physics of turbulent fluid dynamics. This is chiefly done by making the empirical constant $C\mu$ variable with the flow instead of constant as in the standard $k-\varepsilon$ (Rahman et al., 2007). van Hooff et al. (2011) chose to use the Realisable $k-\varepsilon$ in their study of wind flow around and wind-driven rain on stadia because of its 'significant improvements over the standard $k-\varepsilon$ model.' Another reason it was chosen was that it is widely validated for a wide range of flows including turbulent separated flows like those around bluff bodies.

RNG $k-\varepsilon$ is another variation of the standard RANS model developed by Yakhot et al. (1992) using Renormalisation Group (RNG) method to renormalize the Navier-Stokes equations, to account for the effects of smaller scales of motion thus making $k-\varepsilon$ more accurate. Clannachan et al. (2009) found that the RNG $k-\varepsilon$ gave results closest to full-scale physical tests at the CWE 2000 completion. Blocken (2014) stated in his paper, "the best agreement with the PIV wind-tunnel measurements by Karava et al. (2011) was obtained by the SST $k-\omega$ followed by the RNG $k-\varepsilon$ model." However, the aim of that experiment was testing natural ventilation.

The Shear Stress Transport (SST) $k-\omega$ model developed by Florian R. Menter (1994) combines the best of $k-\varepsilon$ and $k-\omega$ using the latter for near body flow and the former for flow further away by use of a transitioning function (Menter, 2009). This, along with an added cross-diffusion term in the ω-equation, gives SST $k-\omega$ better performance than both the standard and realisable $k-\varepsilon$ models.

In LES the major model is the Smagorinsky model. Others include the Smagorinsky-Lilly, Wall-Adapting Local Eddy-viscosity model (WALE) and dynamic SGS kinetic energy model. Huang et al. (2007) recommend the use of dynamic-SGS for high Reynold's number bluff body flows. It gave more accurate results than the RANS models in his assessment.

Detached Eddy Simulation (DES) is a hybrid of RANS and LES. It sought to combine the benefits of the two using LES at areas of flow separation and RANS for other areas (Clannachan et al., 2009). In a comparison of steady and unsteady RANS and DES models, DES showed notably better prediction of mean loads. However, it was concluded that the slight increase in accuracy was not worth the much greater increase in computational time (Clannachan et al., 2009).

As is shown there are several models to choose from each with their pros and cons. The choice of which to use is a complicated balancing act. Fransos and Lo Giudice (2015) mention that the choice of CFD application should be based on goals. They give two criteria:

1. The choice should not be based on model availability. They state that:

   > The choice of a software product is in fact not as relevant, as long as it allows for proper models and boundary conditions and the user has performed a rigorous validation before using it in a design context.

2. The choice should not be based on the pursuit of an unconditionally "best" CFD. This is based on the fact that there is not a single optimal model for every scenario.

In the AIJ guide, Tamura et al. (2008) recommend that RANS only be used for finding time averaged wind forces. For max wind forces, the RANS results should be multiplied by a suitable Gust Effect Factor. Clannachan et al. (2009) state that the benefits of RANS for tall building problems are restricted since it is generally the crosswind and torsional loads that are critical. An unsteady RANS model, such as Realisable or RNG $k-\varepsilon$, or LES would give more accurate results for the flow around the building although RANS models are significantly more efficient than LES. RANS simulation runtime can be one order of magnitude lower than simulations on LES models (Fransos and Lo Giudice, 2015). Indeed Clannachan et al. (2009) stated in a particular experiment that although LES gave results close to field measurements compared to RANS, the solution time for the LES simulation was 160 hours, whereas the RANS solutions ranged between 15 minutes and 6 hours. Sun et al. (2009) state that:

> On the one hand, the LES based approach is accurate and has been validated by comparison with experimental results. But LES is still too sophisticated and computationally intensive to be a viable tool for general engineering FSI applications.

Thus, it appears a balance must be struck between accuracy and computational cost and the choice of turbulence model plays a big role in that regard.

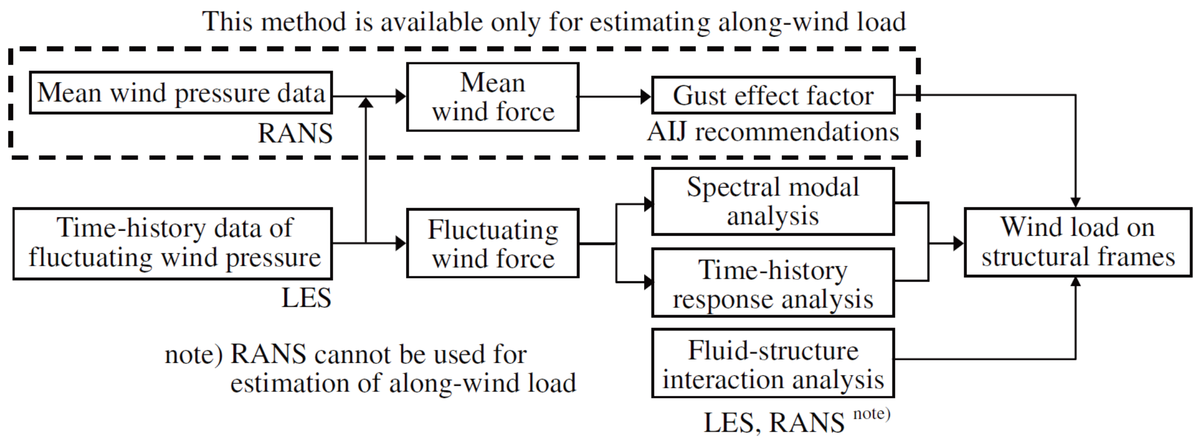

This method is available only for estimating along-wind load
Mean wind pressure data
RANS
Mean wind force
Gust effect factor
AIJ recommendations
Time-history data of fluctuating wind pressure
LES
Fluctuating wind force
Spectral modal analysis
Time-history response analysis
Wind load on structural frames
note) RANS cannot be used for estimation of along-wind load
Fluid-structure interaction analysis
LES, RANS note)

Figure 2-10: CFD Procedure for estimating wind loads on structural frames (Tamura et al., 2008)

2.4.3 Fast Fluid Dynamics

Another development in the area of CFD, particularly to address the long calculation times of traditional CFD models is Fast Fluid Dynamics (FFD). This procedure was first proposed by Jos Stam (1999) for rendering fluids in video games. As a result, the simulations are more focused on visuals and speed of simulation than accuracy. Nevertheless, it has been used by researchers for CWE applications particularly in tandem with an optimisation algorithm. FFD solves the Navier-Stokes equations using fully implicit and lower order methods and decouples the pressure and velocity components. This gives linear equations that are much easier and faster to solve (Waibel et al., 2017). Another advantage is that FFD is stable, meaning that it can take much larger time steps than CFD without worrying about the simulation blowing-up (Stam, 1999). This also allows for much faster convergence which is essential for optimisation algorithms which may run hundreds of variations. This simplification of the Navier-Stokes equations, however, leads to inaccuracy. One source of inaccuracy is FFD's inability to predict turbulent flows (Chronis et al., 2011). Indeed, Stam himself mentioned in his paper that FFD may not be suitable for engineering applications as it suffers from too much numerical dissipation. In other words, the flow dampens too quickly compared to real-world experiments.

Chronis et al. (2011) used a custom FFD code based on Stam's research together with a genetic algorithm to optimise a freeform surface based on pressures due to wind load. They were aware of the drawbacks of FFD regarding accuracy but stated that the aim of their experiment was not to "simulate physical phenomena with maximum accuracy but rather to investigate the potential of a resource effective simulation scheme in a conceptual stage generative approach." They were able to successfully optimise the geometry of a free-form NURBS surface by using FFD to find the surface pressures and moving the control points first in 1

degree of freedom, then, 3 degrees of freedom at each iteration of the genetic algorithm to obtain a lower mean pressure on the surface.

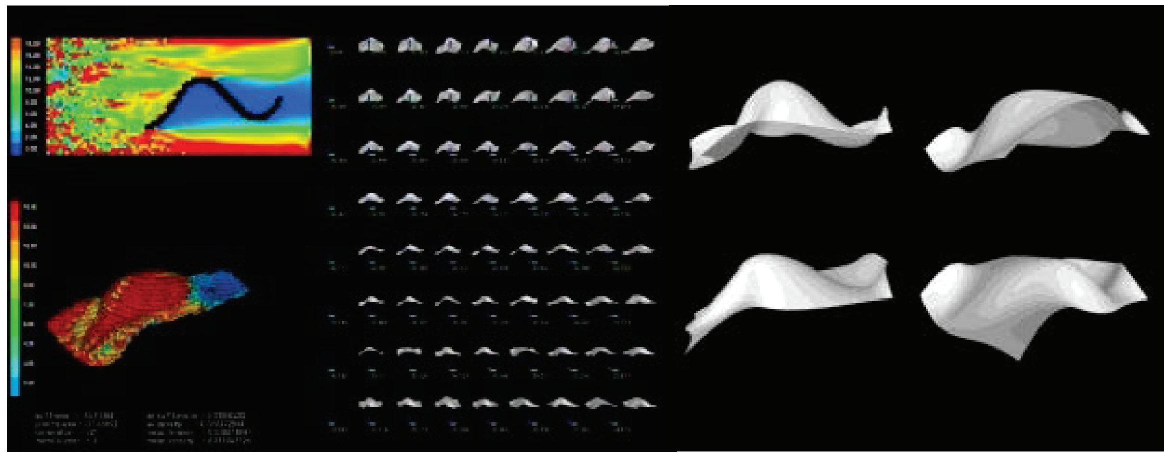


Figure 2-11: Optimisation interface and 4 fittest members (Chronis et al., 2011)

Waibel et al. (2017) developed and validated an FFD solver in Grasshopper called GH Wind. Once again, the aim of using FFD was to reduce computational time especially for the early design phase where the geometry may not yet be finalised. They implemented a few changes to Stam's algorithm to enable more accurate pressure values and parallelise the simulation to allow faster calculation. A major issue discovered was that at high Reynold's numbers the results deviate much more from the reference. This is most likely due to the fact that FFD at this point does not model turbulence effects which are more critical at high Reynold's numbers. To deal with that the researchers set the kinematic viscosity to a higher value of v = 0.1 in order to artificially lower Re but conceded that this approach needs further research. For pressure coefficients on a façade, they observed that for the windward façade the results were very close to the reference from ASHRAE. For the leeward and side faces, the distribution was different, however, the numerical range of values of C_p was within that of the ASHRAE values.

Although it has shortcomings FFD could provide a reasonable trade-off between accuracy and computation time (Waibel et al., 2017) especially for early-stage optimisation. Zuo and Chen (2009) found that their FFD simulations were at least 50 times faster than CFD simulations for the same grid size and time step although the results were not as accurate as RNG $k-\varepsilon$.

2.4.4 Available programs/plugins

In the decades since its inception, CFD has developed immensely and now there are many programs in use today by engineers and researchers. We will examine a few below based on information from the developers and the author's own experience.

Autodesk Flow Design

Flow Design is a virtual wind tunnel program developed by Autodesk. It is easy to use and offers many visualisation options such as flow lines, surface pressures, drag plot, and velocity planes. Many file types can be imported. Options such as wind speed, tunnel size, and resolution can be changed as well as the rotation of the model. It is intended as a quick visualiser and thus it is fast and easy to use but maybe too simplistic. There is no option to pick a turbulence model or roughness length. The results are only presented visually with colours but there is no option to export the numerical data nor to see the numbers at the points they are taken on the model. Resolution is only represented by a percentage and a readout of voxel size. Also, the project cannot be saved so for each use the mesh will have to be reimported and tunnel parameters reset. As of March 28, 2018, Autodesk has stopped distribution of Flow Design.

ANSYS Fluent

Fluent is part of ANSYS's workbench of engineering tools. It is widely known in the industry as reliable and accurate. There are many options for meshing, turbulence models, and many other settings. Custom scripts can also be written to make modifications to turbulence models. Many file types can be imported. Fluent is very powerful however, it is quite complicated, and a novice user could not just pick it up and instantly run an analysis without first becoming familiar with the program. It is ideal for detailed analysis but perhaps less so for early stage exploration.

OpenFOAM

OpenFOAM is a free, open source, CFD software that has been in development by OpenFOAM Ltd. since 2004. It is known to be robust and accurate and its open-source nature allows many researchers to use it and customise it for exactly what they need. Thus, it is one of the most independently validated CFD libraries in existence. It also offers a large array of turbulence models, solvers, meshing algorithms, and other tools. OpenFOAM is a Linux based C++ library that is primarily run through a console, but results can be visualised via the software Paraview. This can make it complicated to use. One must be very familiar with the OpenFOAM syntax in order to perform analyses.

Butterfly

Butterfly is a Grasshopper plugin and python library developed by Mostapha Sadeghipour Roudsari as part of the Ladybug Tools suite. It is a Python wrapper for the OpenFOAM C++ library that allows users to run OpenFOAM CFD simulations from within the Grasshopper environment for cases pertaining to building design such as outdoor airflow, indoor airflow, buoyancy, and HVAC. It greatly simplifies the use of OpenFOAM by using Grasshopper components and allowing the user to integrate other plugins and components of the Grasshopper environment. Installing and using Butterfly has been made much simpler in version 0.0.05 by using blueCFD-core which is a build of OpenFOAM which runs natively in Windows. Butterfly's readily available source code allows users the freedom to modify and improve the software on their own. One also has the benefit of a powerful and accurate solver running within a fully parametric environment. Being part of Grasshopper and the extensively used Ladybug tools it has a very active online community in addition to the active OpenFOAM community. Features such as roughness length, tunnel size, and mesh refinements can be numerically set. It has a large assortment of RANS/RAS models, however, LES models have not yet been implemented.

GH Wind

GH Wind is a Fast Fluid Dynamics solver plugin for Grasshopper developed by Christoph Waibel as part of his 2017 paper, Validation of Grasshopper-based Fast Fluid Dynamics for Air Flow around Buildings in Early Design Stage (Waibel et al., 2017). It includes components for forming the wind tunnel, meshing, solving, and visualising the pressure and velocity fields. Based on the results of the paper it is much quicker than OpenFOAM CFD solutions but suffers from some inaccuracy especially in the wake regions of the flow. Being in the Grasshopper environment it allows users, just like with Butterfly, to couple it with the large array of components and plugins available. The speed of the solution also makes it handy for optimisation problems. However, it is fairly new and as a result not independently validated to the extent of the other programs and plugins on this list. It also has no online community for support and does not appear to be in ongoing development. It is open source with source code provided in C#.

2.5 Fluid-Structure Interaction

Fluid-Structure Interaction (FSI) is the analysis of the forces, deformations, and dynamic motions imposed on a body in fluid flow (Bungartz and Schäfer, 2006). In this case, it would be the static and dynamic forces exerted on a building due to the wind. FSI usually takes place in two forms:

(1) Monolithic: where the equations for fluid flow and structural deformation are solved simultaneously in a single solver.

(2) Partitioned: where the fluid flow and structural deformation are solved separately with two separate solvers (Bungartz and Schäfer, 2006).

Monolithic solvers require specialised code using numerical methods to solve both problems simultaneously. An example of this is seen in Chronis et al. (2011) whose code developed in the Processing language integrated an FFD solver which returned surface pressures and ran a genetic optimisation algorithm. Thus, the partitioned approach is preferred for this thesis. This requires a separate Finite Element Analysis (FEA) software/plugin that the CFD solver could transfer the results in order to obtain information about the building response.

Karamba3D is a parametric FEA plugin for Grasshopper developed by Clemens Preisinger in cooperation with Bollinger und Grohmann ZT GmbH. It is widely used and offers many options for analysis including many types of loads, materials, cross sections, as well as solvers and results. It's parametric nature, given it is part of Grasshopper allows it to be coupled with many of the other plugins available and give real-time results (Preisinger, 2013). This makes it a prime candidate for coupling with a CFD solver. There are already many studies and examples of Karamba being used with optimisation algorithms to optimise structural components as well as overall geometry for structural performance.

CFD simulations return pressure values on the external massing. This will need to be translated to the building elements such as the floors, beams, and core in order to ascertain the deflection imposed on the building by the wind. In the next stage of the research different methods of this will be devised and evaluated with based on how well they address the problem, feasibility to implement computationally, and ability to be integrated into an optimisation loop.

2.6 Optimisation

Optimisation in a mathematical or computational sense involves the manipulating of various input factors in order to minimise or maximise a certain output result. It allows designers to use computational algorithms to determine the best design solutions based on a number of performance factors. This is the essence of performance-based design where a design is driven, not solely by aesthetics, but by the achievement of certain performance goals such as structural deflections, daylight, or energy use (Oxman, 2006). In the case of architectural design, optimisation usually focuses solely on the input variables and the resulting objective outputs with no regard for the mathematical definition of what comes in between. This is known as black-box optimisation (BBO) or derivative-free optimisation (Wortmann et al., 2017). It allows the designer to use optimisation with any number of algorithms or simulations such as CFD and use the resulting outputs as objectives. The optimisation usually follows a loop structure where some parameters, such as length, width, thickness, position, etc., are input. The design is then evaluated for its performance be it climatic, structural, or other criteria using an analysis or simulation procedure that outputs results. These results (structural deflection, energy use, etc.) are compared to the goal set by the designer. If it is not satisfactory, the parameters are then set to another value and the process continues until the goal is met. The outputs are called the objective functions and their closeness to the goal is its fitness.

In single objective optimisation, input variables are evaluated against a single output objective which is aimed to be minimised or maximised. Thus, it converges to a single solution. However, engineering problems usually require the careful balancing of a variety of often contradictory objectives (Evins, 2013). This reality has given rise to the use of Multi-Objective Optimisation (MOO). In MOO, also called Pareto optimisation, the aim is to obtain a range of solutions that span the trade-off between each of the objectives. The MOO loop runs until each objective cannot be improved any further without worsening others. This leads to a range of solutions called the Pareto front (Evins, 2013). From this, an architect/engineer must choose the best design based on the importance of each objective. Indeed it is often much more valuable in architectural problems to give a range of solutions rather than a single optimum as many aspects including architectural aesthetics have to be balanced by many parties involved in the design process (Turrin et al., 2011).

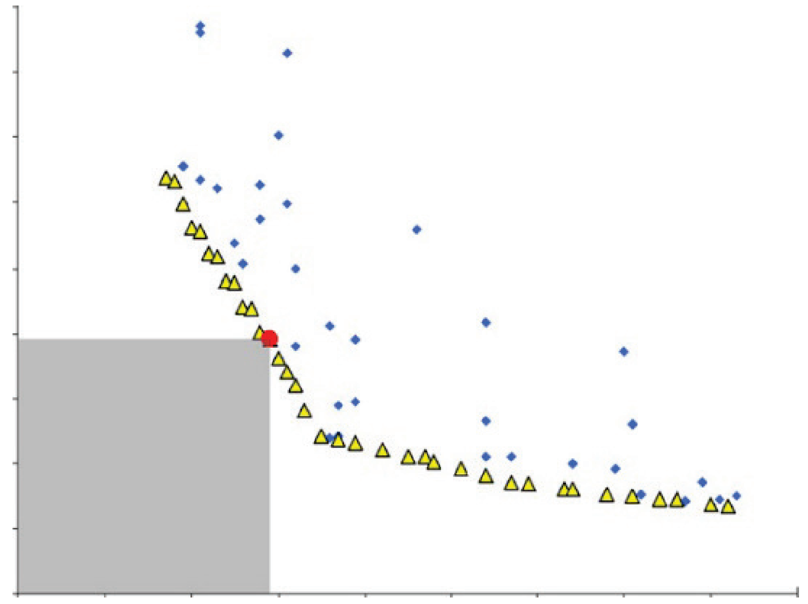


Figure 2-12: Example of optimisation results with Pareto front (yellow triangles) (Evins, 2013)

2.6.1 Optimisation algorithms

There are many algorithms that can be used to perform optimisation ranging from purely mathematical to even those based on natural processes. These can be defined as either deterministic or stochastic. In deterministic, a certain set of input variables will always return a given objective value for a specific case. Stochastic on the other hand are more random so that a given starting point will not always return the same exact value (Ilunga and Leitão, 2018). For BBO methods these are usually divided into three categories: Direct search methods, metaheuristics, and model-based methods (Wortmann and Nannicini, 2016).

Direct Search

Direct search algorithms are deterministic methods which perform sequential examinations of trial solutions using points in the solution space generated by a certain strategy (Rios and Sahinidis, 2013). Examples include Dividing Rectangles (DIRECT) method, Parallel Axis (PRAXIS) method, and Nelder-Mead Simplex (NMS). For example, as shown in Figure 2-13 the Nelder-Mead Simplex algorithm first forms a simplex, i.e. a shape of n+1 vertices in an n-dimensional space, of points in the solution space. The next point is found by first taking the average of the two best points and performing a transformation of reflection, expansion, contraction, or shrink depending on the objective function value of the new point (Gregson, 2018).

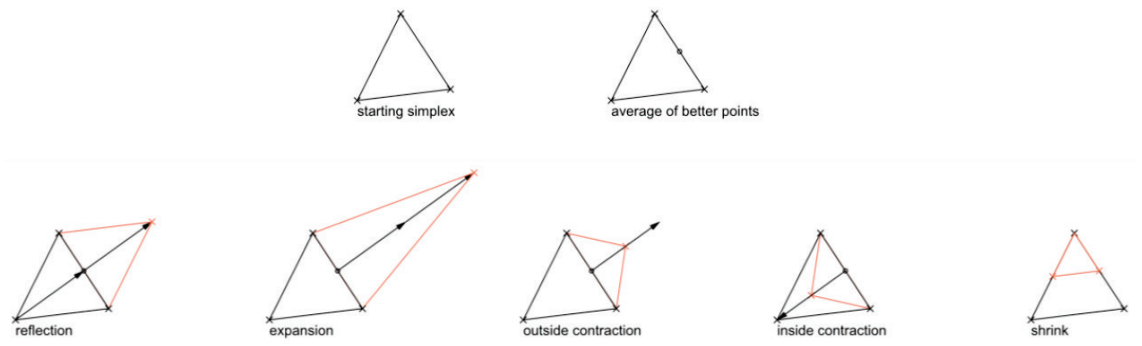


Figure 2-13: Nelder-Mead process (Gregson, 2018)

Direct search methods are highly used in mathematical test problems due to their fast performance and inherent stability however, they are not much used in architectural optimisation problems (Wortmann and Nannicini, 2016). While very efficient they are not very robust (Evins, 2013). This means that they tend to get trapped in local optima converging to a minimum (or maximum) of one section of design space while there may be other sections in the global solution space with better objective values. This can be alleviated by doing multiple optimisation runs at random start positions but of course, doing multiple optimisation runs greatly increases the time taken.

Direct search methods currently present as plugins within Grasshopper include the Nelder-Mead plugin by Eckersley O'Callaghan Engineers, as well as Goat developed by Rechenraum e.U which uses DIRECT, Subplex, and another form of Nelder-Mead Simplex.

Metaheuristics

Metaheuristic algorithms are stochastic methods inspired by natural processes (Ilunga and Leitão, 2018). These include algorithms such as Genetic Algorithms (GA) which are based on Darwinian survival of the fittest principles, Simulated Annealing (SA) which simulates the behaviour of metal molecules during the annealing process, and Particle Swarm Optimisation (PSO) which follow swarm intelligence principles (Wortmann et al., 2017). Metaheuristics, particularly GAs are the most popular choice in architectural applications since they are readily available (they come preloaded within Grasshopper in the plugins Galapagos and Octopus), can be applied to almost any problem, and are easy to understand and use (Wortmann et al., 2015). However, many mathematicians regard metaheuristics as "methods of last resort" (Conn et al., 2009). They tend to perform poorly in benchmarks compared to other algorithms as seen in studies by

Waibel et al. (2019), Wortmann (2018), Ilunga and Leitão (2018), Wortmann et al. (2017), and Rios and Sahinidis (2013). Additionally, metaheuristics typically require a much larger number of function calls to arrive at an optimum which is particularly problematic for cases such as CFD where a single function evaluation could take hours (Wortmann and Nannicini, 2016).

Model-Based

Model-based optimisation is a stochastic method which operates differently than direct search or metaheuristics. It involves replacing a computationally expensive objective function with an inexpensive surrogate model with the same input and output space as the original function. The search for the optimum is then carried on this surrogate rather than the original intensive function (Bernardini et al., 2015). It first creates a set of points called a sampling plan by doing a few iterations of carefully chosen points in the solution space to obtain some results (Bernardini et al., 2015). Then, it generates a surrogate model or response surface (Figure 2-14) of the unknown fitness landscape by interpolating through the points in the sampling plan. From this, it can estimate the performance of design candidates with fewer or no further function calls (Wortmann and Nannicini, 2016). The algorithms can generate local models such as the Trust Region method or global models which create a surrogate of the entire solution space using statistical methods, such as Polynomial Regression and the Kriging method, or machine-learning, using Neural Networks, Support Vector Machines, or Radial Basis Functions (RBF) (Wortmann, 2017).

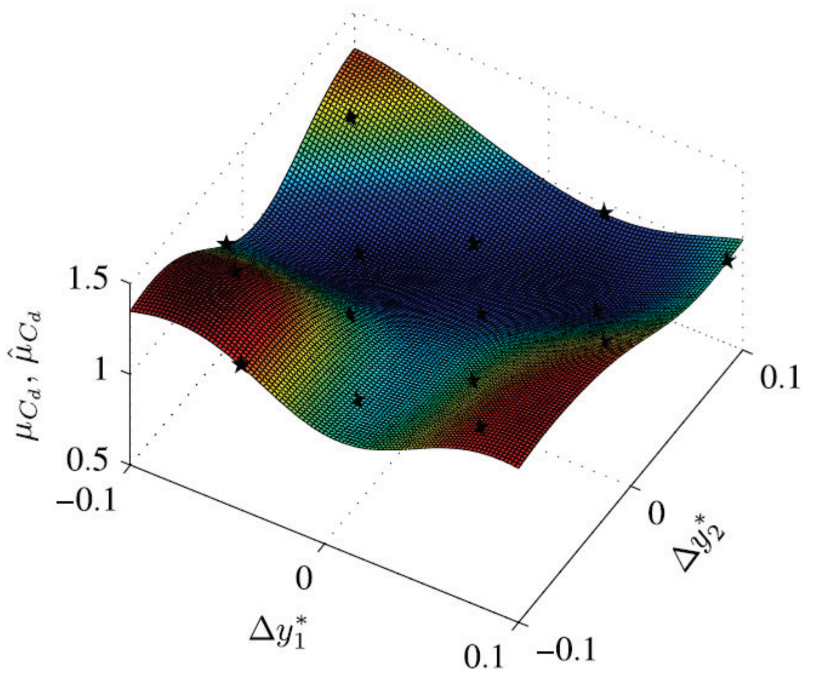


Figure 2-14: Surrogate model (response surface) of a Kriging based optimisation with input variables on the x and y-axes and the objective variable is on the z-axis (Bernardini et al., 2015).

Surrogate model-based algorithms will either construct a response surface from a sampling plan and search for an optimum solely using the model either with or without a separate optimisation algorithm, or it will generate a response surface and iteratively update the model by sampling the carefully chosen points while searching for an optimum. For example, Bernardini et al. (2015) performed 2D shape optimisation on a building cross-section analysed by CFD. They constructed a Kriging based surrogate model from which an evolutionary algorithm was employed to find an optimum. The Grasshoper3D plugin, Opossum developed by Wortmann (2017) uses a Radial Basis Function (RBF) method to construct the initial response surface and iteratively improve it based on carefully chosen evaluation points. The RBF algorithm is found to perform better than other surrogate model algorithms in problems employing time-consuming simulations (Wortmann et al., 2015). Yang et al. (2016) in a study of the effects of sampling strategy and problem scale found RBF to perform the best for a low number of variables.

Many benchmark studies of optimisation algorithms, such as those mentioned in the previous section, conclude that model-based algorithms provide fast convergence, stability, and robustness particularly in optimisation problems requiring expensive simulations. Bernardini et al. (2015) in the CFD optimisation study mentioned above state that:

> In order to find the Pareto fronts discussed here, a total of 90 CFD simulations were carried out while a total of about 12,000 evaluations of the Kriging models were made at each design update. Therefore, by following the proposed approach, only 0.75% of the CFD runs necessary to directly search for the Pareto optimal solution are necessary, which once again illustrates the strong potential of the proposed [Aerodynamic Shape Optimisation] approach.

Since CFD evaluations can take a very long time it is beneficial to reduce the number of function calls as much as possible.

2.6.2 Optimisation Problem Formulation

Optimisation has seen an increase in popularity in the past few years and many engineers and researchers have put it to good use in solving problems of the building industry. However, building design is a complex procedure involving many, often competing, objectives. On the other hand, optimisation is a very definitive process where there needs to be set input variables with a defined range of values and one or many meaningful objectives. Thus, designers have to distil each case into a well-defined optimisation problem that can be used with optimisation algorithms (Wortmann et al., 2015). For this reason, Yang et al. (2018)

assert that Optimisation Problem Formulation (OPF) is even more important than obtaining Optimisation Problem Solutions (OPS). OPF involves evaluating the design to determine first which parameters and objectives are the most valuable in the present case. It involves two main parts:

(1) Formulation of the objective space: selecting objective and constraint variables (outputs) and constraint values. This determines the performance goals and constraints to be achieved.

(2) Formulation of the design space: selecting design variables (inputs) and their domains. This determines the possible design alternatives that can be searched.

It is essential to define the problem carefully first to avoid creating meaningless problems which upon optimisation give useless solutions (Yang et al., 2018). All problems, especially related to computational processes, are ill-structured problems at first according to Simon (1973). It is only after testing and formulating the problem to adapt to the problem solver being used do they trend toward a well-structured problem. Most studies, however, mainly focus on OPS rather than OPF giving results for a singular research setup rather than the process of determining the best setup for the problem. Indeed, in this research, it was difficult to find examples of this type of work.

Careful consideration should be given to, for instance, what parameters have the most impact on performance, can they be easily modified in the design, and how can their domain be restricted so that the obtained solutions are feasible and attractive. In addition, one must take into account the abilities of the optimisation algorithm itself. For example, Waibel et al. (2019) performed a benchmark study of multiple optimisation algorithms applied to building energy problems. The algorithms were tested using the same building models but different numbers of input variables, different ranges, and also continuous versus discrete variables, in order to compare the algorithms' performance in each problem space.

OPF is crucial in the early conceptual design phase where designers are not able to perceive every aspect of the design project and how the chosen workflow can be applied to it. At this stage, goals are usually vague, and a lot of the choices are based mostly on experience, educated guesses and intuition rather than hard information and results. At this point, the process can be referred to as Computational Design Exploration (CDE) rather than Computational Design Optimisation (CDO). CDE, rather than solving a specific problem, involves fixing a problem within the problem space by searching for a solution in the solution space in an iterative manner where the characteristics of the found solution can now reform the original problem space generating a new solution space and so on (Maher et al., 1996). To formulate an optimisation problem that produces a

meaningful optimum, exploration must be done on the problem and solution spaces.

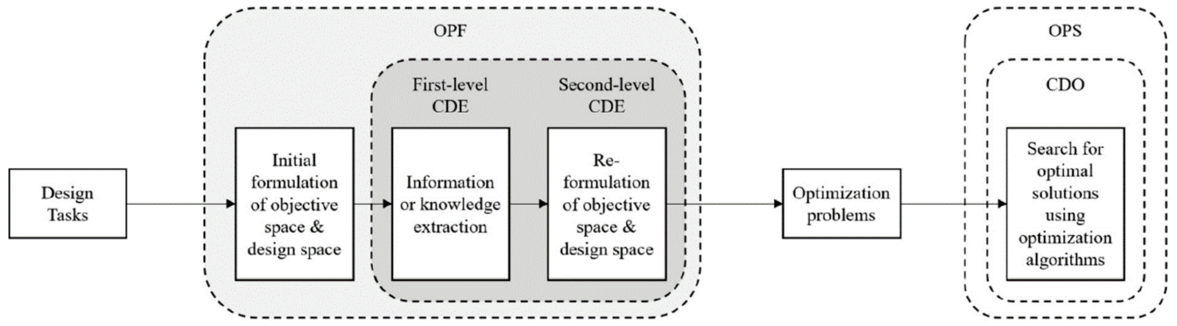

OPF
First-level CDE
Second-level CDE
Design Tasks
Initial formulation of objective space & design space
Information or knowledge extraction
Re-formulation of objective space & design space
Optimization problems
OPS
CDO
Search for optimal solutions using optimization algorithms

Figure 2-15: Relationship between OPF, CDE, OPS, and CDO (Yang et al., 2018)

OPF is an iterative process that requires performing multiple optimisation runs to obtain an ideal OPS. Thus, the choice of optimisation algorithm also plays an important role particularly in this case using expensive CFD simulations. The greater the number of input variables the greater the control over the outcome and the possibility for better-performing objective. However, this increases the computational cost of the optimisation. It may arrive at a point where the increase in runtime is not worth the marginal gains in fitness. Thus, the number of inputs to be manipulated should be kept minimal to be most efficient. Just as important is the domain within which these values can be changed. Too wide a domain requires a lot more generations to ensure adequate samples are taken while too small a domain may leave out truly optimal values.

While the explored examples focused on a more detailed optimisation problem, the method to be in developed in this thesis is on a much coarser level of detail. What is important is the focus on experimenting with the different arrangements of input, objective, and algorithm settings to determine the optimal optimisation. This is especially important in a field such as CFD based optimisation in buildings where not a lot of research has been done.

2.6.3 CFD based optimisation in buildings

While optimisation has been increasingly applied to both environmental and structural simulations in buildings, optimisation using CFD is still comparatively rare. For instance, Ekici et al. (2019) performed a review of the usage of computational optimisation in built environment problems. Out of one hundred papers found, only one used CFD which was used for HVAC flow simulation. CFD based optimisation (CFD-O), however, has been applied extensively in aeronautical engineering for aerodynamic shape optimisation (ASO) of aircraft wings, engines, etc. (Thévenin and Janiga, 2008). Indeed, in this literature review, it was difficult to find research on CFD-O for buildings particularly to optimise structural objectives.

However, Bernardini et al. (2015) see the importance of CFD-O over the traditional trial and error approach using wind tunnels sating that:

> The possibility of taking advantage of computational fluid dynamics (CFD) simulations for the assessment of the aerodynamic performance while using optimization algorithms to find the best aerodynamic shape is therefore very attractive as it would allow not only to rigorously and thoroughly investigate the search domain but to do so automatically, also in principle eliminating the necessity of costly wind tunnel experiments.

CFD by its nature imparts some issues to its use in optimisation. For one, CFD is in itself not an exact evaluation but rather an approximation of a physical phenomenon that is highly dependent parameters such as mesh size and discretisation. This results in optimisations always having a certain level of uncertainty. Bernardini et al. (2015) state that this is to be expected as users may tend to attempt to shorten CFD's long calculation time by using coarser grids or fewer iterations. However, the uncertainty should be small enough to still allow for a meaningful optimisation. CFD simulations can vary widely based on the problem being analysed. It is a balance between time and accuracy and time plays a big role especially in commercial applications. Thus, it is reasonable to conclude that CFD-O is practical only when a single CFD evaluation takes at most a few hours (Bernardini et al., 2015).

2.7 Conclusions

This literature study was done to establish the current state of the art in terms of wind load calculations and the potential of using computational methods for performing those calculations and optimisation of buildings for such. It can be seen from the study of wind in the environment and its actions on structures that wind and its flows are a complex phenomenon. This mostly due to its turbulent nature particularly in its interaction with bodies such as buildings. Pressure drag in the direction of the wind should be considered but also accelerations due to vortex shedding and torsion. The Navier-Stokes equations describe fluid flow but these do not have a closed form solution. This led to the simplified quasi-static equations for wind loading. Eurocode EN1991-1-4:2005 is based on these. About 20 equations are required simply to obtain the wind force at a single height and wind direction. This is grossly inefficient for early-stage design exploration when multiple iterations are usually done. In addition, the code only offers guidance for a limited number of simple building geometries. No procedure is given for

complex shapes. However, the EN does allow for the use of validated numerical methods for the calculation of wind loading. Thus, CFD is a possibility if it can be shown to give good results.

CFD takes the Navier-Stokes equations and discretises them to a mesh or grid so that they can be solved using numerical methods. Many established and validated programs such as ANSYS Fluent and OpenFOAM exist which employ powerful solvers that can simulate the flow of air around a building and the resulting surface pressures. OpenFOAM is integrated into the Grasshopper environment with the plugin Butterfly thus allowing the use of a powerful and widely validated solver within a parametric environment. Plus, its open source nature allows for modifying the tools to fit the needs of this thesis. The main factors in the consideration of CFD in this project are time and accuracy. Mesh size and choice of turbulence model have the largest impact on these. As seen in section 2.4.2 there are many turbulence models to choose from each with their own pros and cons. Fast Fluid Dynamics has been shown to be much faster than CFD but suffers from some inaccuracy particularly due to its lack of turbulence model. Further investigation is needed into the choice of fluid solver as it is a key part of the proposed tool. Table 2-2 below shows the chosen shortlist of solvers and their pros and cons. A variety of RANS models were chosen with the Butterfly component in addition to the FFD component GH Wind. In the next steps of the research, these will be evaluated on the two criteria: accuracy and time to determine the best choice. A comparison will be made to validated results from research to determine their accuracy.

	Pros	**Cons**
Butterfly ($k-\varepsilon$)	• Very fast • Good freestream flow accuracy • Coarse grid	• Inaccurate near-body flow • Inaccurate pressures on sides and in wake
Butterfly (Realisable $k-\varepsilon$)	• Better accuracy in separated flows and wake region • Coarse grid	• Slower than standard models
Butterfly (RNG $k-\varepsilon$)	• Better near body flow and in wake than standard $k-\varepsilon$	• Slow calculation time
GH Wind	• Faster than traditional CFD models • Simpler meshing (no grading)	• Less accurate than CFD mainly in turbulent regions

Table 2-2: Comparison of CFD methods

Karamba3D will be used to calculate the structural performance. Karamba is an established, well-known and validated FEA plugin in the parametric environment. It gives a large array of options for inputs and calculations Karamba will be integrated with the chosen CFD solution with a translation procedure to take the pressure loads to a structural model and obtain results like deflections and moments. Vortex shedding is also important for this application however, it is difficult to obtain the shedding frequency from steady-state RANS solutions.

While genetic algorithms are the most popular method of optimisation, particularly in Grasshopper, their high number of required function calls make them unsuitable for a computationally expensive process such as CFD. Direct search methods, while efficient, suffer from low robustness which is not ideal for building cases where an array of options is desired rather than a single optimum geometry. Doing multiple optimisation runs of a direct search method to mitigate robustness would most likely be just as, or even more, inefficient than a genetic algorithm. Model-based algorithms appear to be the most promising as the benchmarks studies have shown their reliability, robustness, and the ability to arrive at convergence with fewer function calls than other algorithms. Therefore, in this case, the Opossum plugin (Wortmann, 2017) will be used with the RBFopt algorithm.

To ensure a well-balanced optimisation procedure time will be spent evaluating different objectives and selecting an ideal array of objective and input variables to formulate a meaningful optimisation problem. Since Opossum is a single-objective optimisation plugin, objectives will be considered and evaluated to determine which is the most meaningful for this research. The number of input variables will be kept low (max 3) to further help reduce computation time. These inputs will be solely for manipulating the geometry. All other settings for the CFD and FEA will be constant.

Figure 2-16 outlines the computational procedure with the environment it is contained in, i.e. Grasshopper, the key parts, and the components to be used or, in the case of CFD and translation algorithm, still to be determined or developed.

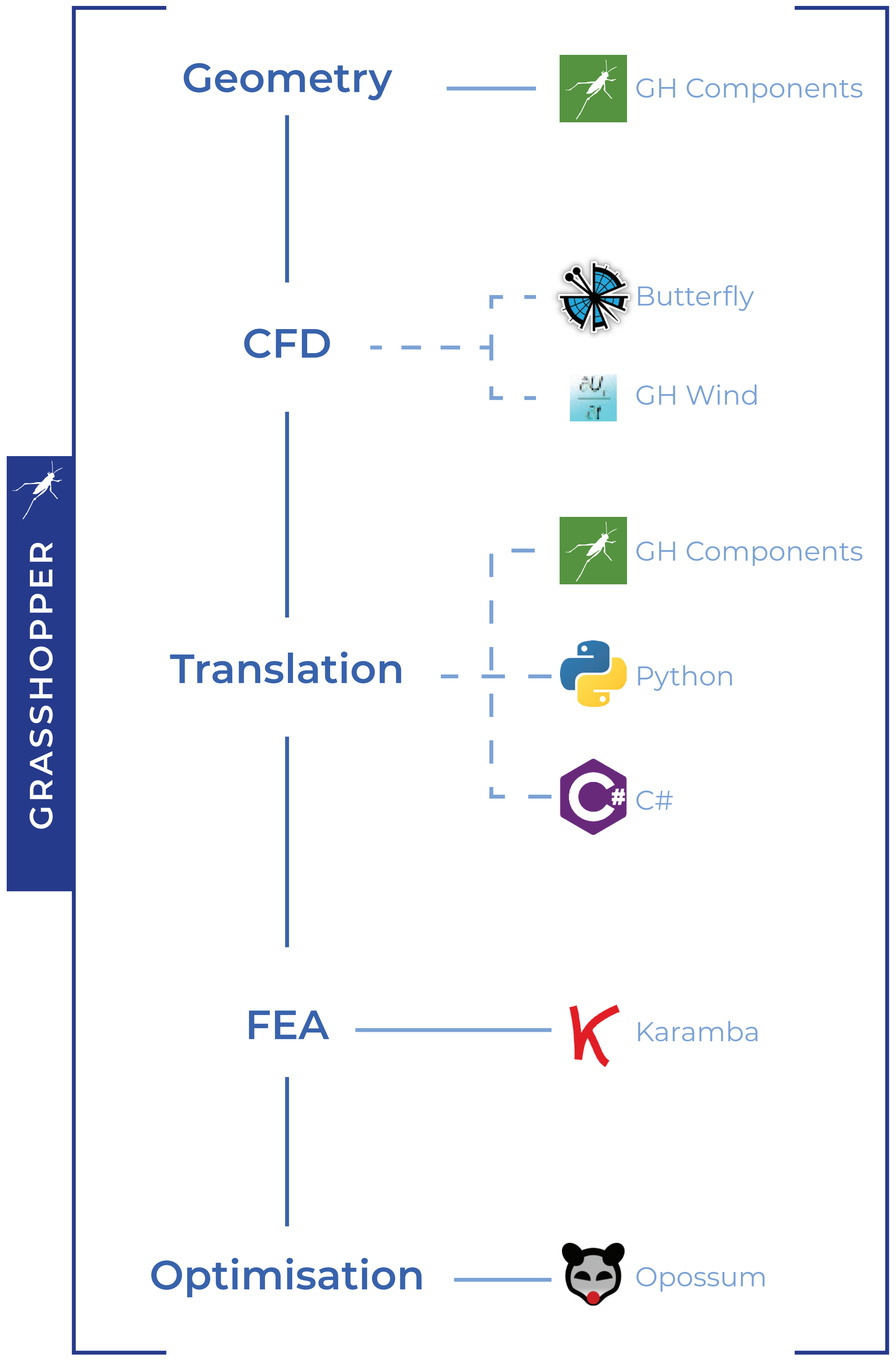


Figure 2-16: Computational procedure

3 DEVELOP

This stage of the thesis is focused on the development and testing of the computational method through development of a tool in Grasshopper. It builds on the knowledge gained in the research portion and continues in a joint research and design method. Firstly, case study buildings are selected on which the method will be tested. The work was divided into the CFD portion, development of the FSI procedure, and the addition of the optimisation algorithm. In the CFD portion, first, the two selected algorithms are set up in Grasshopper. This involved understanding the methods and modifying them to conform to standard CFD practices while making the procedure more adaptable to the geometries expected. The next step is the validation which was needed to establish the accuracy of the CFD method compared to physical wind tunnel tests and the time taken for the simulation in order to achieve those results. This was followed by sensitivity analyses to determine which parameters could be modified to reduce time. After these tests, the results of the two algorithms are compared and the best one was chosen based on accuracy, precision, and time. The FSI chapter details the chosen FEA method and the development of the algorithm for coupling it to the CFD procedure. Lastly, the optimisation chapter details the addition of the optimisation algorithm and the series of tests done in order to arrive at a meaningful optimisation problem.

3.1 Case Study Buildings

To help evaluate and develop the tool case study buildings were chosen to be used as input geometry in the procedure. These were high-rise buildings chosen based on their non-standard geometries and the opportunity they presented to challenge the effectiveness of the tool. These are the Absolute Towers by MAD Architects due to its twisting geometry, Jiangxi Nanchang Greenland Central Plaza by SOM for its varying cross-sectional shape and supertall height, and the Ardmore Residences by UNStudio for its unconventional floor plan shape (Figure 3-1). These buildings were modelled as simple parametric masses. Each has two or three parameters controlling an aspect of its geometry which was used as input variables for the optimisation algorithm.

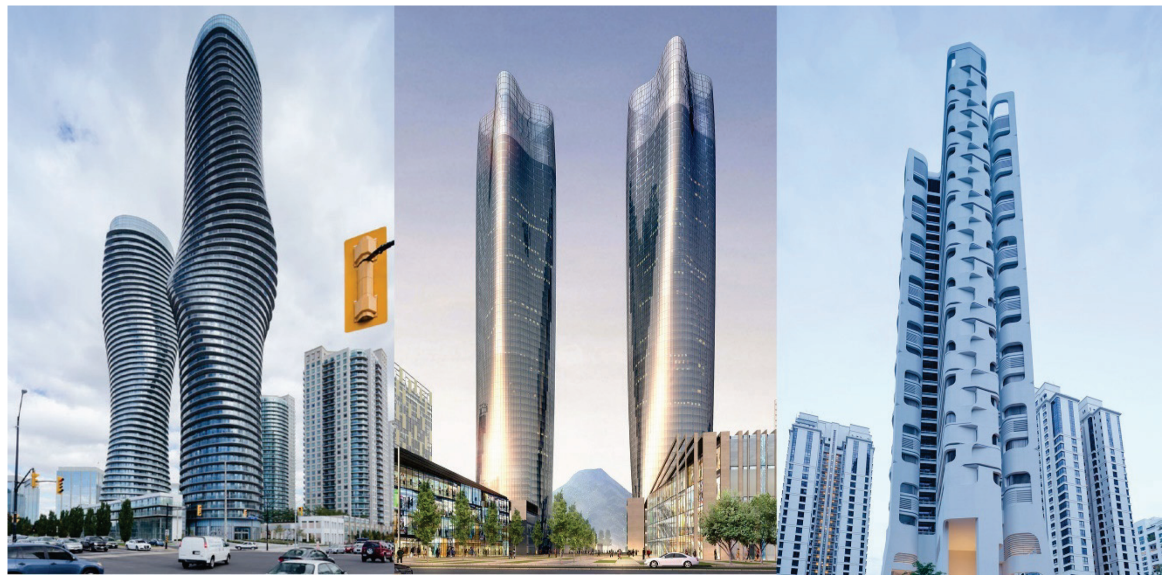


Figure 3-1: Absolute Towers by MAD Architects © Iwan Baan (Left) and Jiangxi Nanchang Greenland Central Plaza by SOM © SOM (Middle), and Ardmore Residence by UNStudio © Iwan Baan (right)

3.2 CFD

3.2.1 CFD script setup

A typical CFD procedure involves inputting the geometry to be analysed then the domain or virtual wind tunnel is created around it. The entire domain is meshed to create a 3D mesh of the space between the tunnel and geometry to be analysed (Figure 3-2). The solver then runs iteratively calculating the flow in each cell until it reaches convergence. CFD has many parameters which affect how well the solution runs. The aim was to make the chosen CFD procedures easy to use and adaptable to any geometry that one would input and obtain the pressure on the facade. Settings were made constant or parametric, based on the dimensions of the input geometry, according to researched standard practices for CFD. The main settings affecting the outcome for this case were tunnel size, mesh cell size, and turbulence model. Other additional settings based on the individual CFD solver were looked at and set to the best option determined for this study. The CFD plugin Butterfly and the FFD plugin GH Wind for Grasshopper were used.

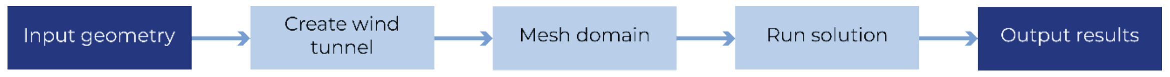


Figure 3-2: CFD workflow

Butterfly

The validation study (Section 3.2.2) was done using Butterfly version 0.0.04 while all subsequent work was done in version 0.0.05. Butterfly has many options that can be adjusted owing to OpenFOAM's complexity. To fit the purposes of this study some changes were made including removing all mesh refinement so that only implicit meshing takes place, i.e. the chosen cell size is used with no further subdivision. Also added were new custom components written in Python or C#. These were mainly to make the simulation faster and make the script as parametric as possible so that new geometries could be input with minimal changes having to be made to settings. See Appendix 6 for all C# and Python scripts.

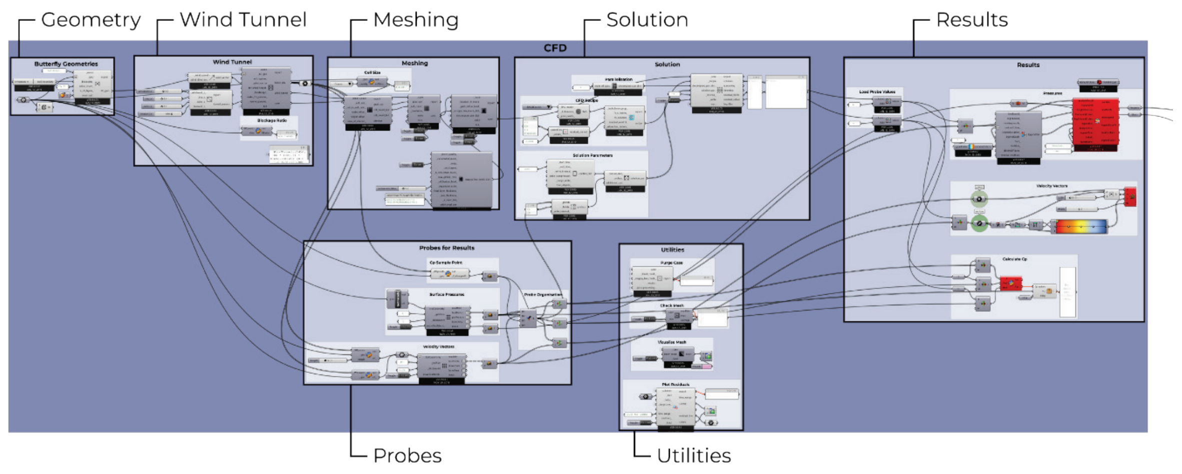


Figure 3-3: Butterfly script

Figure 3-3 shows the layout of the Butterfly script. Firstly, the geometry from Grasshopper is input to create Butterfly geometries. This is then connected to the wind tunnel component which defines the domain of the simulation as a virtual wind tunnel. The wind tunnel size is defined in multiples of the building height for the windward, leeward, side, and top extensions. A python script was written in order to calculate the blockage ratio, the ratio of the cross-sectional area of an obstruction in a wind tunnel (the building) to the cross-sectional area of the tunnel perpendicular to the wind flow. This should be kept under 3%, though some professionals suggest up to 10%, in order to prevent the artificial acceleration of the flow (Franke et al., 2007).

Next, the data moves to the meshing components where first a cell size is set. A Python script was written to parametrise the cell size. This takes the input of the building geometry and allows the setting of different mesh resolutions starting with the coarsest which is equal to the length of the shortest side of the building divided by 10 (Franke et al., 2007). Further refinements are had by dividing by 10 multiplied by root 2 for each level. eg. Medium = building length/(10 x √2), Fine = building length/(10 x √2 x √2), etc. as was recommended in personal correspondence with Adelya Doudart de la Grée, engineer and CFD expert at Cauberg Huygen, on February 18, 2019. Refinement levels are Coarse, Medium, fine, SuperFine, and XXFine. Butterfly uses two mesh types. First, a block mesh is created which fills the domain with regular hexahedral cells which are graded automatically i.e. desired cell size near the building which gradually gets larger further away. Afterwards, the SnappyHexMesh (SHM) component adjusts the block mesh to the geometry by removing cells from within the geometry and applying any refinement if chosen. As the name suggests it can also adjust the hexahedral cells by attempting to snap cells to the geometry of the building. For the tests with complex geometry, snapping was turn on.

In the solution portion, different parameters of the solution are set such as the turbulence model, max number of iterations, and residual values for convergence. The residuals are the scaled errors between successive iterations for different values such as pressure, velocity, k, and ε. These residual values tend toward zero with each successive iteration. A residual value of 0.0001 was set for convergence based on recommendations by Franke et al. (2007).

In order to obtain results of pressure and velocity, probe locations must be given as points. This can be obtained from the Generate Test Points component in the Probes section of the script by inputting a surface or by directly inputting points to the Probes component. After the solution, the values are obtained from the probes and used to colour a mesh based on pressure or show velocity vectors. Finally, the results section outputs the results of the simulation as a coloured mesh showing pressure and coloured arrows showing the velocity vectors of the wind flow.

GH Wind

GH Wind is set up differently to the more traditional CFD of Butterfly. Like with Butterfly some custom components had to be created in order to parametrise the workflow but also to obtain results that GH Wind at this stage could not give.

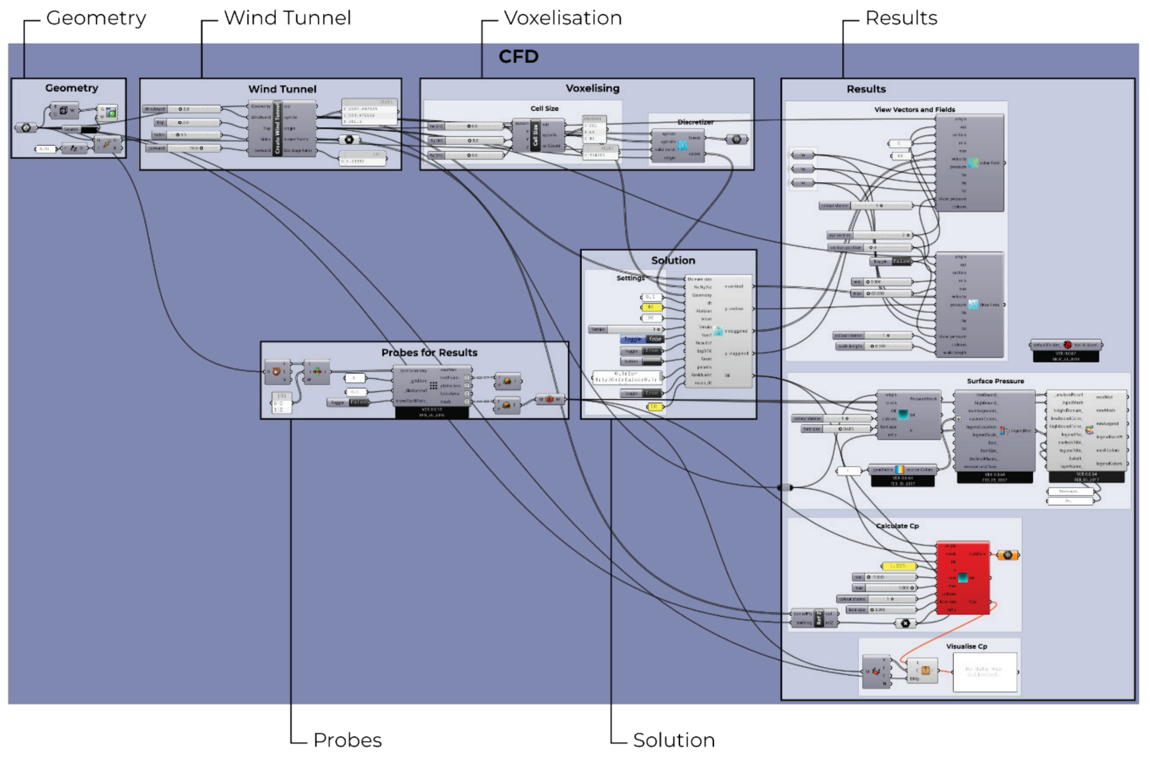


Figure 3-4: GH Wind Script

In GH Wind one can only manually set the tunnel size and the position of the object in it. Therefore, a custom C# code was written in order to define the tunnel in a similar way to Butterfly. GH Wind does not use the meshes of OpenFOAM/Butterfly but rather voxelises the domain and geometry. There is no grading, i.e. all voxels are the same size, and the grid does not snap to the geometry but rather is approximated by the voxels (see Figure 3-18). The Generate Test Points component is taken from Butterfly to be used here to create probe points where results obtained. This was then used with Ladybug components from Butterfly to colour the building mesh according to pressure. Solution settings such as fluid viscosity, time step, and max iterations are set in the solver component. While GH Wind has components for visualising pressure and velocity vectors on a plane, as well as pressure coefficient (C_p) on the building surface, it does not have the capability to output pressures on the surface in Pascals as in Butterfly. Therefore, using the provided source code a new Grasshopper component (Get Pressure) was written to output the pressure values.

3.2.2 CFD validation

A validation study was performed to ascertain the ability of the chosen CFD/FFD methods in this study to give results within an acceptable degree of uncertainty to physical tests. This was done by comparing results from the computational method against results from wind tunnel tests using the same building model. Also, noted as part of this validation is the time taken for each simulation as this is of interest for the project. For the comparison, the Commonwealth Advisory Aeronautical Research Council (CAARC) Standard Tall Building Model was used. This is a building model created in 1969 for the purpose of comparison of wind tunnel tests (Melbourne, 1980). Several studies were done by different institutions to gain results for wind tunnel tests. Today, its use has been expanded to CFD analysis and verification. One such research paper by Meng et al. (2018) is the basis for this study. The physical setup of this building was replicated in the two procedures outlined above for Butterfly and GH Wind and simulations run. The results for each method – Butterfly, GH Wind, and wind tunnel test – are compared.

Aim

The aims of this validation study were to study two aspects of CFD analysis: accuracy and time.

Accuracy was to verify that the tools selected for this thesis were reliable and to deduce to what extent they can replicate established setup procedures and results. This is done by comparing obtained values for the pressure coefficient at specific points. In Butterfly, this will also be done for various turbulence models as the model used can affect accuracy. Time was also important to this study since this thesis aims to develop a tool for designers that can be used in the early design stages where building geometry changes constantly. The time taken for each simulation to complete, Butterfly with each turbulence model, and GH Wind, will be recorded and compared. The balance of accuracy and time will determine what setup is selected at the end of the study. Since, based on the previous literature review, it is known that there is always some discrepancy in results between CFD and even physical wind tunnel tests, absolute accuracy was not what was looked for. Rather, the study aimed to determine the extent of the error between the results of different setups and how much time does the analysis need to reach that level of accuracy. This study also gave insight and understanding of the principles of CFD as well as the proper procedure for setting up the experiment, running it, and recording results.

Methodology

This validation study was done using data from the paper by Meng et al. (2018). The paper includes a table of pressure coefficients from physical scaled wind tunnel tests carried out on the CAARC model by various institutions. In that paper, the authors made tests of this model using the CFD package ANSYS Fluent and measured its sensitivity to different changing settings such as grid type, grid density, turbulence model, incoming wind speed, and wind direction.

Their CFD setup was replicated in Butterfly and GH Wind as outlined below. Pressure coefficients, C_p, from wind tunnel tests by Tonji University, TJ(D), given in table 2 of the paper, were compared to the CFD results. C_p is a dimensionless coefficient relating a reference pressure to the pressure at a body surface. It is calculated by:

$$C_{pi} = \frac{P_i - P_0}{0.5\rho U_0^2}$$

Equation 3-1: Pressure coefficient (Cp)

Where C_{pi} = mean wind pressure coefficient at a point i, P_i = wind pressure at point i, P_0 = static wind pressure at reference height, ρ = air density (1.225 kg/m^3), and U_0 = wind speed at reference height. The reference height is the height of the building: 182.88 m.

In addition to $C_{p,}$ the time for the simulation to run was recorded. All analyses performed were done on a desktop PC running Windows 10 with an Intel® Core™ i5-3470 CPU @ 3.20GHz, and 16GB of RAM.

Building Setup

The CAARC standard tall building model is a rectangular building of dimensions 30.48m x 45.72m x 182.88m (L x W x H). The wind tunnel tests done on the model used an array of 20 pressure taps around the building at 2/3 height to measure the pressure at the surface. The C_p will be measured at these points in the CFD/FFD setup as shown in Figure 3-5 and Figure 3-6.

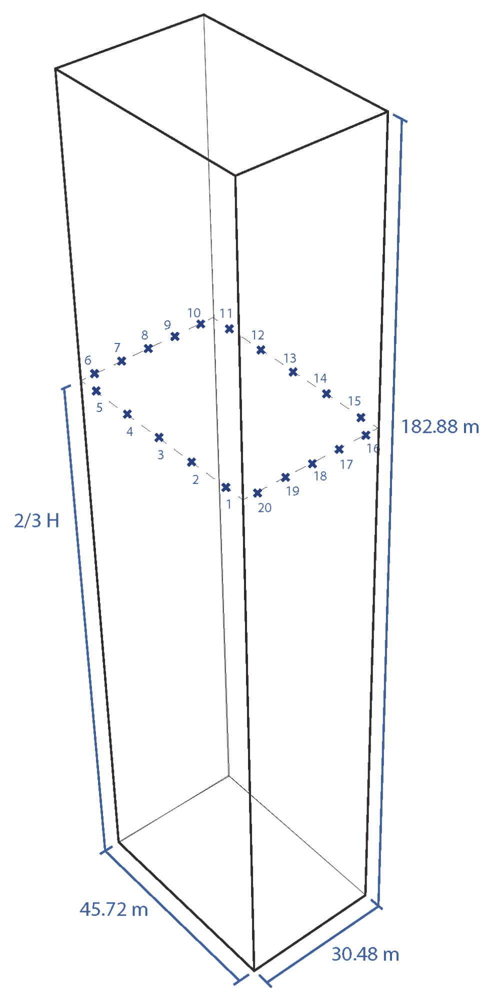


Figure 3-5: CAARC model dimensions

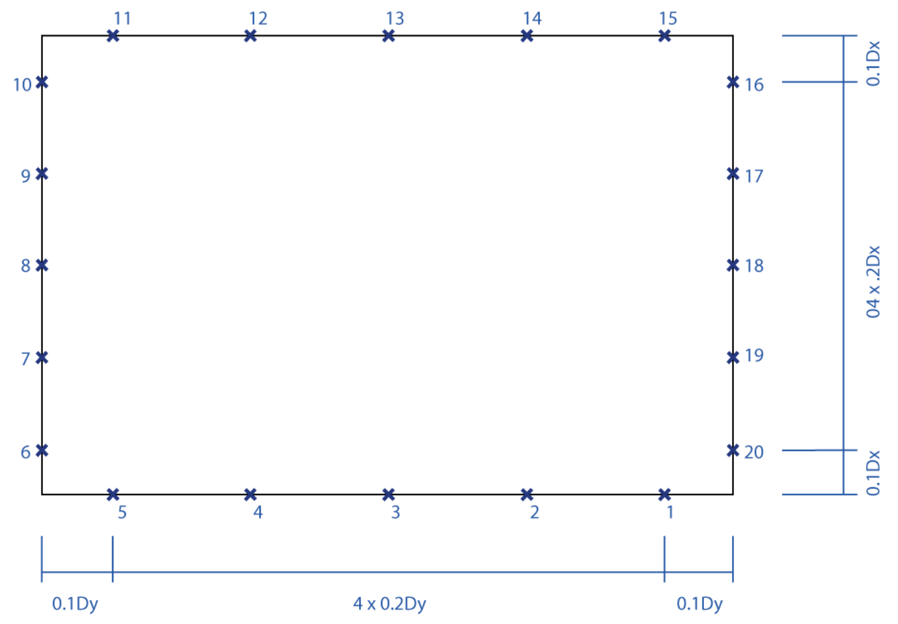


Figure 3-6: Pressure tap locations

Tunnel Setup

The wind tunnel domain from Meng et al. (2018) was replicated in both plugins as shown in Figure 3-7 below. The domain measures 900m x 600m x 400m (L x W x H). The building is centrally located in the Y direction and 300m from the front inlet boundary. This corresponds to a blockage ratio of 3.48% which is less than the chosen threshold of 5% (Meng et al., 2018). P_0 and U_0 are obtained at the reference point located at the inlet boundary, central in the y-direction, at building height (Figure 3-7).

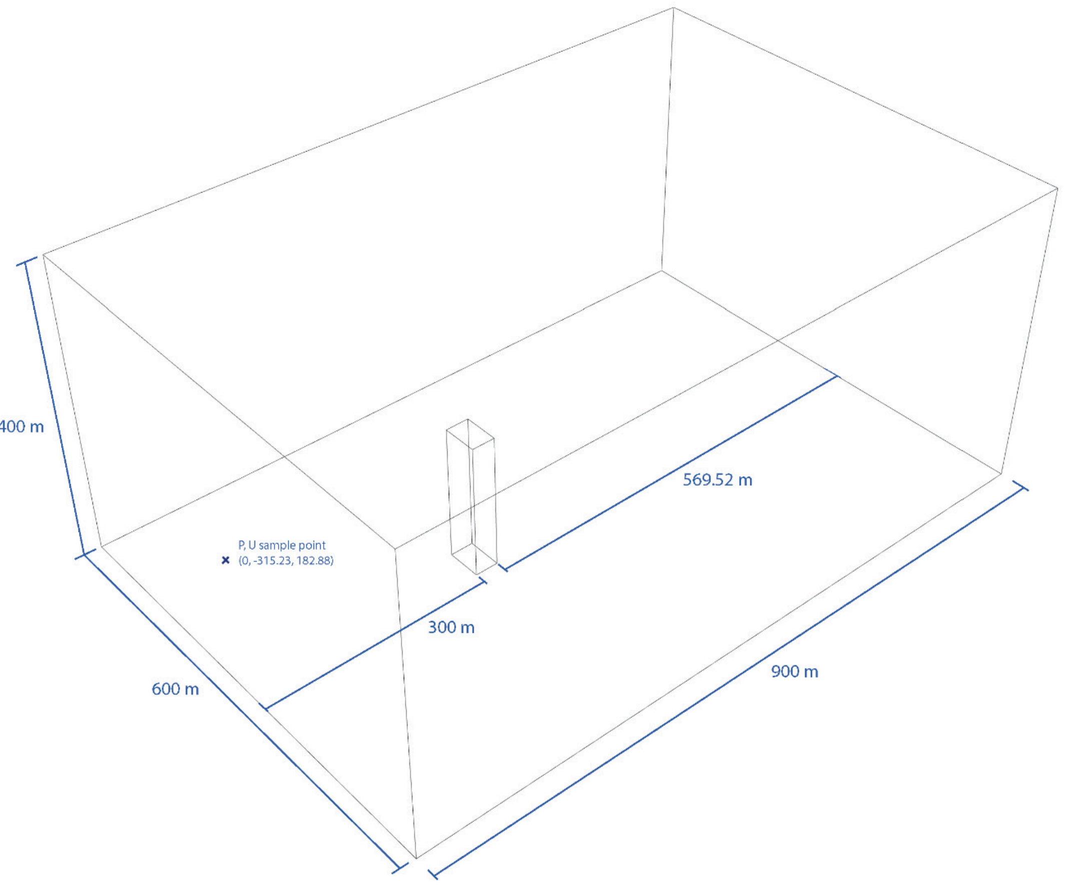


Figure 3-7: Wind tunnel domain

Meshing

The meshing of Meng et al. was attempted to be replicated. The smallest grid length was 0.0054H = 0.987552m at the building with grading in the X and Y direction (Figure 3-8 and Figure 3-9). This gave 615 120 cells which are less than the 850 000 cells from the paper possibly due to the slight difference in meshing methodology between ANSYS and OpenFOAM/Butterfly. GH Wind uses a structured regular hexahedral grid. Grading is not possible. A grid size of 5m was used. This gave 1 728 000 cells. 5m was used as Waibel et al. (2017) in a similar validation of C_p on a rectangular building showed that it gave good results and smaller sizes gave cell counts that were judged to be too high.

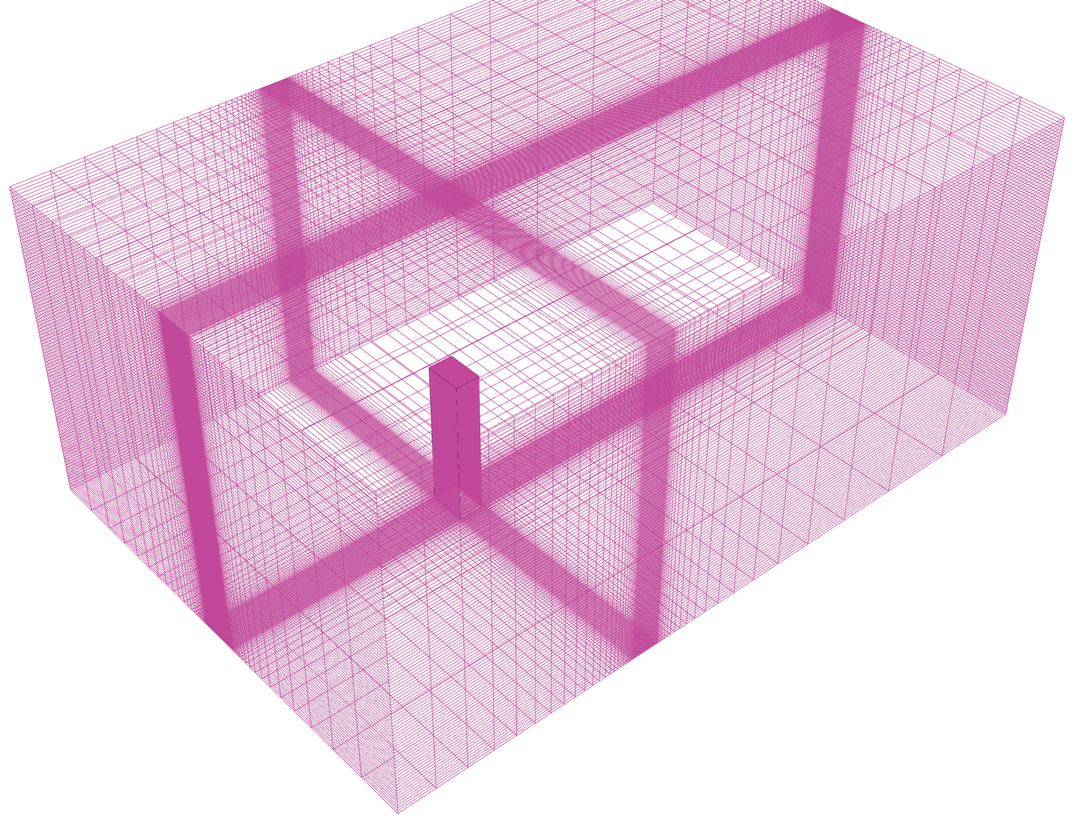


Figure 3-8: Wind tunnel domain mesh from Butterfly

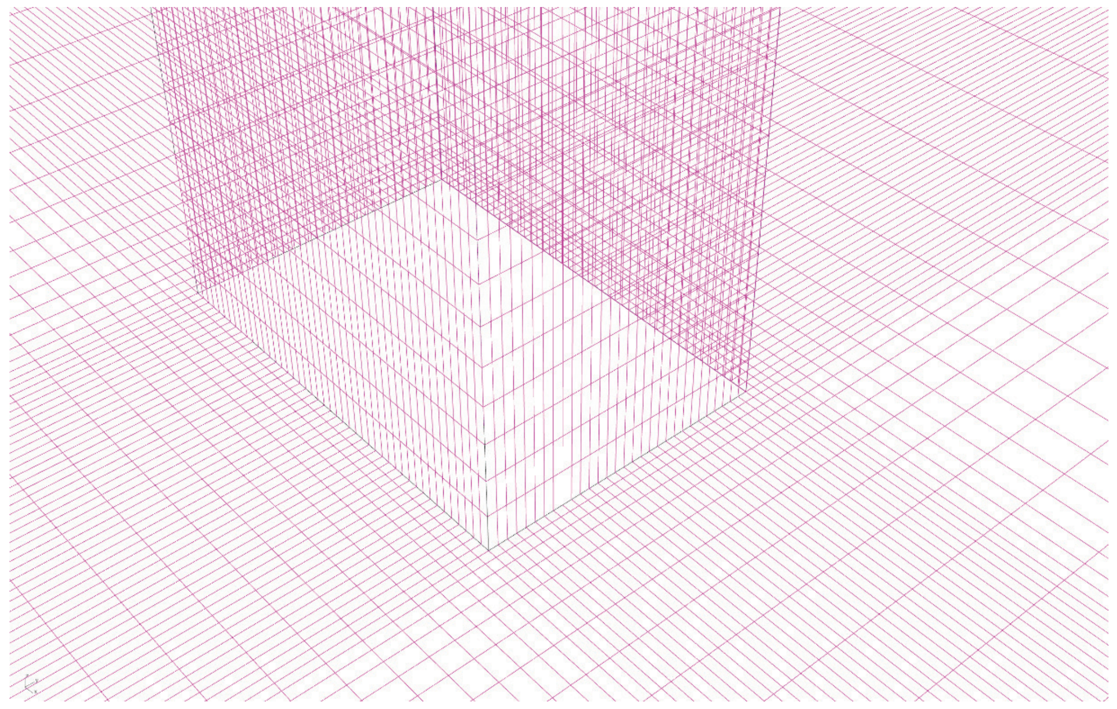


Figure 3-9: Domain mesh at building geometry from Butterfly showing grading

Turbulence models

The turbulence models evaluated were standard $k-\varepsilon$, Realizable $k-\varepsilon$, and RNG $k-\varepsilon$. FFD at present does not have turbulence modelling. To simulate the effects of turbulence in GH Wind the kinematic viscosity, v, of air was set to 0.1, compared to the true value of 1.5e-5, as a means of artificially lowering Reynold's number (Waibel et al., 2017).

Solution

In Butterfly the residuals limit was set at 0.0001 and max iterations at 30 000. The solution was run in serial (on a single processor core). Terrain category 6 was selected which corresponds to an urban area. To obtain the results, probes were placed at the pressure tap locations in the model to get the pressure at each point, P_i. Another probe was placed at the reference point location to obtain P_0 and U_0. These values were then put into a custom GH Python script which calculated C_{pi}.

GH Wind is natively parallelized. The time step was set at, dt = 0.1 and the time horizon (the sum of dt until the calculation stops) is set at 60, totalling 600 interactions. Terrain category 3 corresponding to an urban area is selected. GH Wind has a component (Cp Visualiser) which shows the C_p values over a given mesh at the vertices. The surface of the CAARC building was meshed with UV dimensions that allowed the vertices to match up with pressure tap points and the values were read from those.

After each analysis, the C_p values at each location was recorded along with the time taken in a table. The C_p values from TJ(D) were also recorded in the table for comparison. The settings are summarised in Table 3-1.

Solver	Butterfly	GH Wind
Turbulence models	k – ε, Realizable k – ε, and RNG k – ε	N/A (set v = 0.1)
Cell size (m)	0.987552	5
no. of cells	615 120	1 728 000
Domain (m)	900 x 600 x 400	900 x 600 x 400
Iterations	30000	600

Table 3-1: Validation study settings

Results

The C_p results between tests vary in different amounts from the TJ(D) results (see Appendix 2). The RNG kEpsilon model from Butterfly showed the closest C_p values to TJ(D) followed by standard kEpsilon. The values from the RNG kEpsilon simulation also closely match those found by Meng et al. (2018). The realizable KE model gave unrealistic C_p values as C_p generally should not be greater than 1.0 for incompressible flows (Aynsley, 1999). The reason for this result is unknown thus, it was eliminated. Time wise, RNG kEpsilon, though most accurate took 42.6 hours to complete. Standard kEpsilon took 41.7 hours and realizable KE, 37.4 hours.

The FFD analysis, as expected, took a much shorter time to complete, 9.2 hours, though it is still a very long time. The C_p values, however, were much lower than the TJ(D) and Butterfly results. The average difference between FFD and TJ(D) C_p values was 0.39 whereas for RNG and standard kEpsilon it was 0.18. C. Waibel in personal correspondence on 12 February 2019, stated that this is expected as it is the nature of FFD at present to under-predict C_p values. However, for both FFD and the Butterfly calculations the trend in C_p values is similar to TJ(D) as shown in Figure 3-10.

For the Butterfly results, the C_p values are quite close at the front face of the building (1 - 5). On the sides and rear (6 – 20) the values deviate more. RNG kEpsilon more closely follows the trend of the TJ(D) values compared to standard KE which has a more rounded graph shape. This is likely due to standard kEpsilon's poor performance in predicting flow in separated regions (Tamura et al., 2008).

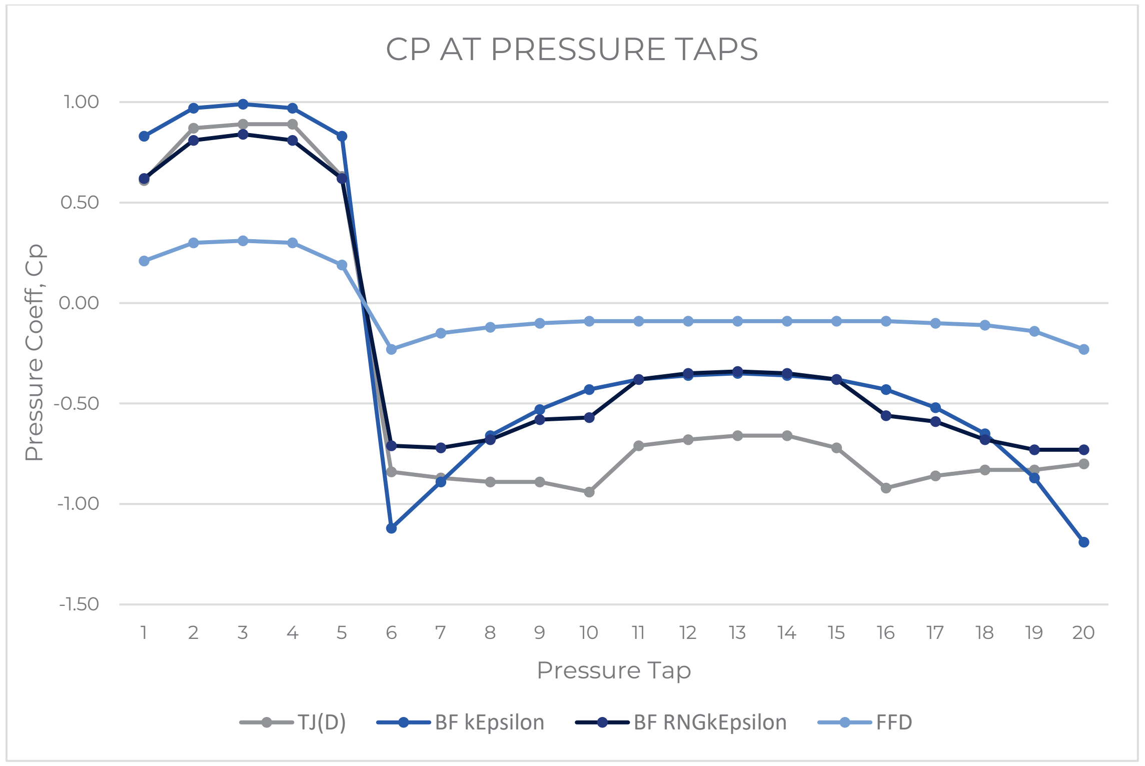


Figure 3-10: Analysis results

The FFD results in the graph as well follow the general trend albeit with a much smoother line than TJ(D) or RNG kEpsilon. Figure 3-11 shows the absolute deviation of the C_p results from the simulation to those from TJ(D) per face of the building. One can see that FFD is the highest others while kEpsilon and RNG are lower.

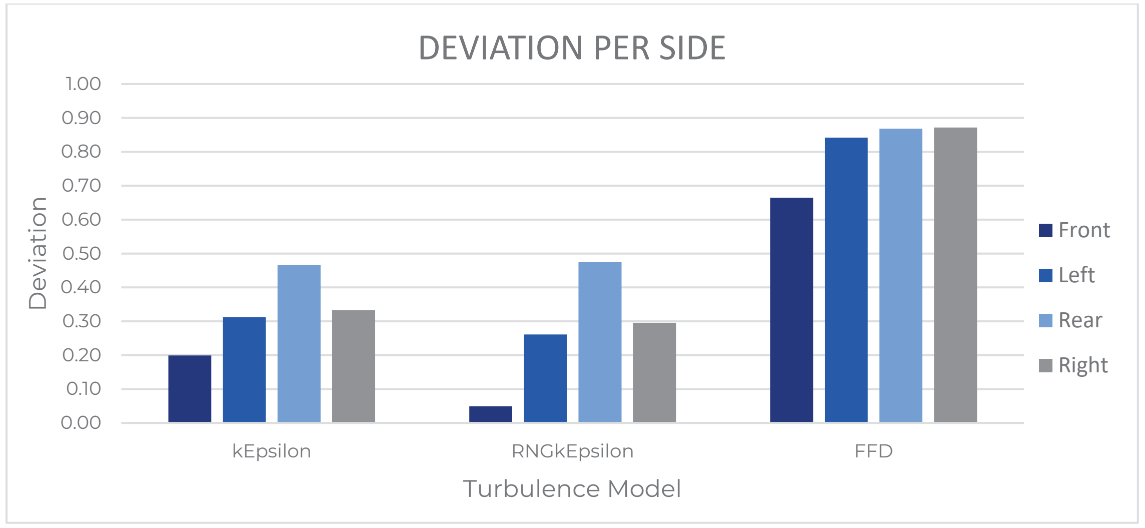


Figure 3-11: Graph of deviation per building side

Conclusions

As expected, the CFD analysis of Butterfly gave closer results to the experimental results. RNG kEpsilon was shown to perform the best by more closely following the graph trend from the wind tunnel results. However, in the scope of this thesis where quick results are desired, the time taken for analysis is still too high though there is potential to reduce it.

The GH Wind analysis was indeed much quicker than Butterfly although the lower quantities for C_p present a problem. In personal correspondence on February 12, 2019, C. Waibel suggested calibrating the reference point at which the static pressure and velocity are found by varying the height. This was tried but the values were not able to be matched to the Butterfly or TJ(D) results. Scaling the C_p was also suggested provided that the error is somewhat consistent. To evaluate this a graph of the absolute difference between C_p of each analysis and TJ(D) at each pressure tap was plotted (Figure 3-12).

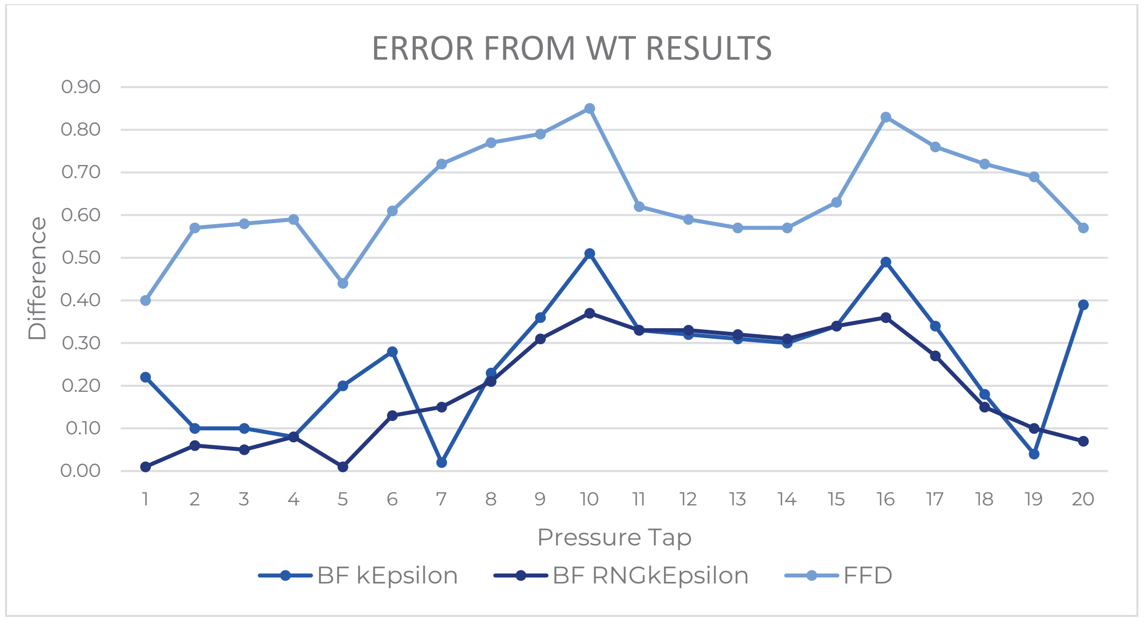


Figure 3-12: Error between TJ(D) results and CFD analyses

FFD varies between 0.4 to 0.8 difference compared to TJ(D) with a standard deviation of 0.14. The average can be used but it cannot be known whether this scale factor can be used for all analyses. Another problem of GH Wind is that it, at present, does not automatically stop the simulation upon the convergence of residuals. Also, when the residuals were plotted, they started to diverge again which C. Waibel, in the same personal correspondence above, stated that this is also a problem native to FFD. Further recommendations given in personal correspondence with C. Waibel on 13 February 2019 were to reduce the width of the domain but extend the leeward portion. This gives the flow time to reattach which would give more numerically accurate results.

It must be noted that while this validation has provided a good foundation for CFD procedure and expectations, the results and conclusions drawn are for a simple rectangular tower which is not the intended case for this thesis. It remains to be seen if these conclusions can be applied to more complex geometry buildings.

3.2.3 Sensitivity analysis

The aim of the sensitivity analysis portion was to determine how much the calculation time can be reduced while maintaining reasonable accuracy. The biggest factors affecting CFD/FFD simulation time are the cell size and number of iterations. The first sensitivity analysis using the number of iterations was done with the CAARC model. However, tests for cell size were done on the Absolute Tower model. The tests were again run on a Windows 10 PC with an Intel® Core™ i5-3470 CPU @ 3.20GHz, and 16GB of RAM.

Number of Iterations

It was noticed from the validation study in Butterfly that at 30 000 iterations the solution did not converge on its own even though the residuals looked sufficiently low. It was inferred that the solution could be stopped even earlier and still give satisfactory results cutting down on time. This was confirmed by plotting the residuals of the solution where it is seen by 10 000 iterations the residuals are sufficiently below the threshold of 0.0001 with a few errant peaks (Figure 3-13). Analyses with 10 000 and 5000 max iterations were done to test this theory. The C_p values were nearly identical for each pressure tap and the analysis took 15.7h and 6.95h respectively (see Appendix 3). Much quicker but still very long.

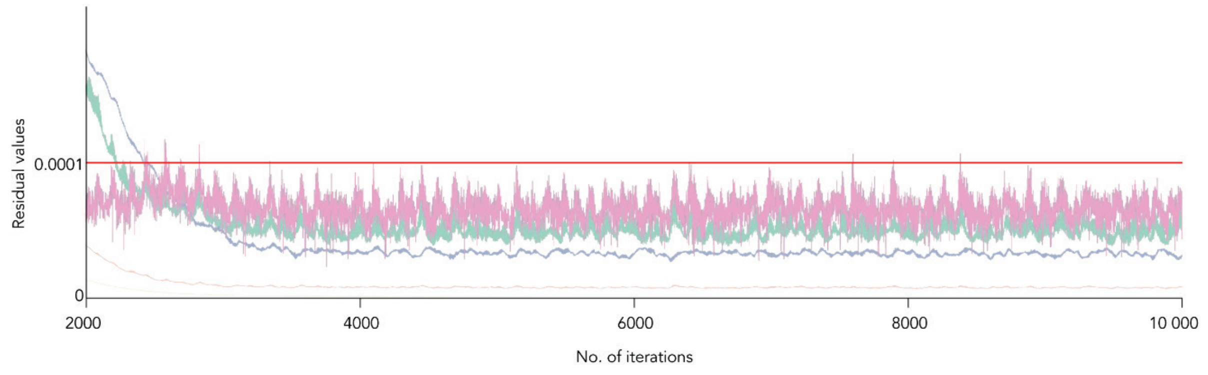


Figure 3-13: Residuals plot of RNG kEpsilon analysis

The GH wind residuals were not so straightforward. As seen in Figure 3-14 the graph decreases but at a certain point begins to increase again. In personal correspondence with Christoph Waibel, the developer of the GH Wind plugin on February 13, 2019, he stated that the divergence of residual values is a phenomenon that has been observed with many FFD simulations. His suggestion was to stop the simulation at a point before they begin to diverge. Based on the graph it was decided to reduce the simulation run from 600 iterations to 400 iterations. The results for C_p between 600 and 400 iterations were identical (Appendix 3).

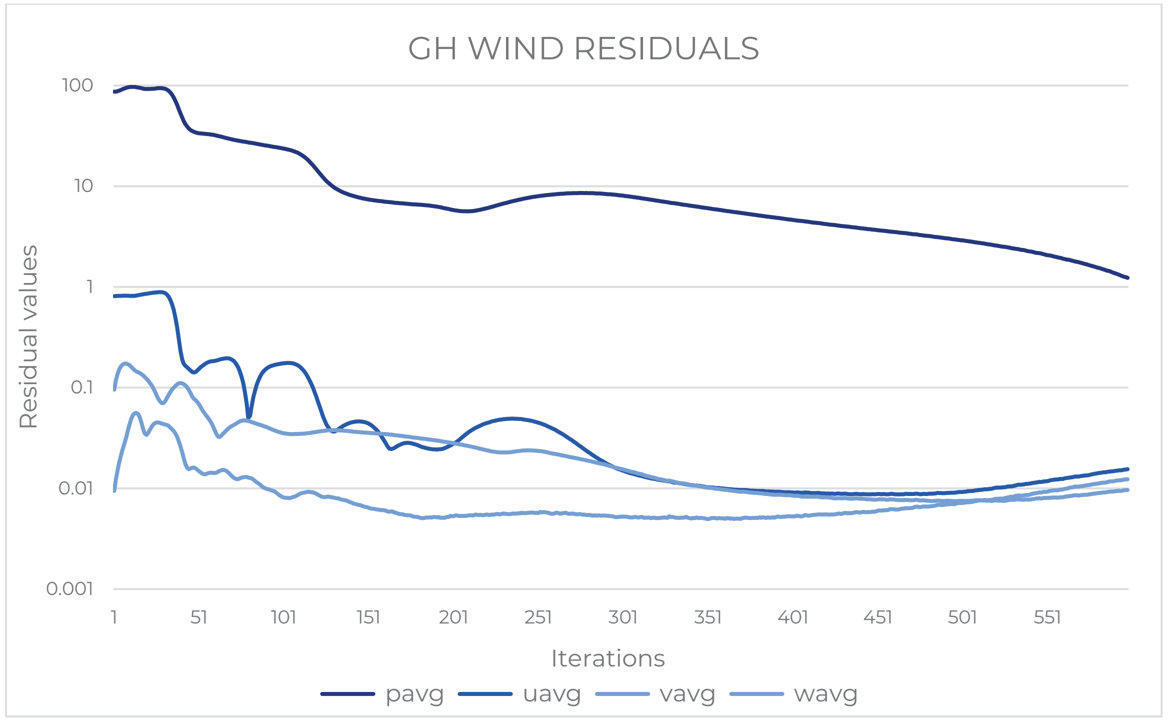


Figure 3-14: GH Wind residuals graph

Cell Size

Cell size and thus, the number of cells, are crucial parameters for CFD simulations since they greatly affect time and accuracy (Franke et al., 2007). The model of the Absolute Tower was used for this test. To compare accuracy a similar setup to the previous validation was used where the C_p at 30 points on the front and back of the building was recorded (Figure 3-15). The domain has the following dimensions: windward = 3H, Leeward = 10H, sides = 2.3H, and top = 2.3H where H is the height of the building based on recommendations from Franke et al. (2007). This corresponds to a tunnel size of 826.27m x 2258.35m x 392.15m (LxWxH). Wind speed was set at 30 m/s which corresponds to a violent storm on the Beaufort scale in order to test an extreme case. Terrain category was chosen for an urban site (roughness length = 1m). The tests were done in both Butterfly with RNG kEpsilon turbulence model and GH Wind.

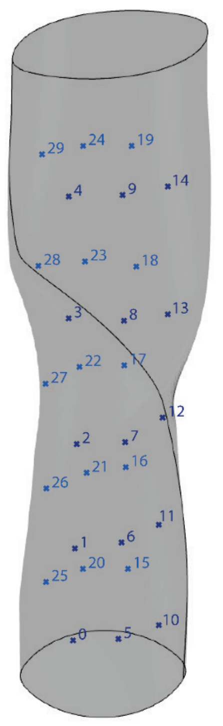


Figure 3-15: Pressure tap locations

Butterfly

The tests were performed for various cell sizes. These start from the standard minimum size of the length of the shortest side of the building divided by 10 for the coarsest mesh (Franke et al., 2007). Then, divided each time by $\sqrt{2}$ for Medium, Fine, SuperFine, and XXFine. The simulation was run for 10 000 iterations and this number was verified as reasonable after checking a graph of the residuals.

Test	MAD_1	MAD_2	MAD_3	MAD_4	MAD_5
Turbulence model	RNGkEpsilon	RNGkEpsilon	RNGkEpsilon	RNGkEpsilon	RNGkEpsilon
Resolution	Coarse	Medium	Fine	Super Fine	XXFine
Cell size (m)	4.18	2.96	2.09	1.48	1.08
no. of cells	176545	236050	346647	525640	732422
Time	5.7h	6.3h	8.6h	14.5h	20.3h
Iterations	10000	10000	10000	10000	10000

Table 3-2: Settings and time results for Butterfly simulations

Table 3-2 shows the settings used as well as the time taken for each run. MAD_5 with a cell size of 1.08 is assumed to be most accurate however, a time of 20.3 hours is too high.

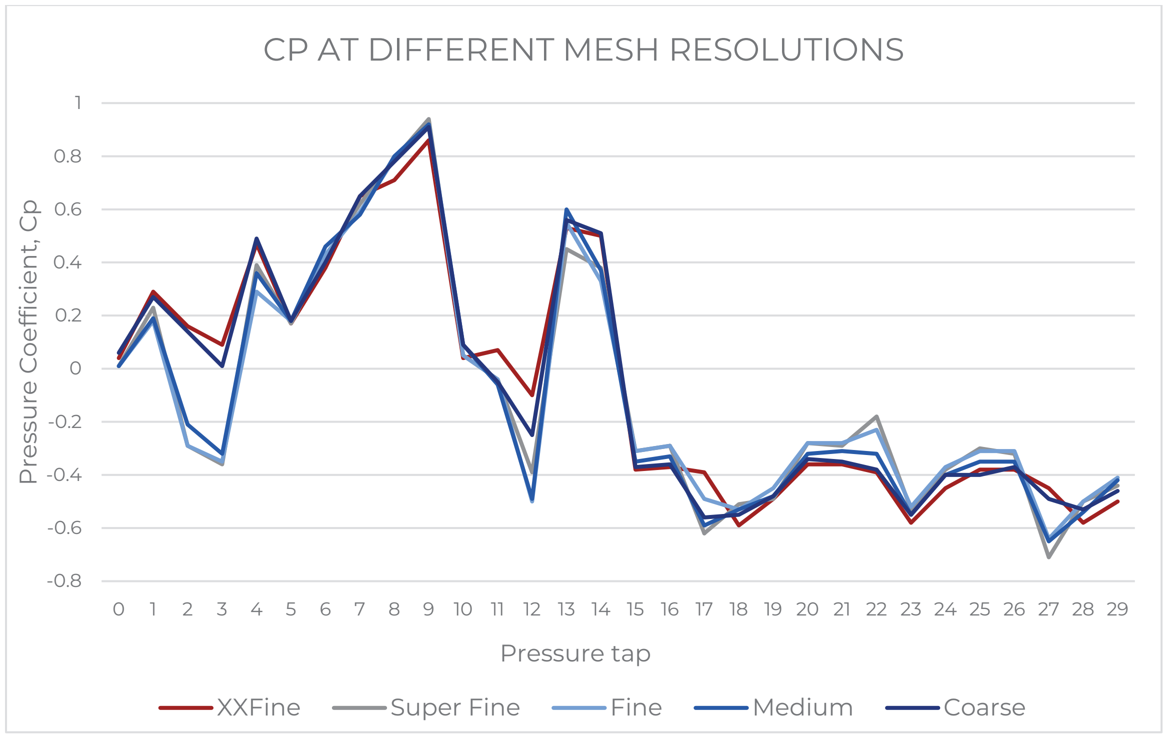


Figure 3-16: Butterfly results

	Mesh Resolution			
	Coarse	**Medium**	**Fine**	**SuperFine**
Front	-18.46%	28.08%	8.71%	-0.77%
Rear	17.63%	15.36%	10.19%	-1.46%

Table 3-3: Deviation from XXFine results

Table 3-3 shows the percentage deviation between the results at XXFine resolution and the others. Fine resolution has a relatively small deviation compared to Medium and a time of 8.6h is reasonable compared to 14.5h for SuperFine. Medium and Coarse have higher deviations but the time reductions are significant compared to the others especially if it has to be repeated in an optimisation loop. From Figure 3-16 it is that the values closely match in trend showing the precision of Butterfly.

GH Wind

For GH Wind the simulations were done for three cell sizes: 10m, 8m, and 6m. The time taken for each is shown in Table 3-4 below. The time increases exponentially with each jump in cell size. Moreover, the results do not appear to be accurate. The values are underpredicted as expected however, the trend mostly does not follow the Butterfly (Figure 3-17).

Test	MAD_5	MAD_6	MAD_7
Turbulence model	FFD	FFD	FFD
Resolution	Medium	Fine	SuperFine
Cell size (m)	10	8	6
no. of cells	390830	774387	1824912
Time	3.4h	7.1h	11.8h
Iterations	400	400	400

Table 3-4: Setting and time result for GH Wind simulations

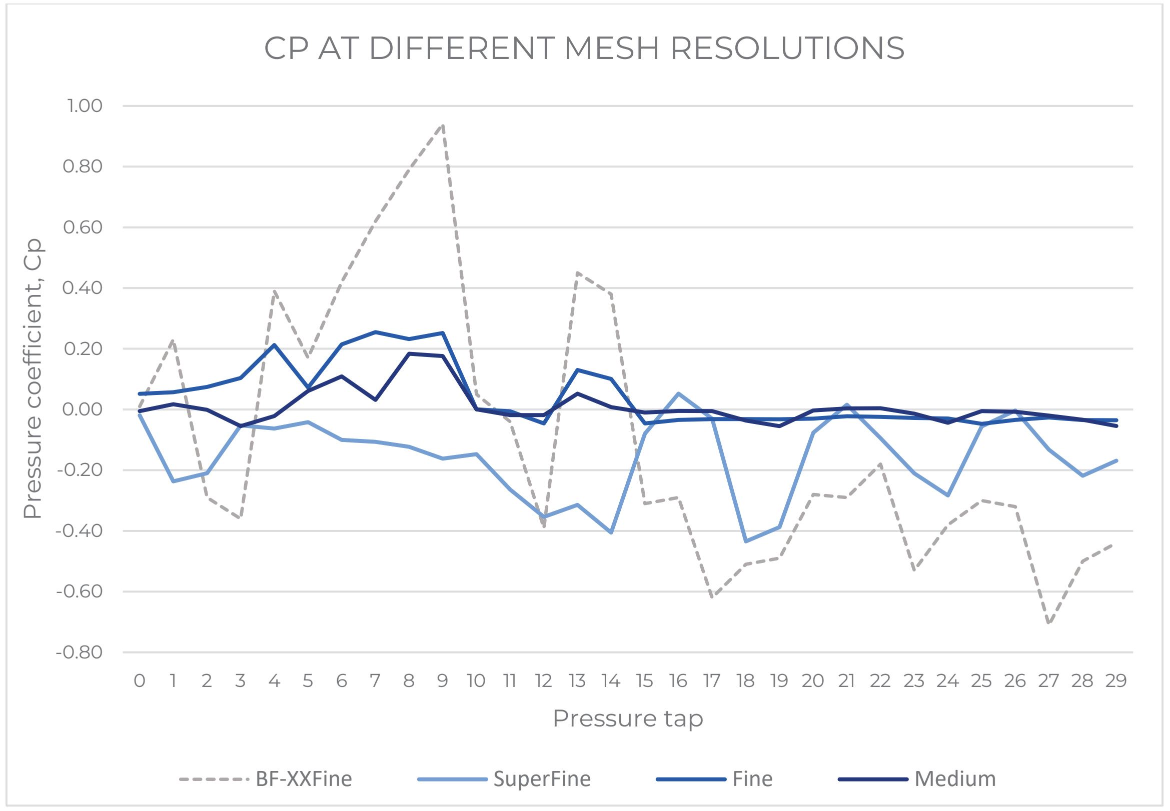


Figure 3-17: GH Wind results

Conclusions

Butterfly with the RNG kEpsilon model looked to perform very well. There was good correspondence between values as the graphs are very close. The time taken is still large however, it can be further reduced by lowering the number of iterations and using the coarse mesh setting. While accuracy may be lowered it may still be reasonable for early-stage evaluation especially as part of optimisation which requires many iterations to be run.

GH Wind appears to be unsuitable for complex geometries as the results do not appear to be precise. It could be made better by decreasing cell size however, based on the previous tests it appears it will end up taking a longer time than Butterfly. The values could also be scaled to closer match Butterfly however, it requires a lot of experimentation to come to a constant value that would be good for all geometries. At this moment, it seems futile to continue using GH Wind for this case. The main advantage was its potential time savings, but it appears that to work for complex geometries the cell size would have to be reduced so much that it will take longer while still being less accurate than Butterfly. It seems reasonable to think that this is due to the meshing of the geometry. While the Snappy Hex Mesh of Butterfly can adapt itself to complex geometries the voxelization of GH Wind is a much more inaccurate representation (Figure 3-18). Therefore, going forward Butterfly with RNG kEpsilon will be used.

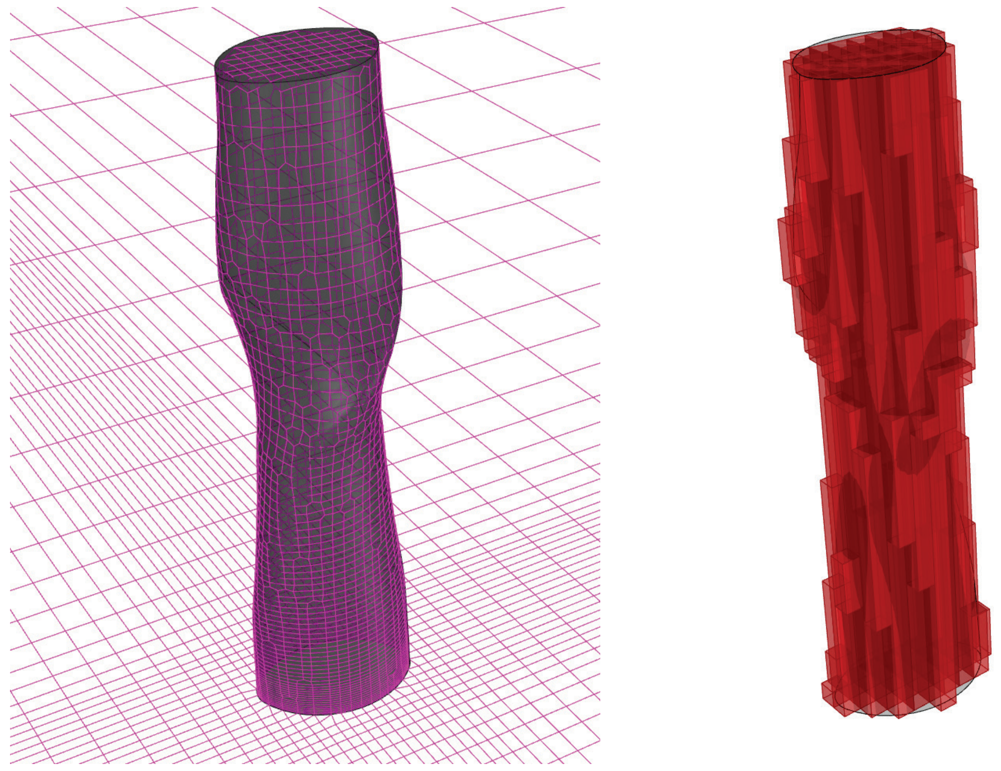


Figure 3-18: Butterfly Snappy Hex Mesh (left) and GH Wind voxelization (right)

3.3 FSI

A partitioned FSI procedure will be used for this tool. This involves creating an algorithm to join the output of the CFD solver to the input of the FEA solver. Specifically, the pressure on the façade into a FEM using an upright fixed beam with a number of point loads and moments. Thus, the procedure follows two interconnected lanes: one for translating the pressure values and one for translating the geometry (Figure 3-19). These are then assembled and analysed with the FEA solver to obtain the structural reactions.

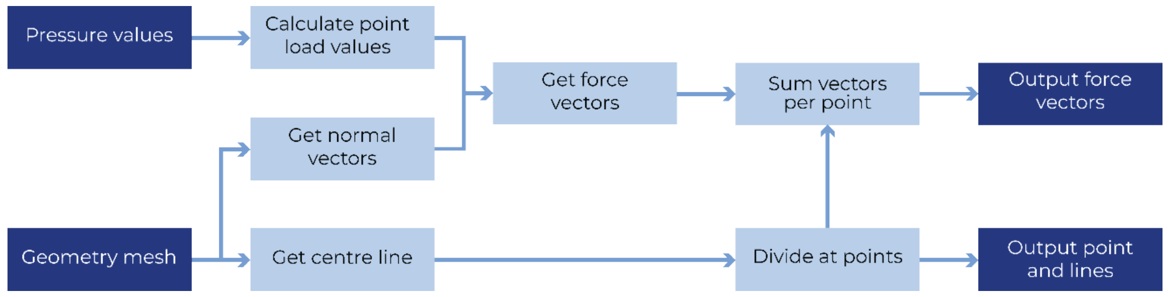


Figure 3-19: FSI translation algorithm workflow

3.3.1 Translation procedure

This translation procedure developed in Grasshopper takes the pressure loads on a mesh of the building surface from Butterfly and consolidates it into point loads and moments along the core of the building represented by line segments. The idea was to divide the mesh into sections and sum the forces on each sector to get point loads and moments. The script layout is shown below in Figure 3-20.

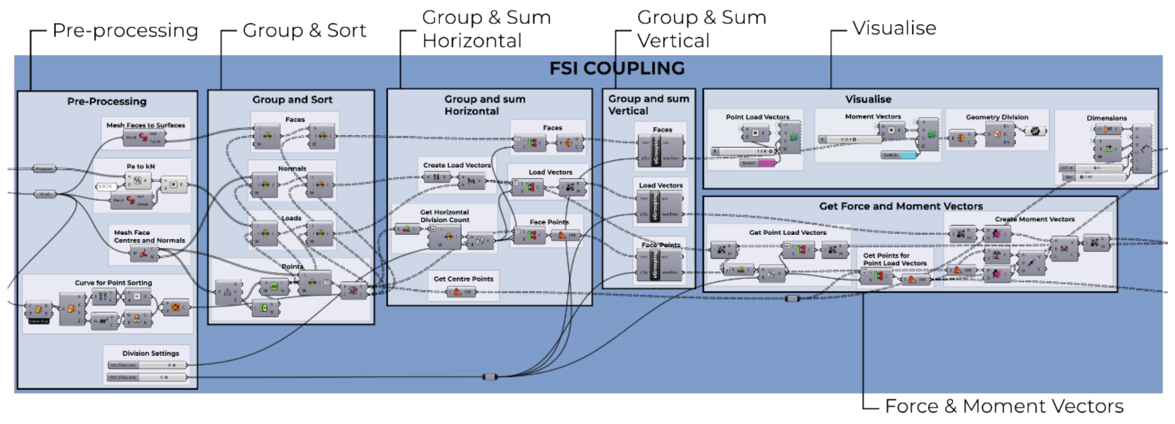


Figure 3-20: FSI translation script

Butterfly outputs a mesh and a list of pressures in Pascals which corresponds to each mesh face (Figure 3-21a). The resolution of the mesh is set before the solution and is unrelated to the resolution of the CFD mesh. First, the list of mesh faces is sorted. Since the list comes out from Butterfly in an unpredictable order this sorting algorithm ensures the lists will always be in the same order. The centre point of each face is obtained and then sorted and grouped by height so that all points with the same Z component are in a branch of the data tree. Then using a bounding circle, the groups of points are sorted in an anticlockwise direction. The pressure values are multiplied by the area of their corresponding mesh face to give a force in kN. This becomes the magnitude of the normal vectors. This list, as well as the list of face normal vectors, is sorted using the same ordering algorithm for the list of points so that each element of one list corresponds to the right element in the other lists. The points at each height are average to get the centre points of the building. The force vectors are summed at each height to give a point load acting at each corresponding centre point. Input for the number of vertical divisions is given which determines where the central polyline is split and where forces will be applied. The point force vectors are summed and applied to their nearest division point.

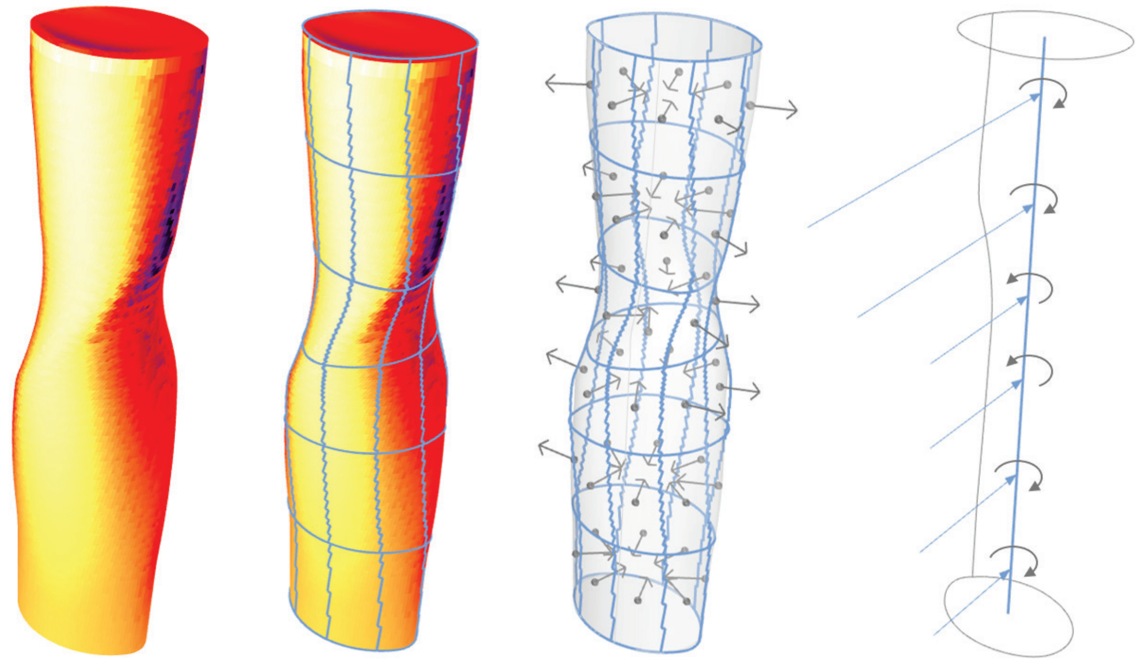


Figure 3-21: (a) Mesh from Butterfly with corresponding pressures, (b) division into segments, (c) force vectors summed per segment and applied to centre point plus distance vector from centre of the building, (d) force vectors and moment vectors applied to core beam

To get moment vectors an input for the number of horizontal divisions is also given. This, as well as the number of vertical divisions, guides an algorithm for arranging the lists of force vectors per mesh face and corresponding face points into rectangular segments on the building mesh (Figure 3-21b). These force vectors are then summed per segment and applied to the centre point of the segment (Figure 3-21c). The moment vectors are obtained from the cross product of the segment force vectors and distance vectors and applied to the core at the division points (Figure 3-21d). These force and moment load vectors can then be put into Karamba along with a polyline of the core split at the points of application of the loads.

3.3.2 Finite Element Analysis

To perform structural analysis in FEA software a Finite Element Model (FEM) is first constructed. Elements are the beams, shells, or other components being analysed. The loads, usually in the form of vectors, are added as well as the points on the elements that will be the supports. Finally, the material and size of the cross-section are given. This is assembled into the FEM where the solver can analyse the resulting structural behaviour and give results such as reaction forces and moments, deflections, and stresses.

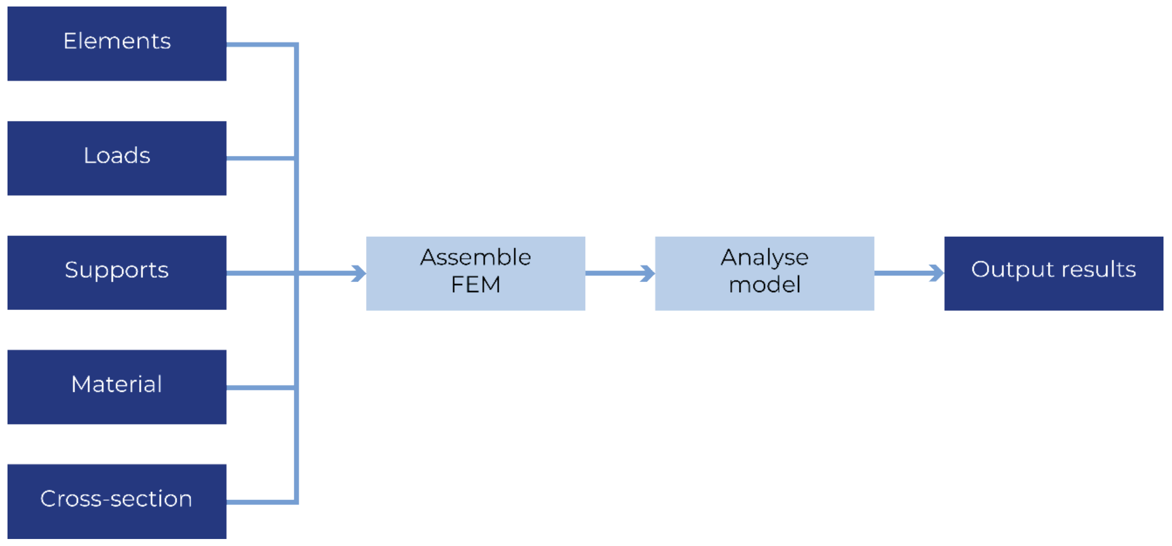


Figure 3-22: FEA workflow

Karamba version 1.3.1 was integrated into the Grasshopper tool. To perform FEA in Karamba the elements, loads, supports, cross-section, and material must be defined and assembled into the FEM in the "Assemble" component. The elements will be the lines obtained from splitting the central polyline of the

building which represents the core at the points the loads will be applied. The cross-section of the elements will be the dimensions of the core. In this case for the Absolute Tower, the core is 8.6m x 8.0m and 0.4m thick (Figure 3-23). The script layout can be seen in Figure 3-24.

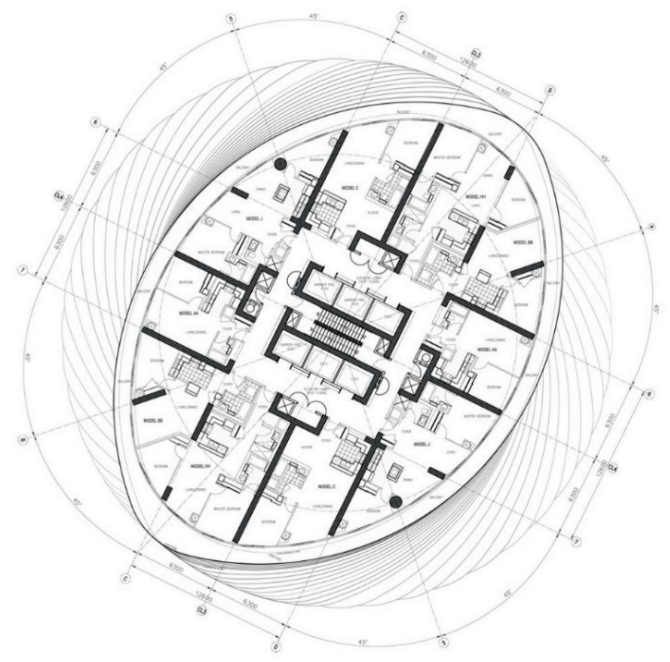


Figure 3-23: Typical floor plan of Absolute Tower © MAD Architects

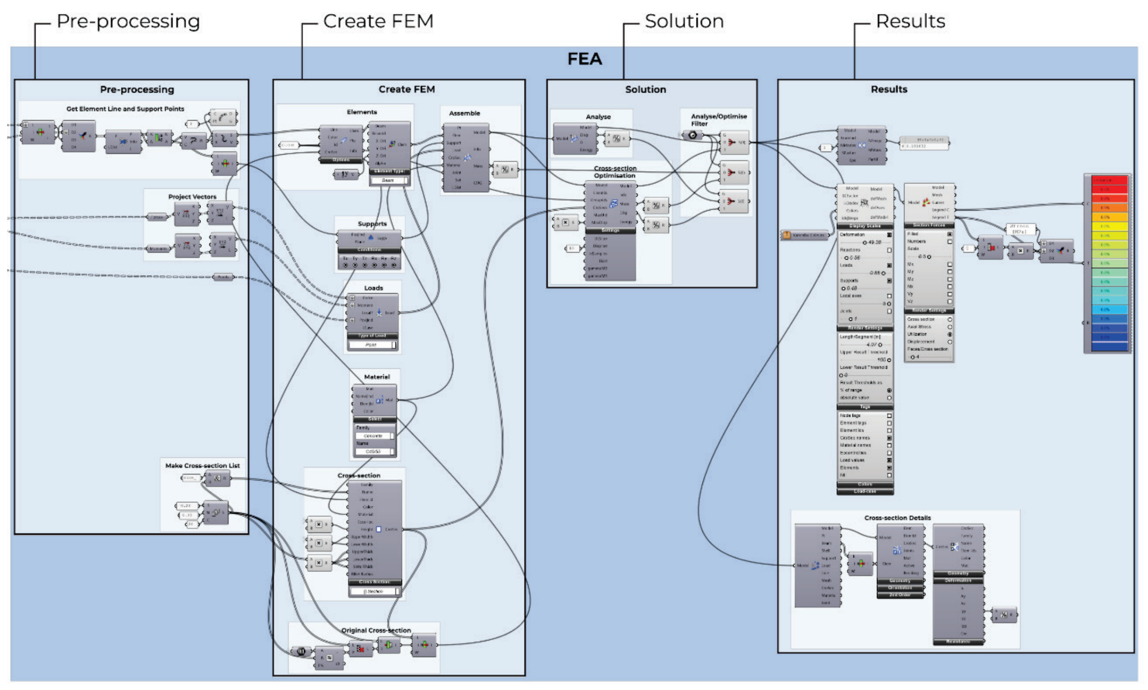


Figure 3-24: Karamba FEA script

The material chosen is C45/55 concrete. The supports are fixed in all degrees of freedom. The loads are the point forces and moments obtained from the translation procedure. After the model is assembled and analysed the results are obtained (Figure 3-25 and Figure 3-26).

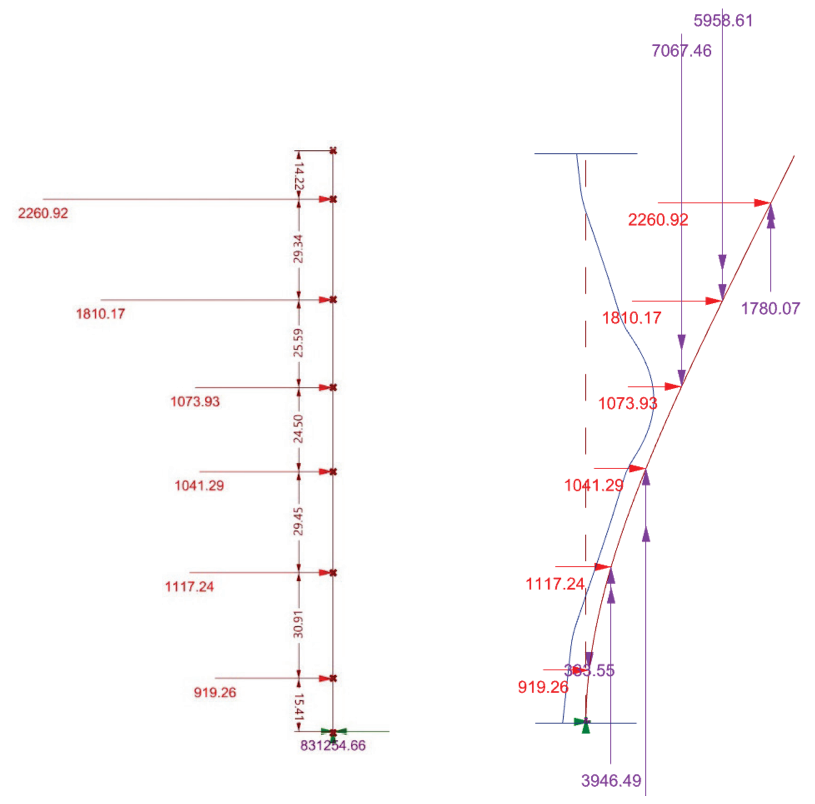


Figure 3-25: FEM in Karamba (left) and deflected model (right)

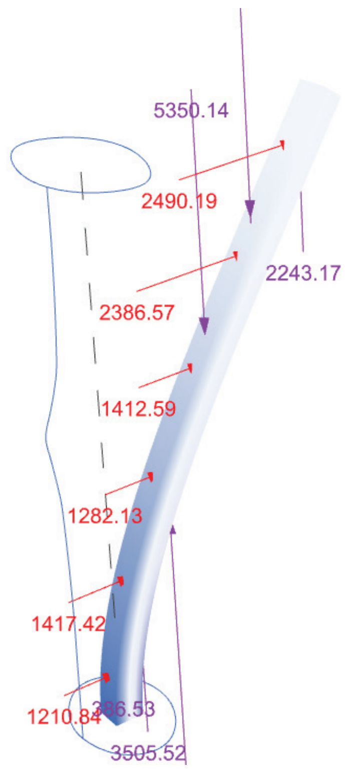


Figure 3-26: Deflected model of core showing stresses

The script was run for 10 000 iterations at Fine mesh setting. Max deflection at the top was 1.28m. This is large as the SLS requirement from Eurocode 6 for concrete of L/250 gives max deflection of 0.677m. This could be due to the fact that only the core is modelled to take all the load when in reality there would be additional columns, floors, and beams that could add to the stiffness. Additionally, these loads are for an extreme wind case. Other results include max stress of 25.7 MPa, base overturning moment of 831 254 kNm, and base twisting moment of 3717 kNm. These results were also verified by hand calculations (see Appendix 4).

3.3.3 Conclusions

The devised FSI translation procedure was able to be implemented successfully in Grasshopper. Through examinatio, the results align with what was expected which shows that the sorting and grouping algorithms perform as they should with the test buildings used.

For the FEA the deflection obtained in this test is very large thus it may not be reasonable to compare to Eurocode limits. For the optimisation portion, the improvements could be minimised relative to the first obtained values. Nonetheless, the results show that the script works and is able to produce results for the wind-induced reactions on a building structure. The wind forces were quite significant on the building, especially at higher points. Twisting moments were greater near the middle of the tower corresponding to the section with the most twist which was expected. However, the twisting moment is relatively small and its contribution to deflection is negligible with -0.0037 degree max rotation possibly due to the slender shape of the tower.

Dynamic loading is still a factor that is not analysed. Although the displacement from torque is small it could have a significant impact on comfort if the frequency of displacement is significant. This also applies even more so for crosswind displacements due to vortex shedding. This could be accounted for analytically using custom scripts in further research.

3.4 Optimisation

The next step in the research involved exploring the viability of optimisation with the FSI tool. The goal is to reduce structural objectives solely by manipulating geometry. Thus, an array of parameters which morph the geometry would be the input variables while an output of the FSI algorithm would be the objective of the optimisation algorithm. This joining of an optimisation component with the FSI method thus creates a Fluid-Structure Interaction based Optimisation (FSIO) method. It involves iteratively changing the variables and reading the objective until an optimum is found. Thus, Figure 3-27 shows the workflow of the complete method.

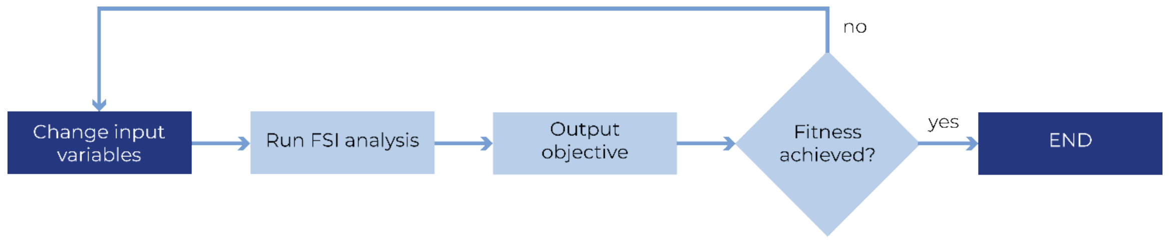
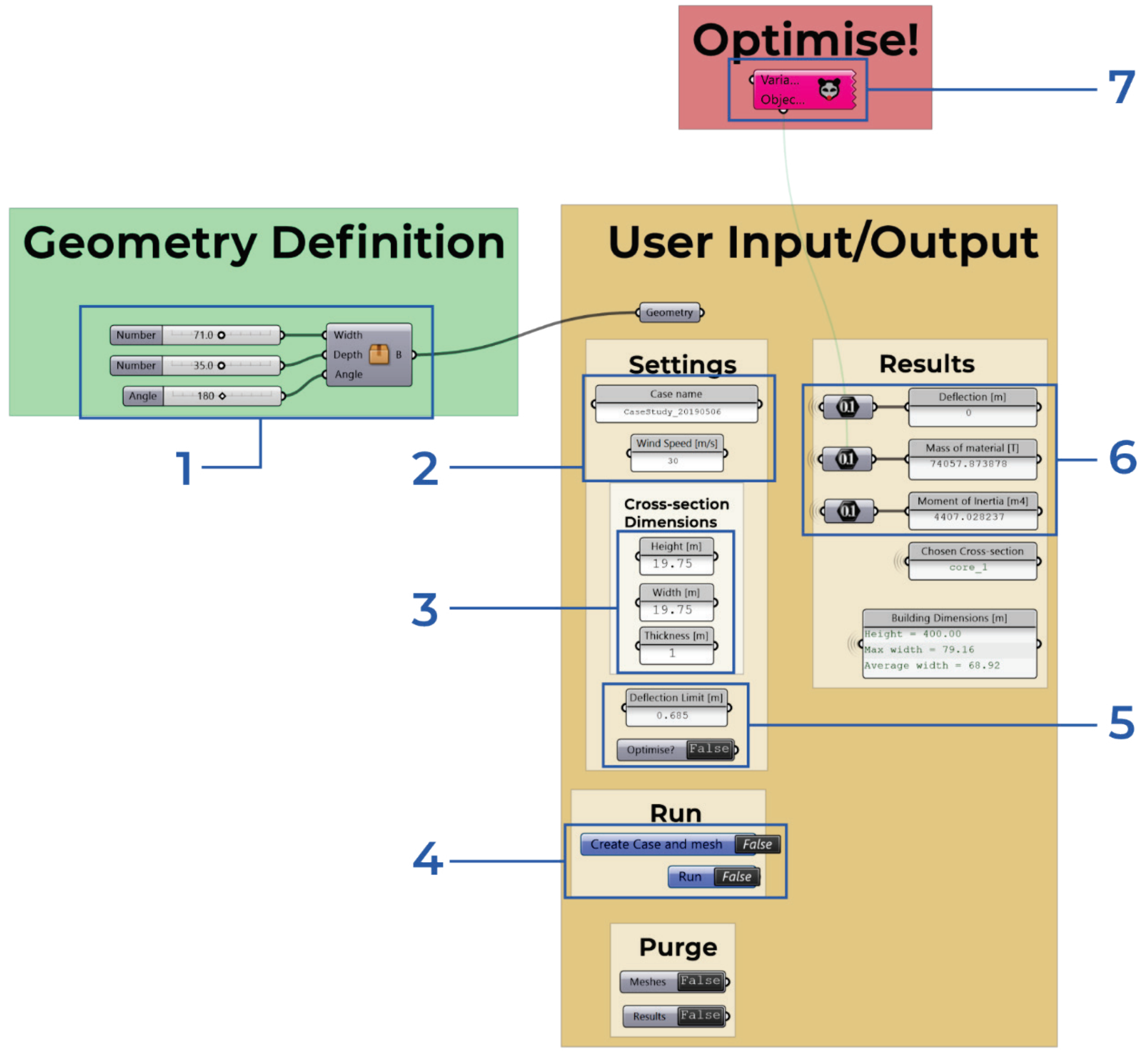


Figure 3-27: Optimisation workflow

3.4.1 Optimisation setup

The Opossum plugin was used with the developed FSI tool in Grasshopper. Different options for objectives and variables were explored to determine a viable optimisation problem for this case. All optimisation tests were run on a Windows 10 PC with Intel® Core™ i7-5820K CPU @ 3.30GHz 6 Cores and 16GB RAM. In order to prepare the full FSI script for optimisation, it must be able to run without any other input apart from a change in the chosen input variables. This required some changes to the script. A small user input (UI) area was made and the script modified so that only the settings in this area need to be changed in order to run the tool. The layout and steps are illustrated in Figure 3-28.



Number 71.0
Number 35.0
Angle 180

Case name: CaseStudy_20190506
Wind Speed [m/s]: 30
Height [m]: 19.75
Width [m]: 19.75
Thickness [m]: 1
Deflection Limit [m]: 0.685
Optimise? False
Create Case and mesh False
Run False
Meshes False
Results False

Deflection [m]: 0
Mass of material [T]: 74057.873878
Moment of Inertia [m4]: 4407.028237
Chosen Cross-section: core_1
Building Dimensions [m]:
Height = 400.00
Max width = 79.16
Average width = 68.92

Figure 3-28: User Input (UI) area of script and steps to run the FSIO tool

The geometry is input first (step 1). This geometry must be parametrically defined in order to have input variables for Opossum to manipulate. The case study buildings were built parametrically to have 2 or 3 sliders controlling the geometry. A name is given to the case and a wind speed set as required by Butterfly (step 2). The cross-sectional dimensions of the core are then set as required for Karamba (step 3). After these settings have been input, one sets the first toggle to true to create the case and mesh and the second to run the FSI tool (step 4). The two toggles below (Purge) allow the user to clear the existing meshes and/or results. To the right is a readout of information about the building and the different objectives that can be used in the optimisation. After running an initial analysis to obtain baseline results the user then inputs the obtained in the deflection limit field and sets the "Optimise?" toggle to true (step 5). The "Optimise?" toggle enables Karamba Cross-Section Optimisation plugin to allow for optimising material mass as explained in the tests following section. If the user is only using deflection as the optimisation objective leave the toggle on False. Then, connect the objective output of Opossum to the desired objective and the variables input to the sliders controlling the geometry (step 6). Finally, open the Opossum component and run the optimisation (step 7).

3.4.2 Optimisation tests

The aim of these tests was to determine the extent to which structural performance due to wind can be optimised by making relatively small changes to the geometry of the building. By trying different arrangements of inputs, objectives, and settings for the optimisation algorithm with multiple buildings models the hope is to determine a robust optimisation problem that can be used for many different building models.

The case study buildings shown in section 3.1 were used in tests to formulate the optimisation problem. This involved exploring different input variables and ranges of values and objectives to determine the most meaningful arrangement. Since optimisation with CFD will take a very long time given the number of iterations to be done it was decided to try to minimise the length of the CFD simulations. Thus, a coarse mesh setting was used and max iterations for Butterfly was set to 2000. This would not give totally accurate results in absolute terms but by keeping these constant along with wind speed and other settings and only allowing the optimisation algorithm to manipulate the geometry one would still see a relative improvement in the objective which is still valuable.

The building models themselves were parametrically defined in order to allow the optimisation algorithm to manipulate the geometry and find a better performing arrangement. The parameters chosen to be modified allowed the building to change enough so that there would be an impact on wind reactions but still maintain much of the general architectural intent of the building. No parts were added or subtracted but existing features were morphed.

For this study, like the rest of the tool, the building models were built in Grasshopper as detailed below so that two or three characteristics were able to be modified using number sliders.

Optimisation 1 – Absolute tower with deflection

In the first optimisation test, the Absolute Tower model was used. This building is characterised by an elliptical cross-section that twists as it rises. It was decided to manipulate the size of this ellipse as well as the amount of twist. A base ellipse was created whose width and length are controlled by sliders. This curve was then copied and moved upwards in the position of each floor and each rotated according to the angles given in the original design by the architect. A slider was added to act as a multiplier to these angles so that the twist could be increased or decreased. These three sliders: base length, base width, and twist multiplier, were the input variables for the optimisation (Figure 3-29).

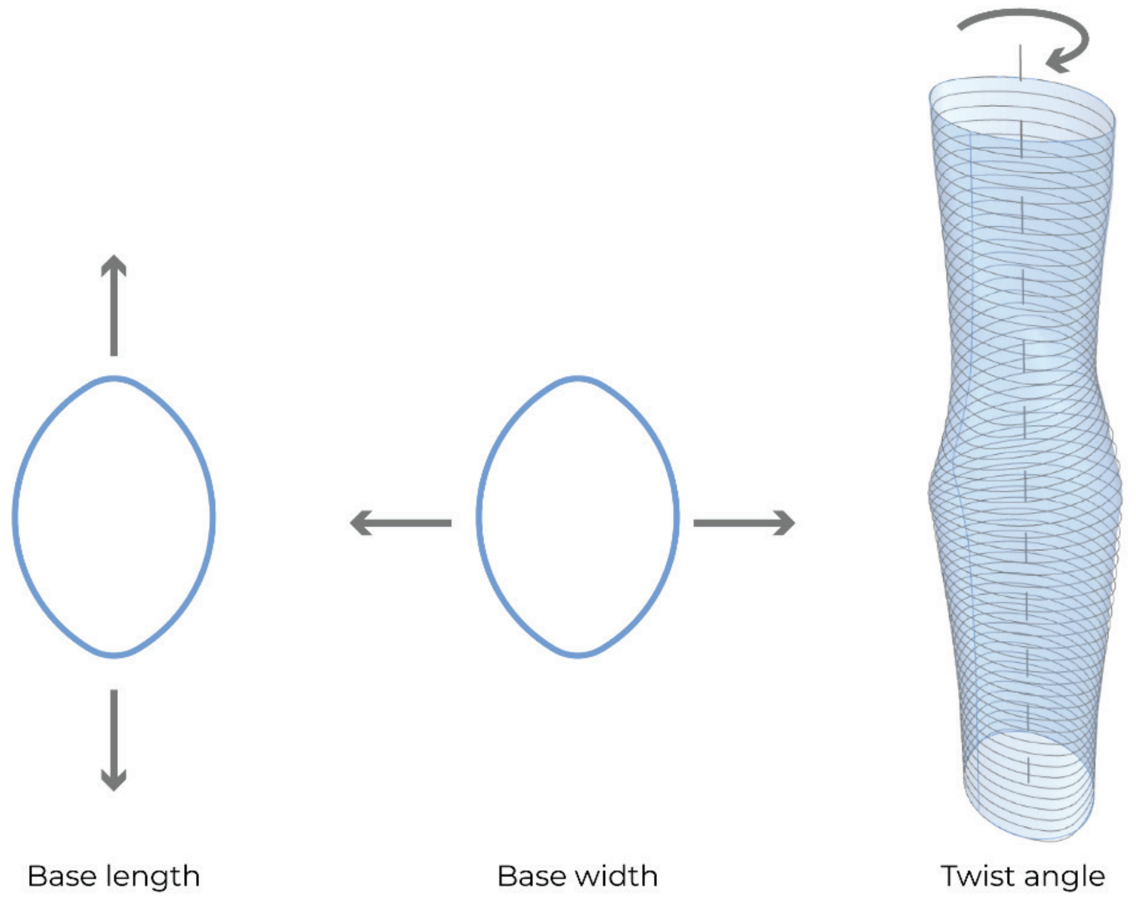


Figure 3-29: Absolute Tower model parameters

Case study	Absolute Tower		
Max iterations	60		
Parameters	Base length	Base width	Twist
Ranges	13.0 < x < 15.0	13.0 < x < 15.0	0.5 < x < 1.5
Objective	Deflection		

Table 3-5: Absolute Tower optimisation 1 settings

Table 3-5 shows the ranges and initial values of each input variable. The objective for this optimisation run will be to minimise deflection. The deflection value is obtained from Karamba. An initial FSI analysis was done to determine the baseline objective value. This gave a max deflection value of 1.5444m. In Opossum the RBFOpt algorithm was selected with max iterations of 60. After completion, the deflection saw a 38% reduction to 0.9454m with input variables of 13.25, 13.00, 0.50 for length, width, and twist respectively. Figure 3-30 shows the gradual improvement of the deflection objective with every iteration which appears to converge at around 40 iterations. The optimised shape (Figure 3-31) is rotated about 90° in the top portion. This results in a smaller area of high pressure on the front windward side of the building and thus lower point loads on the FEM.

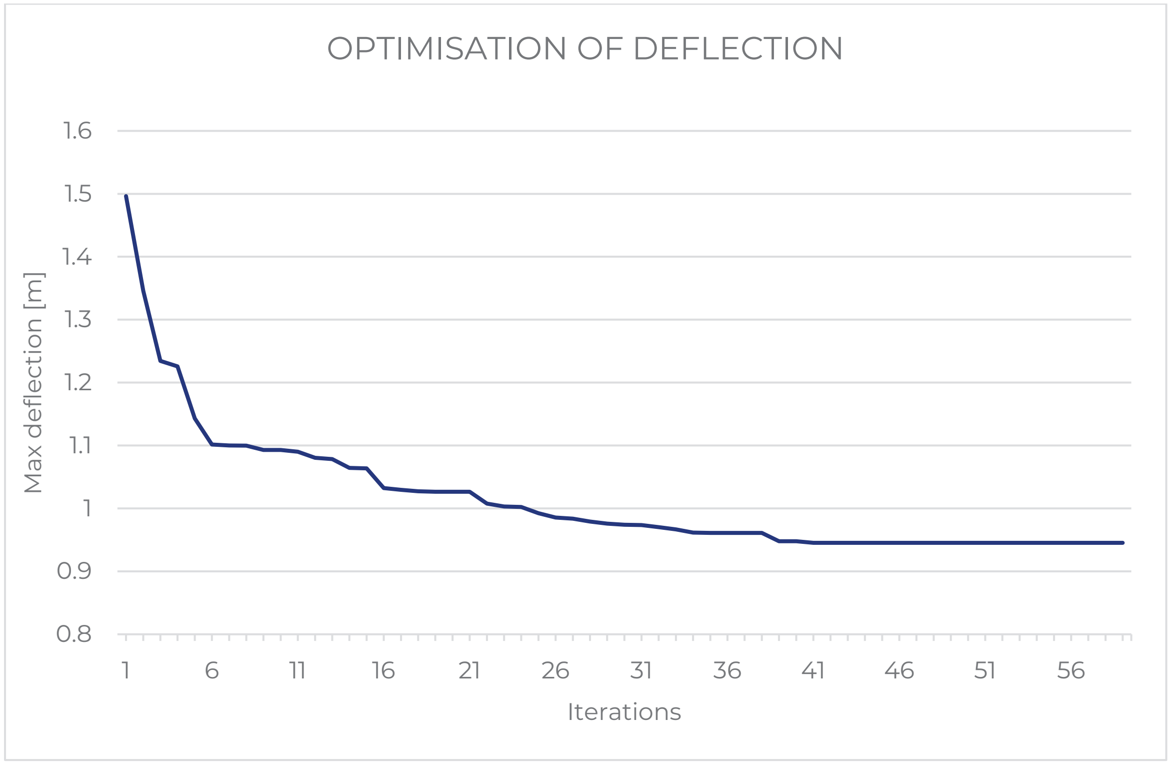


Figure 3-30: Results of Absolute Tower optimisation 1

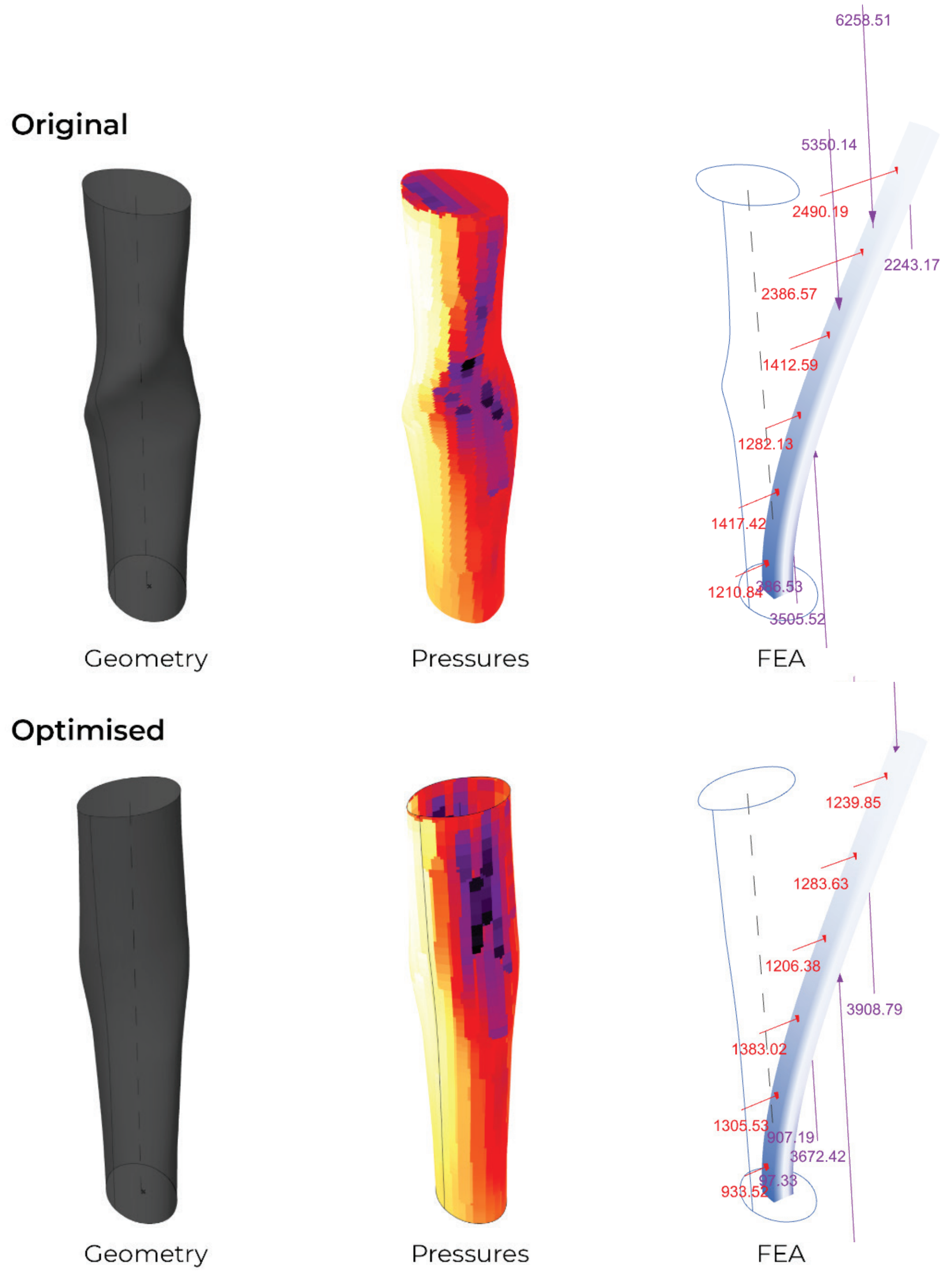


Figure 3-31: Absolute Tower optimisation 1

While this first test was successful in reducing the objective there was room for improvement. Firstly, the design space i.e. the range of values for the input variables was quite small thus there could be more optimums that are missed. Also, it was thought that material mass could be a more meaningful objective to minimise rather than just deflection as this can translate to lower cost and lower carbon footprint which is very valuable to building stakeholders.

Optimisation 2 – Absolute tower with mass

For the next optimisation trial, the ranges of input variables were expanded allowing the size to expand or contract more and also a greater range of twist so that the tower could go from double the twist to completely reversed and even no twist at all (Table 3-6). This was done in order to possibly catch an optimal configuration that may have been missed in the smaller design space of the previous optimisation run. This came with the caveat that the geometry may stray even further from the original architectural design.

Case study	Absolute Tower		
Max iterations	60		
Parameters	Base length	Base width	Twist
Ranges	12.0 < x < 16.0	12.0 < x < 16.0	-1.0 < x < 2.0
Objective	Material mass		
Cross-sections	0.10m to 0.80m in 0.05m increments		

Table 3-6: Absolute Tower optimisation 2 settings

In order to reduce material mass, the Cross-Section Optimisation (CSO) component in Karamba was used. This does not utilise an optimisation algorithm but rather consecutively searches a list of cross-sections, ideally sorted from lightest to heaviest, until it finds one that meets the requirements set for utilisation and deflection. Therefore, as the wind pressure reduces so does the stiffness required to stay within the deflection limit and thus a thinner cross-section will be selected resulting in reduced material.

For this test, a list of cross-sections for the core was created with the length and width, 8.6m and 8.0m, kept constant but thickness ranging from 0.1m to 0.8m in 0.05m intervals. The deflection limit was set at 1.5444m as per the initial analysis and max utilisation ratio kept at the default of 1.0. It was at this point that a deflection limit input was added in the UI area of the script. Also, a toggle was implemented so that a user can switch between “True” where the FEA runs through the CSO component to optimise material mass, or “False” where it optimises deflection by using the input cross-section thickness and running through Karamba’s first order analysis component as normal (Figure 3-28).

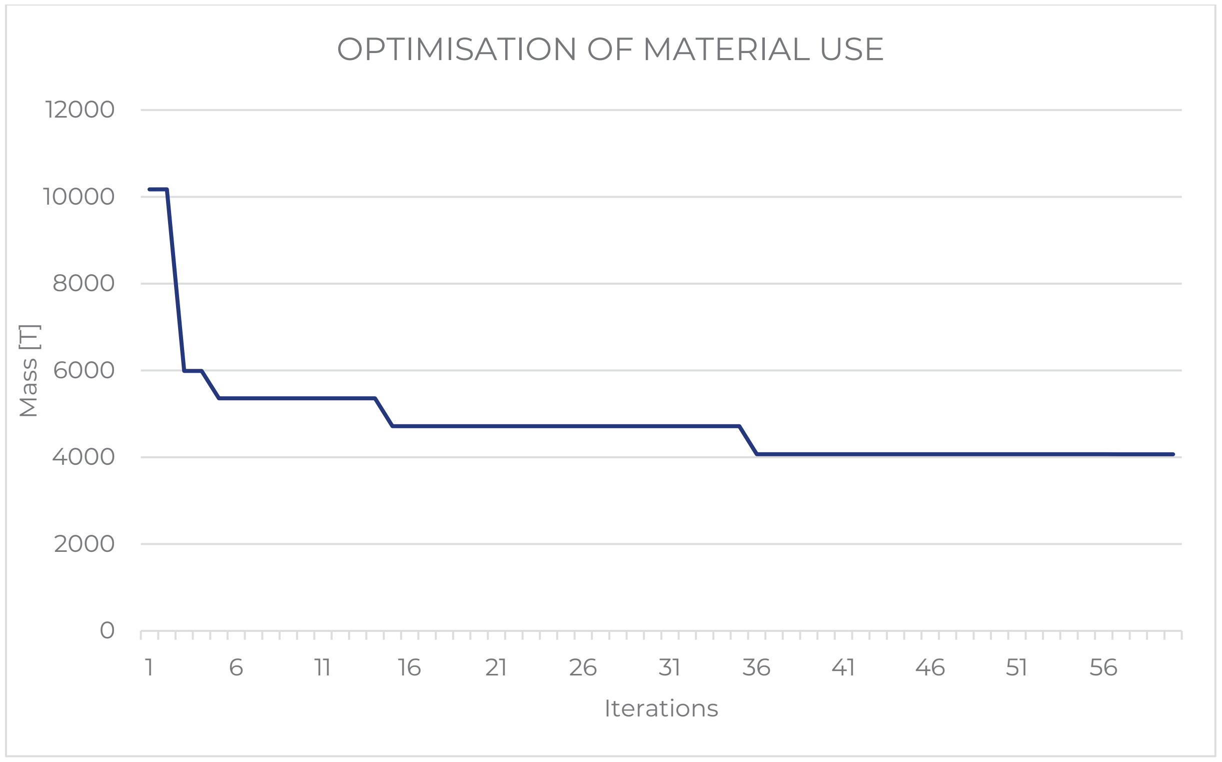


Figure 3-32: Results of Absolute Tower optimisation 2

Properties	**Original**	**Optimised**
Material mass [T]	5984.924	4068.394
Core thickness [m]	0.40	0.30
Deflection [m]	1.544	1.073

Table 3-7: Results of Absolute Tower optimisation 2

Figure 3-32 shows the trending down of the material mass objective. This graph, however, is more stepped compared to the smooth graph of the deflection in the previous optimisation. This is due to material mass not being a continuous variable like deflection but is the property of a fixed list of cross-sections. Nonetheless, the mass of concrete needed in the core was reduced by 32%, a reduction of close to 2000 tons, simply by manipulating the geometry of the building. This was as a result of reduced thickness of 0.3m versus the original 0.4m. The deflection was also reduced to 1.073m which was unexpected. It was concluded that this could be a result of the deflection limit; perhaps the previous smaller thickness, 0.25m, would have put the deflection slightly over the 1.544m limit. A smaller step size for thickness could be used to mitigate this problem.

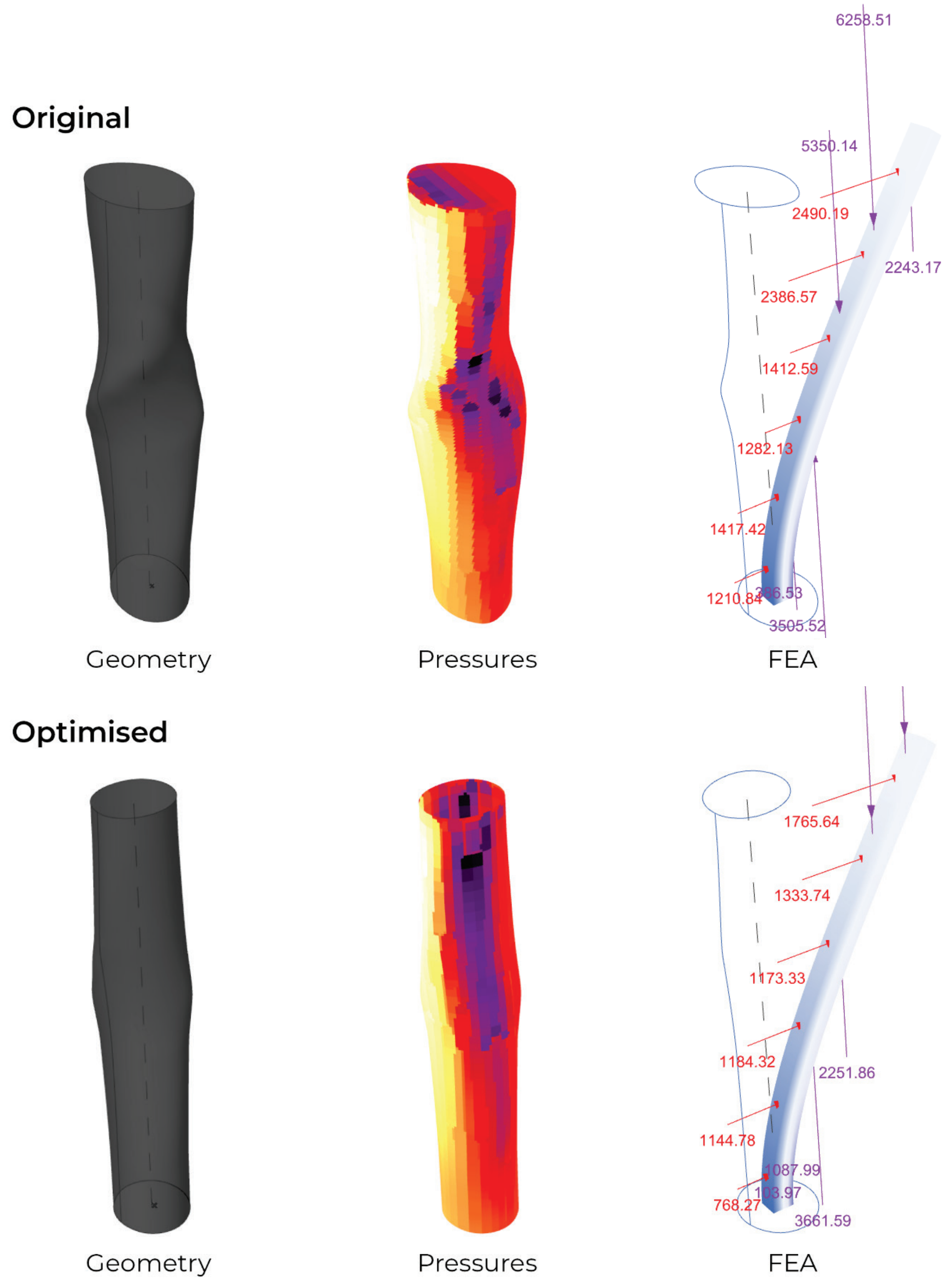


Figure 3-33: Figure 3 26: Absolute Tower optimisation 2

The optimised design, in this case, is different from that in optimisation run 1. The building is now only slightly rotated from the original and the cross-section is a bit wider/more rounded. This, though being the best performing option is quite different from the original design of the building and may not be acceptable to an architect.

Optimisation 3 – Nanchang tower with mass

The Nanchang tower features a gradually changing smooth cross-sectional shape. To parametrise the geometry, it was thought to manipulate this shape along the tower. The model was made in Rhino based on floor plan and section drawings and then referenced into Grasshopper. Contour curves were made all along the height of the building and their control points obtained. Based on a bounding attractor circle for each curve a multiplier with slider was made to control the attraction or repulsion of the control points to the circle. This was split between the top and bottom half of the tower with a slider controlling each. From these control points, the curves were reformed and lofted and capped to create the model to be put into Butterfly (Figure 3-34). Table 3-8 shows the initial values and ranges of the input variables.

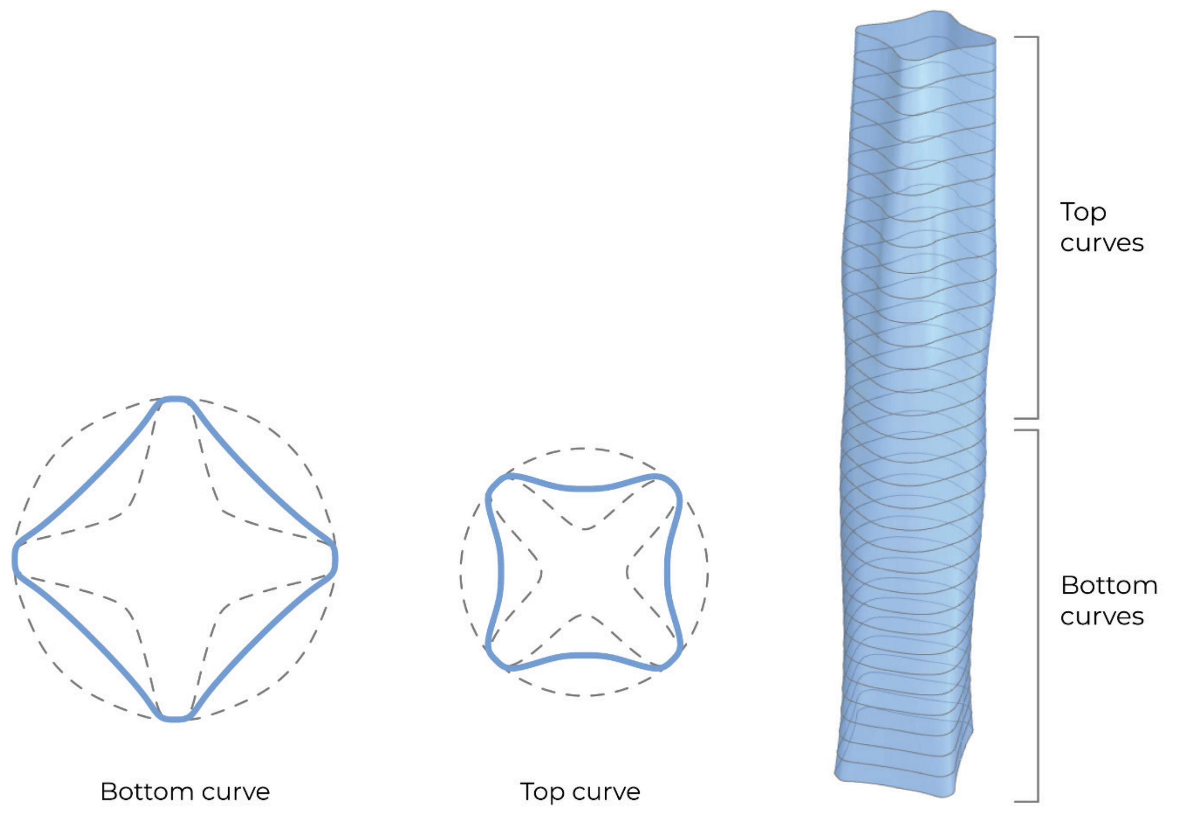


Figure 3-34: Nanchang Tower model parameters

Case study	Nanchang Tower	
Max iterations	60	
Parameters	Top curves	Bot curves
Ranges	-1.000 < x < 1.000	-1.000 < x < 1.000
Objective	Material mass	
Cross-sections	0.20m to 1.15m in 0.05m increments	

Table 3-8: Nanchang Tower optimisation settings

The objective for this optimisation was material mass. As the true core of this tower has a shape that starts as a square and, at near halfway up the height, begins to transition to an octagon it could not be modelled directly for Karamba. Thus, for these tests, a square core was modelled with a moment of inertia equal to the average moment of inertia of the true core geometry. This equates to a cross-section of 19.75m x 19.75m x 1.0m thick. The building is 302.91m tall. An initial FSI analysis was done to establish a baseline deflection of 0.6836m. This was set as the deflection limit. For the list of cross-sections for CSO, the length and width were kept constant at 19.75m but the thicknesses ranged from 0.20m to 1.15m in 0.05m intervals.

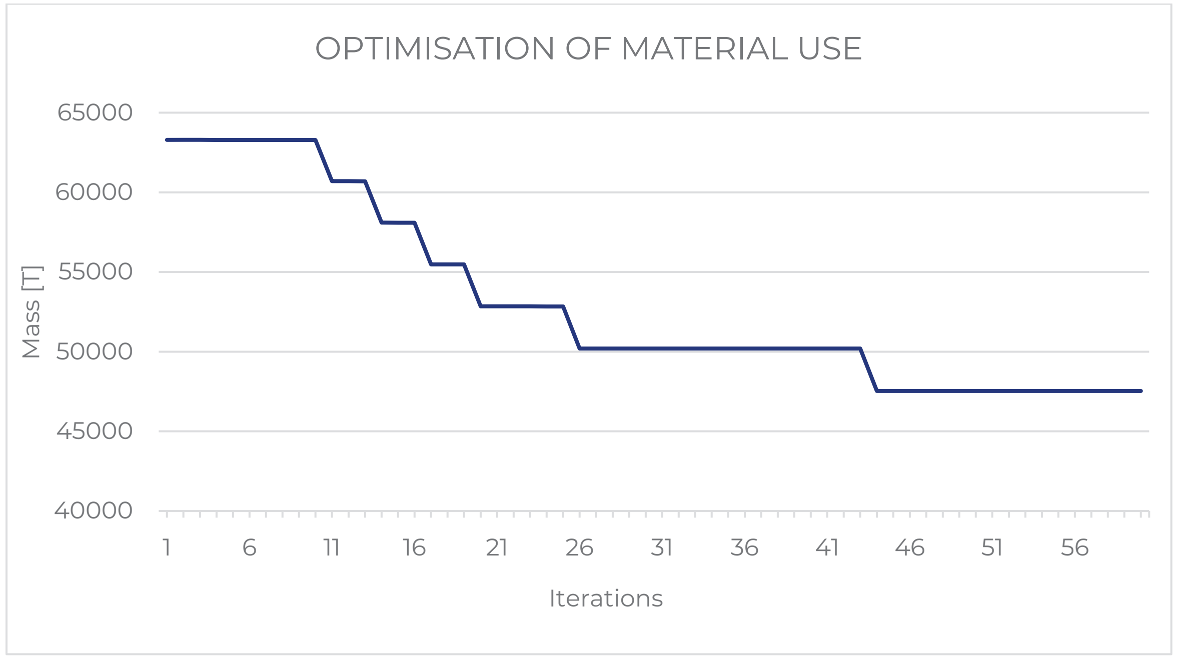


Figure 3-35: Results of Nanchang Tower optimisation

Properties	**Original**	**Optimised**
Material mass [T]	56240.466	47533.016
Core thickness [m]	1.00	0.85
Deflection [m]	0.6836	0.66132

Table 3-9: Results of Nanchang Tower optimisation 2

Table 3-9 shows the core thickness was reduced from 1m to 0.85m corresponding to a reduction in material mass of 15% to 47533 tons. A small decrease in deflection of 0.022m was also observed. As with the results from the second Absolute tower optimisation run this could be due to the step size selected from the thicknesses though it is less of a problem in this model.

Figure 3-36 shows the original and optimised Nanchang Tower model. The optimised version looks more rounded in the top half while the lower half is contracted. The largest point load on the initial model is the one before the top corresponding to the concave shape of the facade at that height. The optimised geometry smooths out this area possibly allowing air to flow easily around it rather than get caught in the concave area and impart higher pressures. The resulting loading is now a gradual increase from bottom to top corresponding to the increase of wind velocity with height. The contraction of the bottom cross-section was unexpected as it would be thought that the most optimal shape would be smoother and rounder. On further examination of the pressure values of the mesh and the resulting point loads it was noticed that while the loads between the original and optimised version near the bottom were more or less equivalent, the decrease in pressure at the top portion between the optimised and original geometry was so great that it is possible that the bottom simply didn't matter so much. Since black-box optimisation methods such as the one used here have no knowledge of the actual subject of the optimisation but only look at the numeric values of the inputs and outputs it is plausible that an unexpected result like this can occur.

As with the Optimisation 2, the optimised version of the geometry is noticeably different from the original design which can be an issue for an architect. The Opossum plugin includes a feature where a text log file is saved during the optimisation run which records for each iteration the variables and objective values. This is very useful as it allows the designer to have a list of potential designs and their resulting material use or other objective. Thus, they have the option of selecting a design that is a balance of performance and aesthetics. In the next optimisation, this was further explored as well as other methods of recording the options of each iteration.

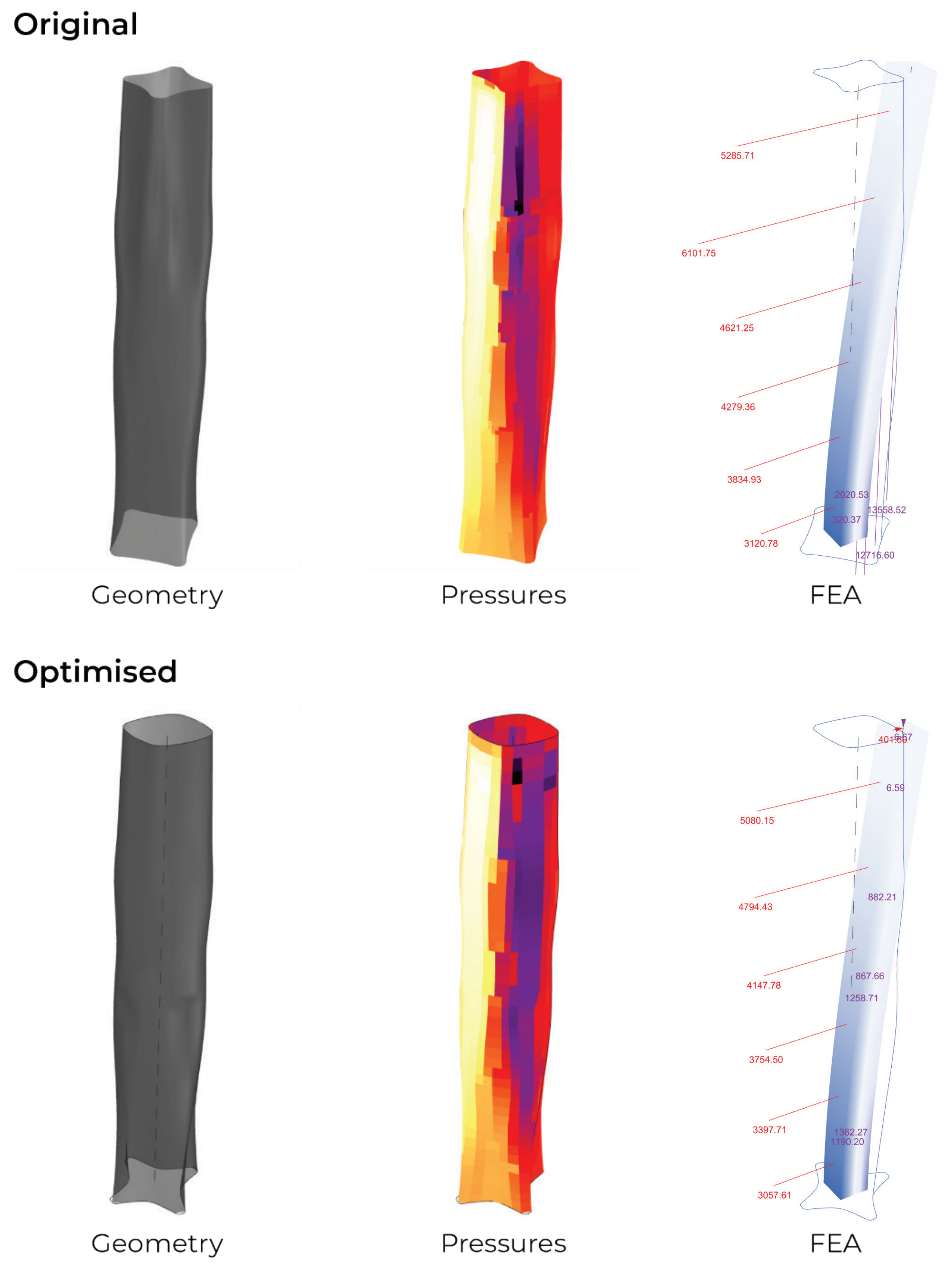


Figure 3-36: Nanchang Tower optimisation

Optimisation 4 – Ardmore Residence with mass

The Ardmore Residence's floor plan shape was built as a curve to be modified by sliders. There were three parameters. Two sliders control the position of each of the wings along the main body. A third slider modifies the edges of these wings from straight to a more angled position. From this curve outline, the massing was extruded to the 136m height of the building. In addition, the building was rotated 45 degrees so the wind would impinge on the building off axis.

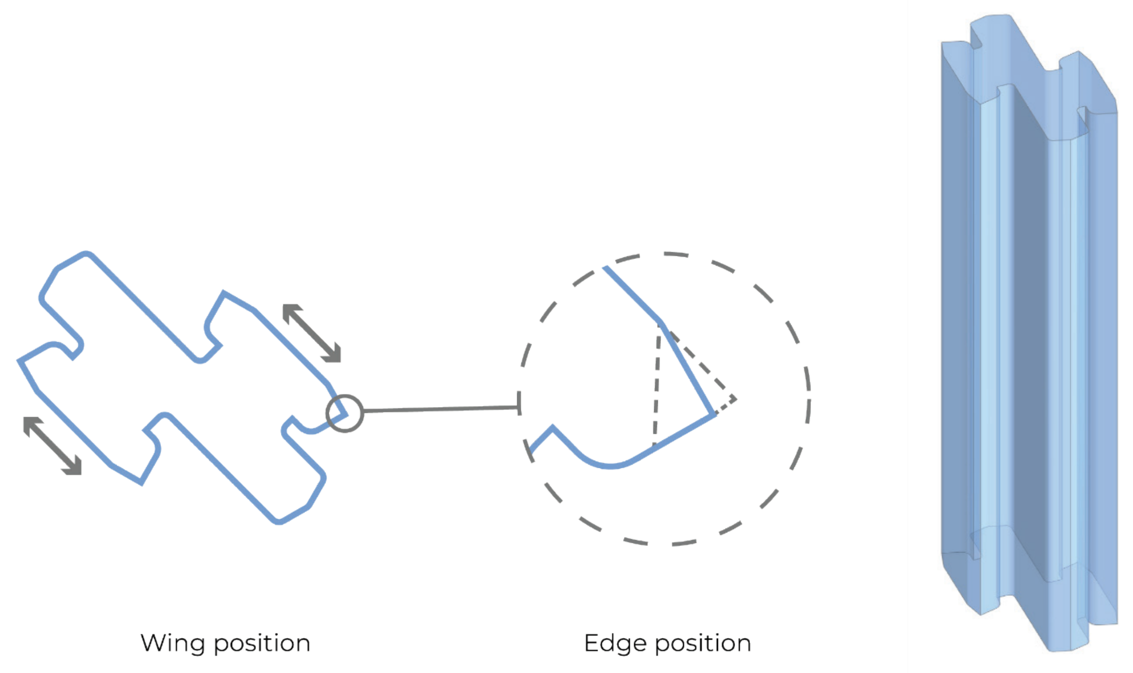


Figure 3-37: Ardmore Residence model parameters

Case study	Ardmore Residence		
Max iterations	100		
Parameters	Top position	Bot position	Edges
Ranges	-15.00 < x < 5.00	-5.00 < x < 15.00	-2.00 < x < 1.00
Objective	Material mass		
Cross-sections	0.10m to 0.59m in 0.01m increments		

Table 3-10: Ardmore Residence optimisation settings

The core of the building measured 11m by 7m and 0.4m thick. The lists of cross-sections range from 0.10m to 0.59m thick in 0.01m intervals in contrasts to the previous optimisation tests in order to have a wider range of possible objectives. The max number of iterations was increased to 100 to allow for more time to reach an optimum owing to the wider objective range. The deflection limit was set at 0.6553m based on an initial FSI run.

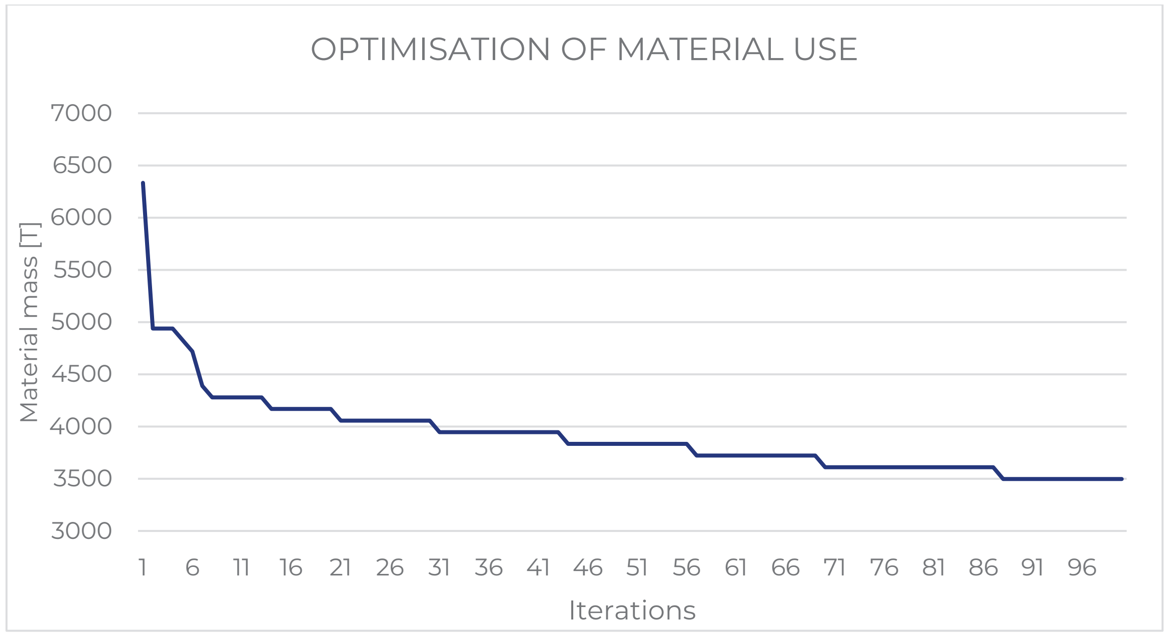


Figure 3-38: Results of Ardmore Residence optimisation

Properties	Original	Optimised
Material mass [T]	4609.59	3497.39
Core thickness [m]	0.40	0.30
Deflection [m]	0.66	0.66

Table 3-11: Results of Ardmore Residence optimisation

Figure 3-38 shows the gradual reduction of the objective over each iteration. It is much smoother than the previous two tests owing to the smaller step size in cross section. A 24% reduction in the material mass was achieved by reducing the core thickness from 0.30m to 0.40m (Table 3-11). Deflection remained the same in contrast to the past two tests. This shows that a small step size in thickness is preferable. Figure 3-40 shows the resulting optimum shape. The lower wing is moved to the front resulting in a more symmetrical cross-section. This was most likely to reduce the large flat wall area on the windward side in the original layout. The edges were also pulled to a sharper angle. The optimised layout performs more like an airfoil allowing wind to flow better around it imparting less pressure (Figure 3-39).

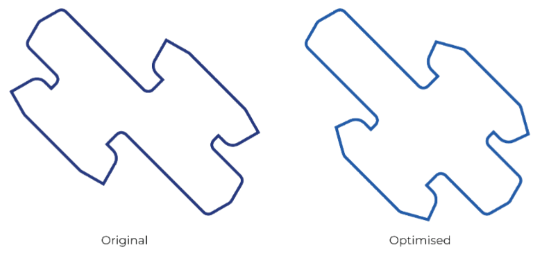


Figure 3-39: Original vs. optimised plan layout

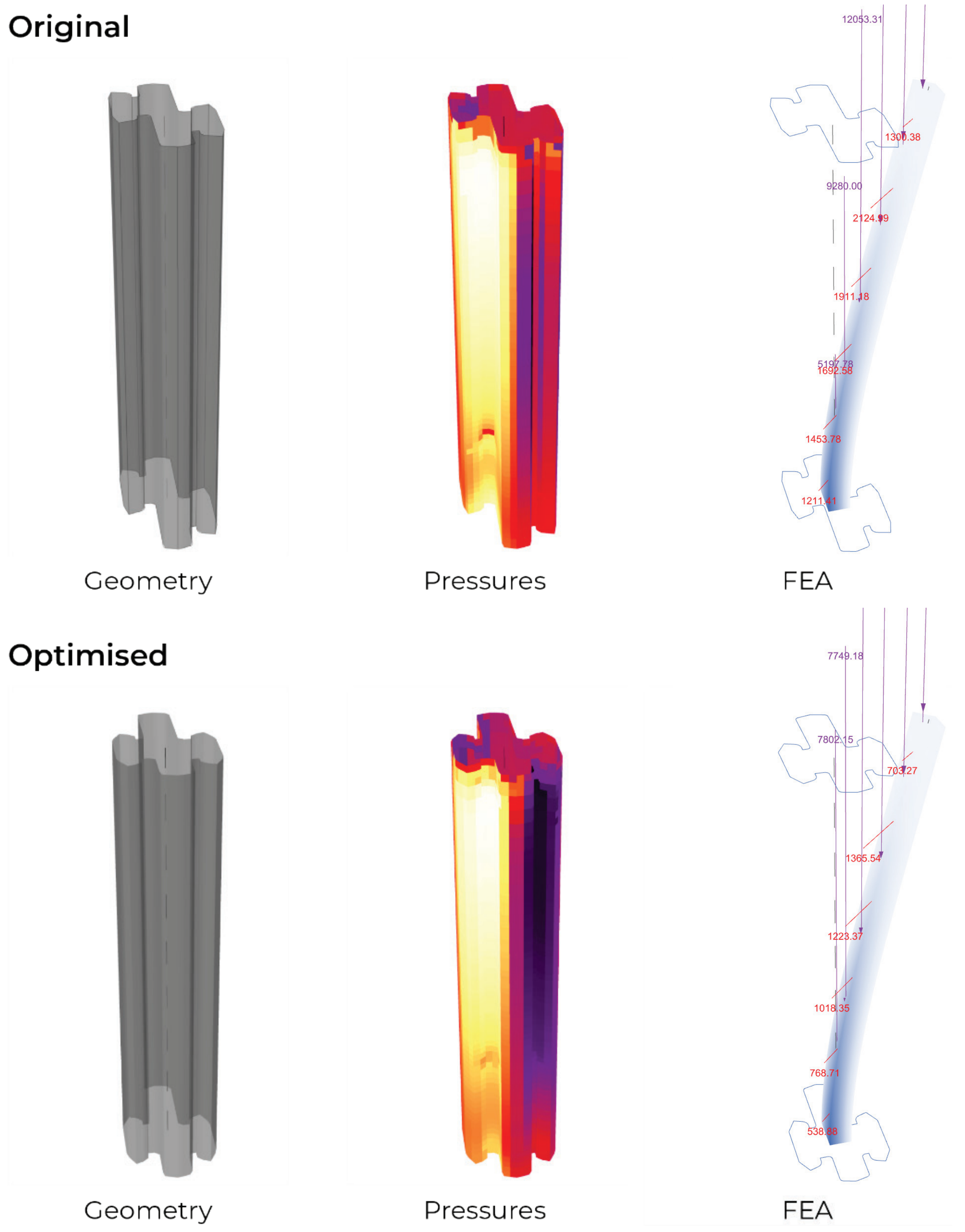


Figure 3-40: Ardmore Residence optimisation

Again, this shape is quite different from the original. However, Opossum is able to produce a log file of each iteration showing the parameters and resulting variables. Table 3-12 shows a portion of this log. From this, an architect/engineer could choose an option that may be of lower performance but closer to the desired design.

Objective	Parameters		
3497.389648	-1.33485	-0.929071	10.44161
3497.389648	-1.31458	-0.939001	9.601758
3497.389648	-1.304247	-0.723498	10.06166
3497.389648	-1.339462	-0.923453	10.42814
3497.389648	-1.336708	-0.888233	10.44881
3497.389648	-1.314301	-0.783084	9.776589
3497.389648	-1.337035	-0.913835	10.44251
3497.389648	-1.297746	-0.760683	9.94928
3497.389648	-1.30454	-0.738041	9.98592
3497.389648	-1.308119	-0.837547	9.784086
3497.389648	-1.285229	-1.288027	9.214127
3497.389648	-1.299032	-1.106098	9.43351
3497.389648	-1.306246	-0.775974	9.83051
3609.815648	-0.49105	-4.586158	5.201521
3609.815648	-0.521421	-4.93481	5.170925
3609.815648	-0.56391	-4.800343	5.513151
3609.815648	-0.510737	-4.983012	5.248821
3609.815648	-0.519789	-4.899149	5.24057
3609.815648	-0.50487	-4.58078	5.22472
3609.815648	-0.497973	-4.571504	5.19698
3609.815648	-1.346463	-0.985015	10.70364
3609.815648	-0.497863	-4.576131	5.228151
3609.815648	-0.564426	-4.473779	5.532446
3609.815648	-0.54981	-4.542832	5.322157
3609.815648	-1.295984	0.159091	8.790278
3609.815648	-1.332086	-0.940997	10.1768
3609.815648	-1.351015	-0.962754	10.36552
3609.815648	-1.328714	-0.983009	10.43439
3609.815648	-1.274754	-0.71672	9.536599
3609.815648	-1.317449	-0.892674	9.659267
3609.815648	-1.293029	-1.047025	9.644604
3721.973648	-0.5	-5	5
3721.973648	-0.197962	-6.041961	6.439614
3721.973648	-0.382408	-5.530504	5.857319
3721.973648	-0.335814	-6.489037	6.152292
3721.973648	-0.36223	-5.833817	6.099619
3721.973648	-0.54279	-4.838407	5.015133

Table 3-12: Part of log produced from Opossum optimisation run

In addition, data recorder components were added in the Grasshopper script connected to the mesh and pressure value outputs so the results at each iteration were recorded in a list. This allowed the user to scroll through resulting geometry and structural results at each iteration after the optimisation run. This is a feature that could be further developed and better integrated into the tool in future work.

Optimisation 5 – Absolute Tower with smaller step size

To confirm the findings regarding cross-section step size discovered in optimisation 4, optimisation 2 was rerun using a different list of cross-sections. This time the list went from 0.10m to 0.59m in 0.01m steps. All other settings remained the same as optimisation 2.

Case study	Absolute Tower		
Max iterations	60		
Variables	Base length	Base width	Twist
Ranges	12.0 < x < 16.0	12.0 < x < 16.0	-1.0 < x < 2.0
Objective	Material mass		
Cross-sections	0.10m to 0.59m in 0.01m increments		

Table 3-13: Absolute Tower optimisation 3 settings

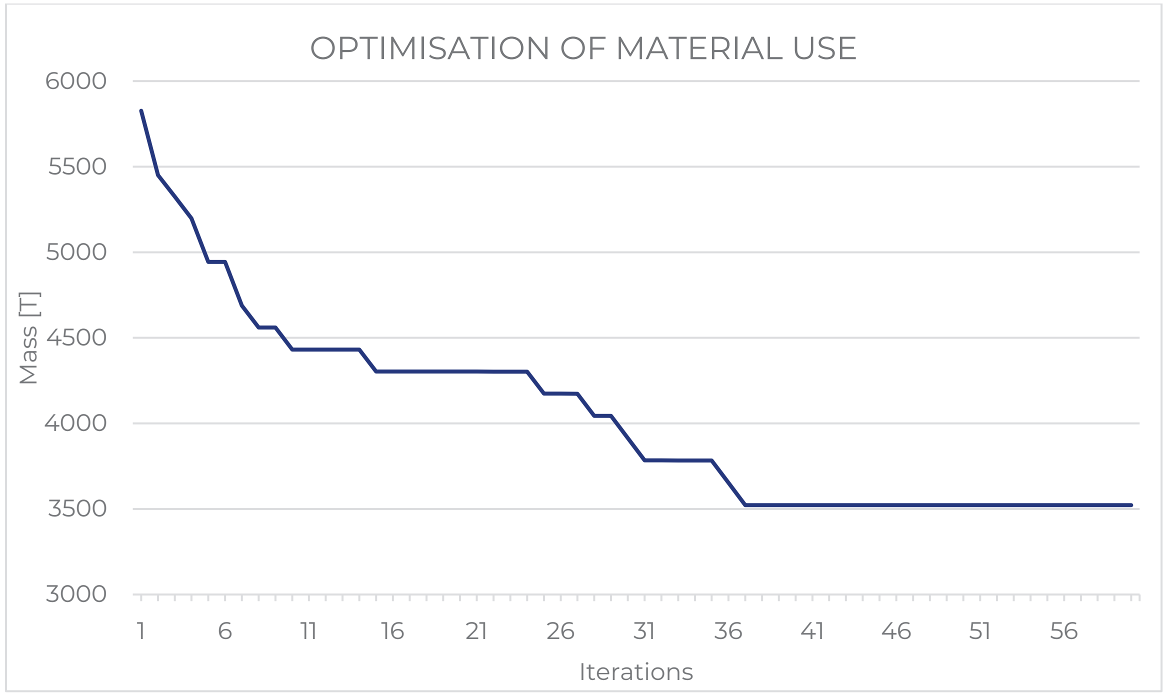


Figure 3-41: Results of Absolute Tower optimisation 3

Properties	**Original**	**Optimised**
Material mass [T]	5984.924	3522.112
Core thickness [m]	0.40	0.26
Deflection [m]	1.544	1.420

Table 3-14: Results of Absolute Tower optimisation 3

The resulting graph is much smoother than that in optimisation 2. It was also able to arrive at an even better-performing objective of 3522 Tons due to a smaller cross-section of 0.26m with a deflection value of 1.4199m which is much closer to the limit than the 1.017m in optimisation 2. This confirms the hypothesis that the 0.05m step size in optimisation 2 was too large and thus missed the optimum gained by selecting a 0.26m cross-section.

3.4.3 Conclusions

The goal of these optimisation tests was, as previously stated, to ascertain the extent to which structural performance based on wind load can be improved by manipulating the external massing of the building and what input variables, objectives, and optimisation settings can produce the most ideal optimisation problem. The computational method being developed should be able to readily accept any geometry that a user puts into it and get good results. The optimisation settings, in particular, should allow for that. The result would be a set of guidelines for performing optimisation runs using this method.

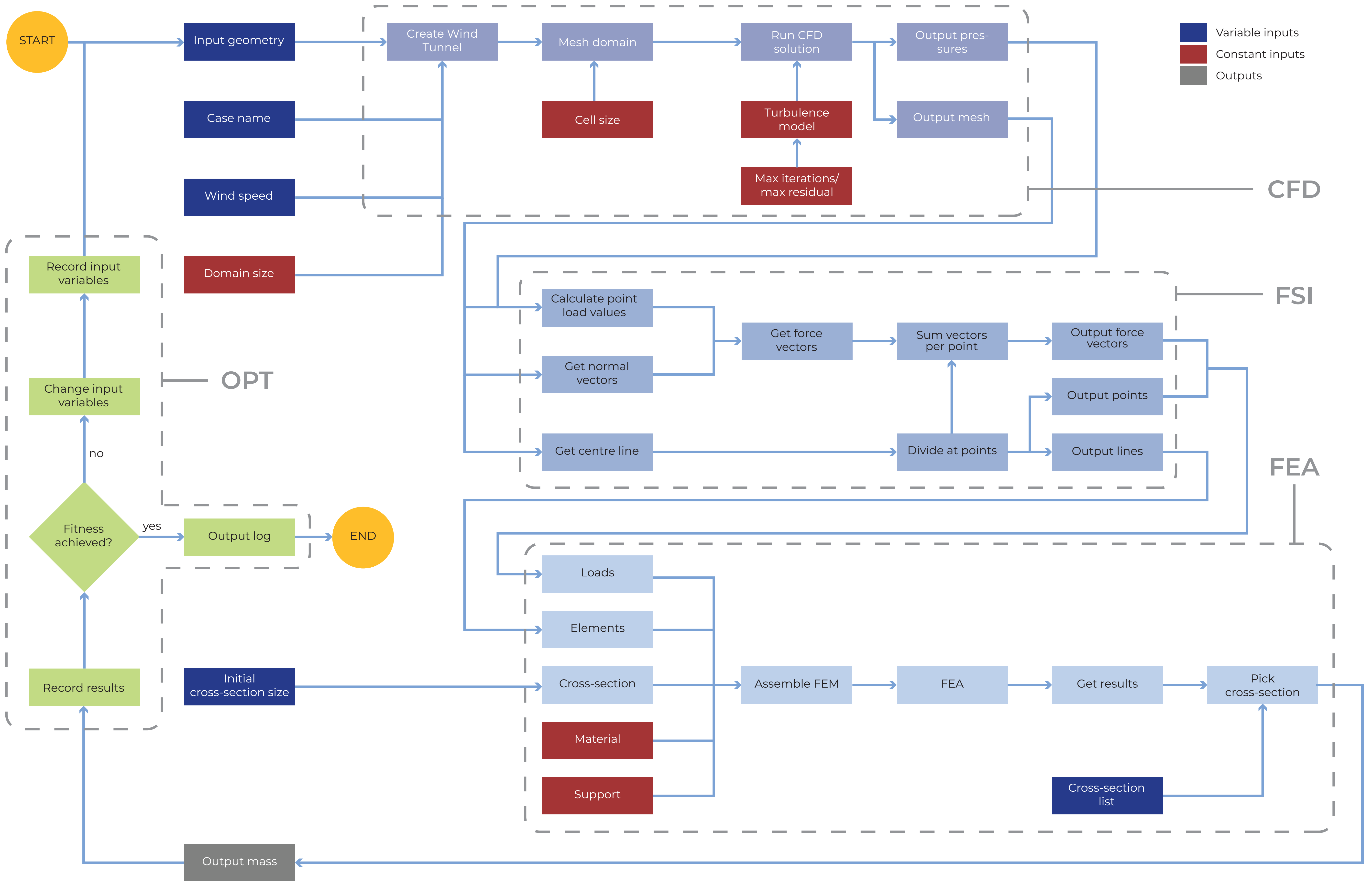
In general, each of the optimisation tests was able to successfully reduce the objective by manipulating the numeric sliders controlling geometry. While it was sought to keep the shape-changing within the general architectural layout of the building there was, in all cases, noticeable difference between the original and optimised geometry. This shows one of the ways building design is complicated with one aspect of performance, in this case, the size of the structural core, competing with another like architectural aesthetics. In this, it is concluded that it is better to have a range of options rather than a single optimal result. Thus, the user can pick an option that balances performance with the desired shape. The Opossum plugin was a good choice in this regard as it can produce a table of input variables and resulting objective value at each iteration. This along with the ability to use data recorders in Grasshopper can allow a useful result selection feature to be added to the developed tool.

The research showed that CFD-O is rarely done due to the computationally expensive and time-consuming nature of it. In these tests, each iteration took a maximum of 2 hours for the Nanchang tower, the largest building tested, and a minimum of 30 minutes for the smallest case the Ardmore residence. Therefore, a full optimisation run took between 1 and 2 days in each case. This was possible due to using low mesh and iteration settings for the CFD and using a model-based optimisation algorithm which works best at a low number of iterations. Though it is still quite a bit of time it would have been much worse using, for example, a genetic algorithm which typically requires hundreds or even thousands of iterations to converge. It can be concluded that CFD-O and particularly in this

case FSIO is feasible and reasonably efficient using this method. Opossum with the RBFOpt algorithm looks to be able to produce optimised results within a relatively small number of iterations. For optimisation 1 & 2 the objective seems to converge before the max iterations of 60. In number 3 with the Nanchang Tower, the final level of the graph is steady for about 15 iterations. This may mean that optimisation 3 could have run for some more iterations, but it is uncertain whether this additional time would have produced a fitter result. In optimisation 4 the max iterations were increased to 100 however, there was not a long period of static results like in the previous cases so a slightly longer run may have been better to confirm convergence. Nevertheless, it did come to much improved objective value in the time given. It looks that between 60 – 120 iterations with the RBFOpt algorithm is ideal.

Material mass seems to be a more meaningful objective than just deflection as it makes the tool more applicable in practice. The combination of Opossum with Karamba CSO is very useful in this case especially if one already has selected an initial core size. It can show as a result of changing the geometry of the building you can reduce core size by the found amount and save this amount of material. However, if it is even earlier in the design phase, deflection can still be a good objective as it shows what shape would generally perform better. The step size of 0.05m was too large as deduced in optimisation 2 and 3. A reduced step size of 0.01m performed better as shown in optimisation 4 and 5.

3.5 Final Method and Tool



The final developed method is shown in Figure 3-42. The variable inputs are the ones which change with each project, the constant inputs remain the same between projects as there is no need to modify them, however, a more advanced user could alter them if desired. The FSIO method takes a 3D model of an object performs CFD analysis to determine the wind pressure imparted on it and passing those results to FEA solver to obtain structural results. This is then paired with an optimisation algorithm in order to generate better performing options and output a list of results. This thesis developed this method within Rhino/Grasshopper using existing plugins such as Butterfly for CFD, Karamba for FEA, and Opossum for optimisation. This was combined with own custom scripting with GH components, Python, and C# code to obtain the tool based on the method (Figure 3-43). This custom scripting allowed the combination of the existing plugins as well as allowing the procedure to be as parametric and adaptable as possible so that precise and timely results can be obtained regardless of the building model input by a user. See Appendix 6 for Python and C# code.

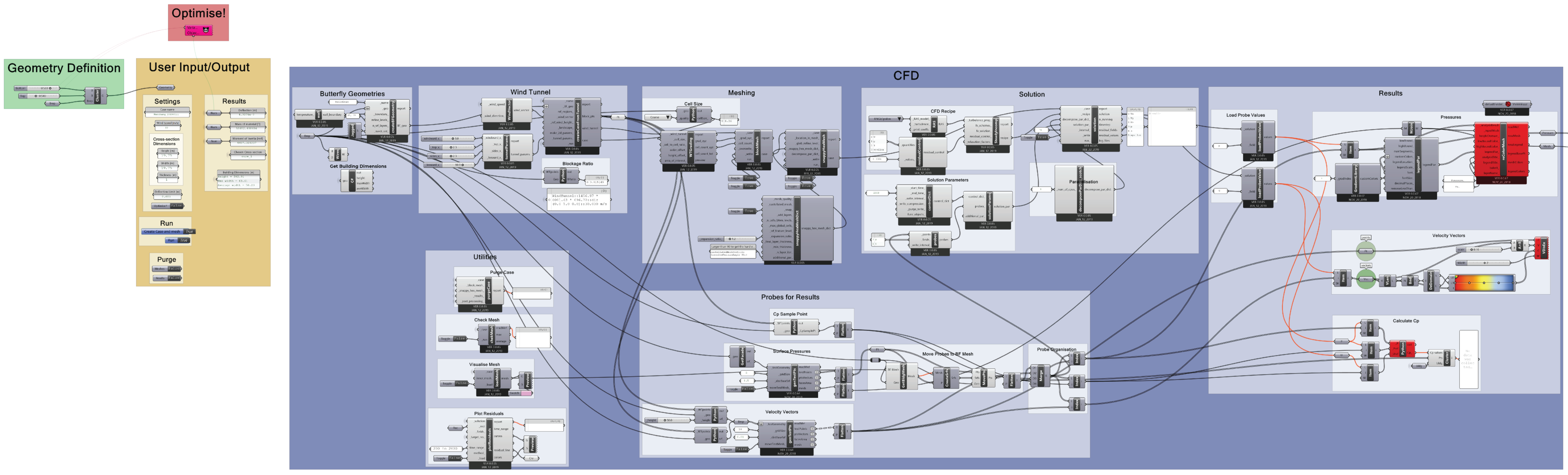


Figure 3-43: Full FSIO script

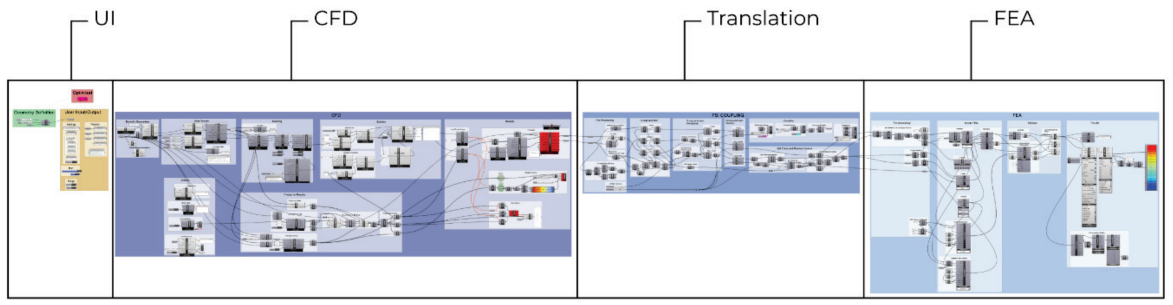
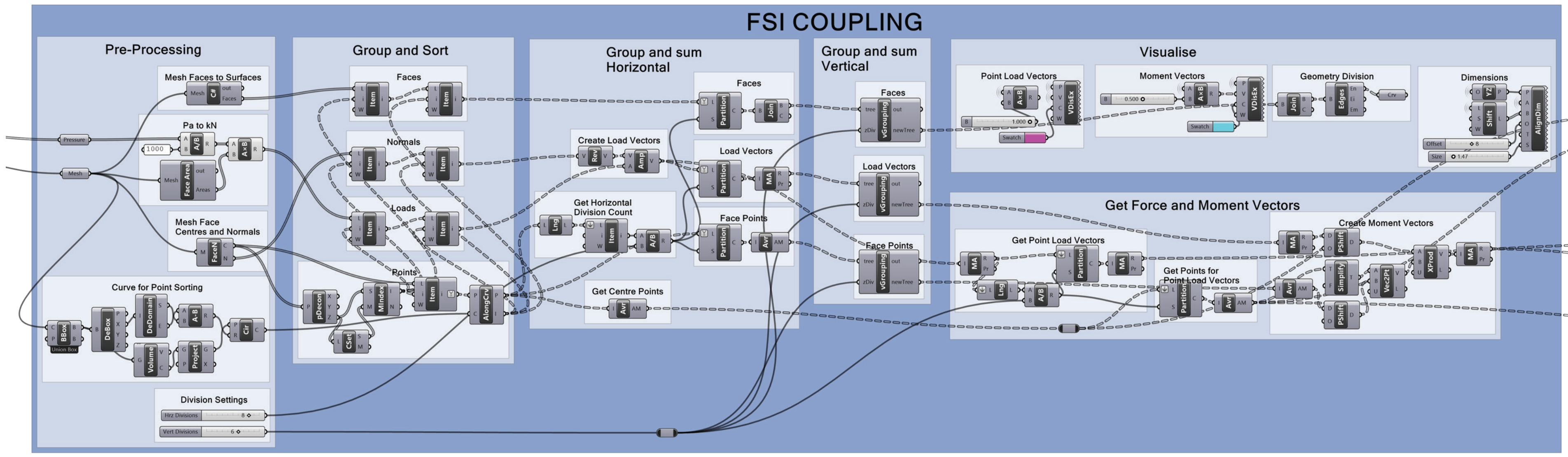
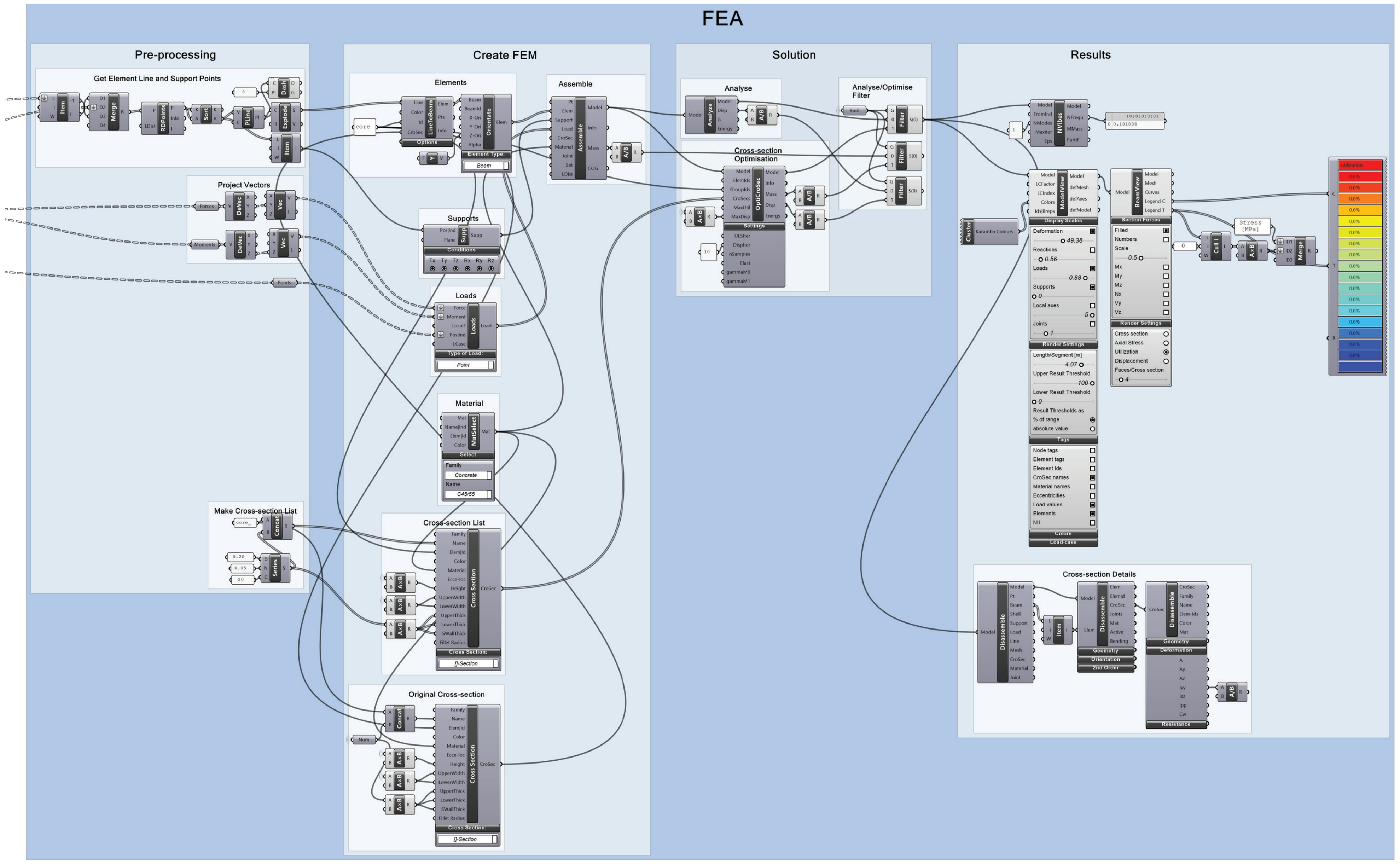


Figure 3-44: Total FSIO script overview


FSI COUPLING
Pre-Processing
Mesh Faces to Surfaces
Pa to kN
Mesh Face Centres and Normals
Curve for Point Sorting
Division Settings
Group and Sort
Faces
Normals
Loads
Points
Group and sum Horizontal
Create Load Vectors
Get Horizontal Division Count
Load Vectors
Face Points
Get Centre Points
Group and sum Vertical
Visualise
Point Load Vectors
Moment Vectors
Geometry Division
Dimensions
Get Force and Moment Vectors
Get Point Load Vectors
Get Points for Point Load Vectors
Create Moment Vectors


FEA
Pre-processing
Get Element Line and Support Points
Project Vectors
Make Cross-section List
Create FEM
Elements
Assemble
Supports
Loads
Material
Cross-section List
Original Cross-section
Solution
Analyse
Cross-section Optimisation
Analyse/Optimise Filter
Results
Cross-section Details

4

VERIFY

As mentioned in the Research section, the current methods for evaluating wind loading on buildings are the Eurocode EN1991-1-4 procedure with corresponding National Annex and scaled boundary layer wind tunnel tests particularly when the building is of complex geometry. As a complement to the CFD validation study (section 3.2.2) which validated the CFD procedure's accuracy against wind tunnel tests, this section will seek to compare the FSI tool against the Eurocode calculation method. The aim was to see how similar (or different) the values obtained from EN methods are to those obtained from the FSI procedure. The Absolute Tower and Nanchang Tower models were used in this test.

4.1 Eurocode Calculations

In EN1991-1-4 many of the values given or derived from graphs are provided for standard cross-sectional shapes like rectangles or circles. For these calculations, it was chosen to assume values for a circular cross-section as these building models have a smooth cross-sectional shape which is imagined having airflow closer to a circle rather than a rectangle with sharp corners causing flow separation. Basic wind velocity was taken as 30m/s and roughness length of 1m to match the CFD simulations. A_{ref} was set at 1m^2 in order to obtain wind force, F_w, per area. In the FSI script, the point loads were obtained at 6 points along the height of the building. These same 6 heights were used to obtain F_w from the EN. The areas to which the F_w would be applied was obtained in Grasshopper by finding the areas around the point loads of the building perpendicular to the wind flow (Figure 4-1). This was done for each of the case study buildings and the loads compared to those from the FSI procedure. See Appendix 4 for the calculation procedure.

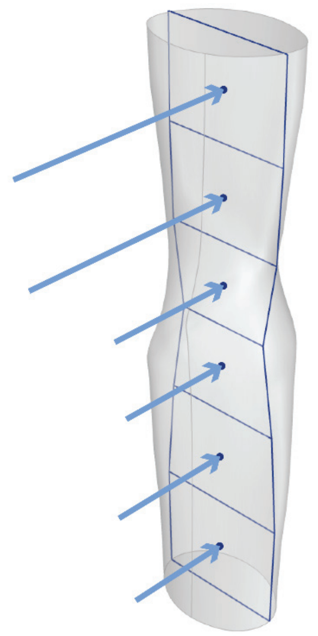


Figure 4-1: Areas for wind force, F_w, application

4.2 Results

The results for the Absolute Tower model are shown in Figure 4-2 and Table 4-1. The EN numbers begin to rise then fall with respect to the decrease in the perpendicular area near the middle of the tower then rise again to a maximum value of 1255.92kN. The values from the FSI tool follow a similar pattern but are much higher. The values as well vary according to the height which accounts for the change in geometry of the tower. For instance, near the midpoint of the tower, there is a dip in wind force corresponding to the location of the twist. This area has lower wind pressure due to the long axis of the elliptical cross-section facing the wind which is a more aerodynamic arrangement. The higher forces at the top and bottom are possibly due to wind impacting the ellipse along the short axis which is a flatter area than the perceived circle of the EN calculation.

However, the discrepancy in the magnitude of the loads between the FSI procedure and the EN is quite large. This was thought to be so as CFD with RANS turbulence models calculates the mean static pressures. In reality, wind flow in the boundary layer is more random and peak pressures do not occur simultaneously over a structure (Cook, 2007). This is accounted for in the EN by the structural factor, c_sc_d, and the force coefficient, c_f, which are multiplied to the peak velocity pressure $q_p(z)$ (Equation 2-14). To account for this the FSI values were multiplied by the by c_sc_d and c_f (FSI Reduced). As seen in Figure 4-2, the reduced values from GH are now closer in line with those from EN calculations.

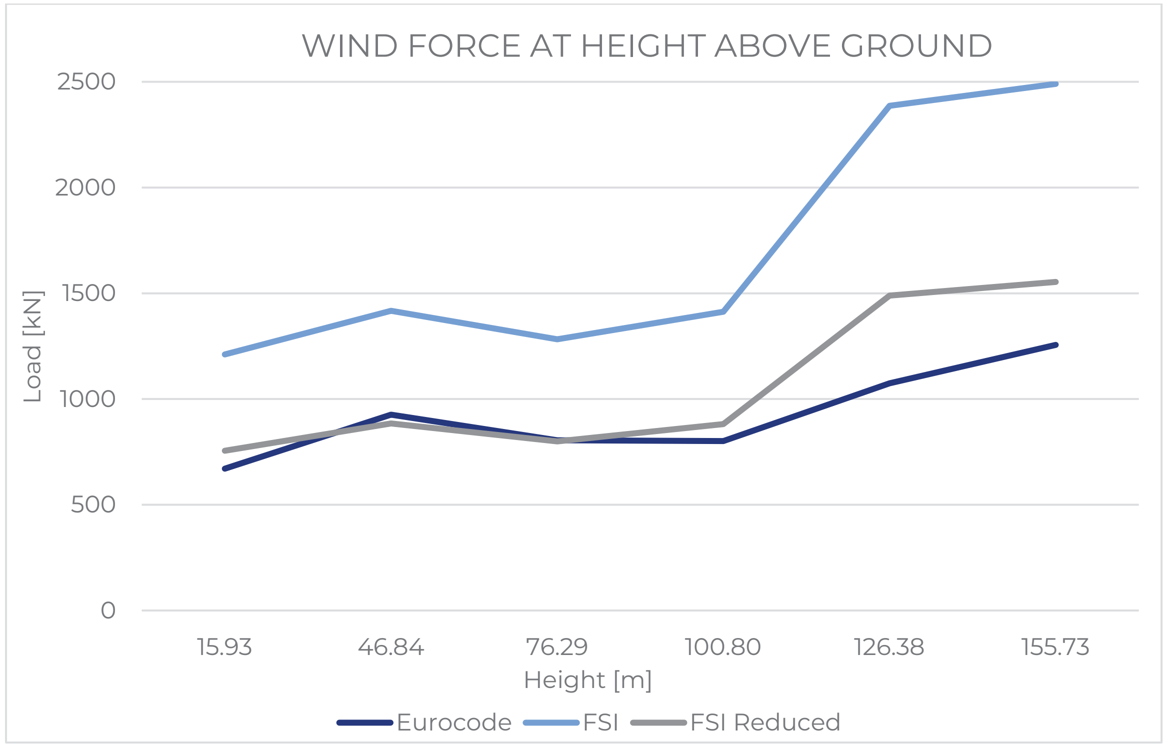


Figure 4-2: Absolute Tower EN/FSI calculation comparison

Heights [m]	Areas [m²]	Wind Force [N/m²]	EN Point Load [kN]	FSI Point Load [kN]	FSI reduced [kN]
15.93	1232.57	544.10	670.64	1210.84	755.47
46.84	1148.51	806.31	926.05	1417.42	884.37
76.29	867.27	927.85	804.70	1282.13	799.95
100.80	803.19	997.78	801.41	1412.59	881.35
126.38	1018.82	1054.79	1074.64	2386.57	1489.04
155.73	1133.90	1107.61	1255.92	2490.19	1553.69

Table 4-1: Absolute Tower EN/FSI calculation comparison

The calculations were as well carried out for the Nanchang Tower model. The EN values follow a smooth curve with a peak at a height of 178.44m. The GH values are much higher than EN. They smoothly increase until a height of 178.44m then jump at 229.46m. This is likely due to the concave façade at this point which leads to a higher pressure as the air would have difficulty flowing around the building at this point. The value then drops back down at the highest point where the wind can then flow over the top of the building (Figure 4-3). To mitigate the problem of the high values from the FSI procedure, the values were again multiplied by the factors $c_s c_d$ and c_f for the Nanchang building. This gives values that closely follow the EN curve of values except for the deviations discussed earlier.

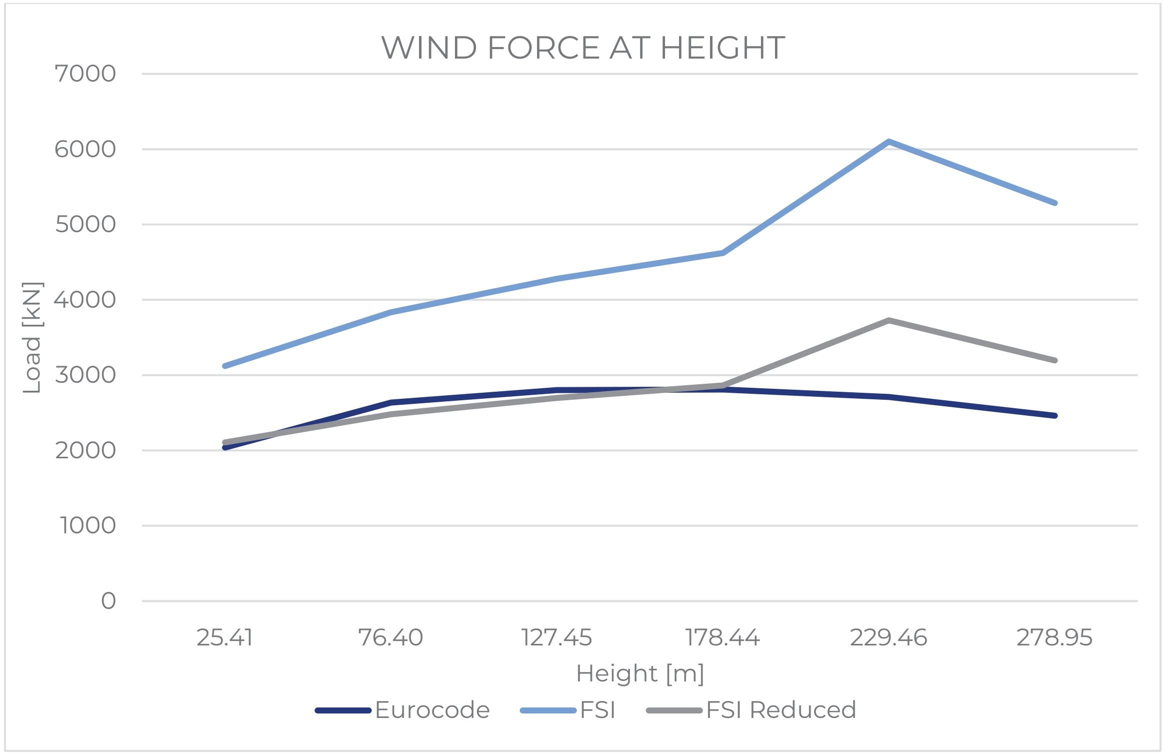


Figure 4-3: Nanchang Tower EN/FSI calculation comparison

Heights [m]	Areas [m²]	Wind Force [N/m²]	EN Point Load [kN]	FSI Point Load [kN]	FSI reduced [kN]
25.41	3013.03	676.56	2038.49	3120.78	2106.78
76.40	2738.92	962.12	2635.16	3834.93	2479.81
127.45	2555.49	1096.21	2801.35	4279.36	2698.20
178.44	2371.95	1184.31	2809.12	4621.25	2862.45
229.46	2168.35	1250.04	2710.52	6101.75	3727.84
278.95	1892.52	1301.02	2462.21	5285.71	3194.20

Table 4-2: Nanchang Tower EN/FSI calculation comparison

4.3 Conclusions

These results show that the FSI procedure does indeed give values similar to the Eurocode procedure if the structural factor and force coefficient are taken into account. The FSI tool, however, has the added benefit of being able to capture local effects of geometry on wind pressure along the building height as shown in the graphs above for the two building models. Moreover, performing optimisation one could see the benefits in the results whereas with the EN calculations they would remain mostly the same as the only variables used relating to building geometry are overall building width and height. The values obtained from the FSI procedure, however, were expected to be appreciably lower than the EN as Cook (2007) stated:

> The simplification in the [EN1991-1-4] model inevitably involves a degree of conservatism to ensure that the most onerous loading case is included. For this reason, design assisted by testing and measurement, as permitted by clause 1.5, often results in lower design loads and a more efficient structure.

These simulations were run with a coarse mesh setting and only 2000 iterations in order to save time. A finer mesh with a higher number of iterations may give better results. Nonetheless, the similarity in results to the EN with the deviations based on geometry prove that this tool can be used, at least in earlier stages, as a complement to the EN.

5 CONCLUSIONS

The objective of this study was to create a computational method that designers and engineers could use in the early stage of design to analyse the structural responses due to the wind and optimise the geometry to reduce it. The project followed a three-step process of research, develop, and verify with each integral to the success of the tool. Research was done to further the knowledge needed in the areas of wind and its action on structures as well on CFD and pairing it with FEA to obtain structural objectives. Finally, research on the science and process of optimisation was done. Also researched was current calculation methods. This all helped establish the current state of the art and how the presently available processes and tools could be used and improved upon. As a result, development of this method could take place by creating a proof-of-concept tool in Grasshopper using available plugins coupled together with own custom scripts in a way that it was made as parametric as possible. The verification and validation studies, as well as the tests throughout the development, helped show that it can be a useful tool for design. This will be further explained as answers to the initial research questions. However, there were limitations to the development as well as to further use of the tool. There is also much room for improvement to make this better that could not be done in the scope of this thesis. The recommendations given in Table 5-1 were compiled as a result of the thesis. It is believed that following these steps will result in a successful FSIO method not just in Grasshopper but other software packages as well.

Parameter	Recommendation
Input variables	Select features that mostly maintain the architectural intent. Wide enough range of values to ensure optimum is found.
Objective	Material mass is useful in practice. If using a method similar to Karamba CSO be sure the list of cross-sections is thorough enough to ensure a wide range of possible solutions.
Optimisation algorithm	Model-based algorithm allows for convergence within a smaller number of iterations than metaheuristics and is more robust than direct search.
No. of iterations	60 – 120 in a model-based algorithm depending on the complexity of the building model.
Results	List of results at each iteration is preferable to a single optimum allowing the user to choose a result that balances with other objectives of building design.
CFD settings	A coarse mesh with a low number of iterations is preferable in order to save time on large building models. Be sure results are precise and that the solution can reasonably converge within a selected number of CFD iterations. RANS turbulence models such as RNG $k-\varepsilon$ provide a good balance between accuracy and time.

Table 5-1: Recommendations for the setup of a computational FSIO method

5.1 Answers to Research Questions

This section will seek to answer the research questions established at the beginning of the thesis.

Main question:

How can computational methods be used to accurately and efficiently calculate wind load on a complex geometry building and optimise the geometry to reduce wind responses in the early design phase?

The product of the thesis, namely the FSIO tool, was developed using computational methods in order to solve the issue of easy calculation of the wind loading on more complex geometry buildings at an early design stage as posed in the question. Based on research, this did not yet exist at least not in an easy to use and widely applicable way. The parametric and single environment nature of its development makes it an efficient process compared to traditional methods. The verification step in section 4 showed it's comparativeness to the Eurocode and the CFD validation in section 3.2.2 shows comparativeness to scaled boundary layer wind tunnel tests thus establishing the level of accuracy of the tool. In the optimisation tests, each one was able to reduce wind-induced responses relative to the original geometry solely by manipulating the external massing of the building. This is key for an early design phase as detailed structural plans are not made yet and geometry is continuously being changed. Using this tool is another method of implementing performance-based design where design decisions are made not only on aesthetics but how they contribute to certain performance objectives, in this case, structural response but also sustainability by reducing material use.

Sub-questions:

What are the existing methods for wind load analysis and how do they consider complex geometry buildings? Where do they fall short?

As explored in the research, the two main methods for calculating wind-induced responses are the relevant building codes, in this case, EN1991-1-4 plus national annex, and boundary layer wind tunnel tests. The Eurocode reduces the highly dynamic nature of wind loading to equivalent static functions in an analytical process. However, upwards of 20 equations need to be solved and it can be a complicated process to choose the correct equations and values for the specific case. The main issue, however, is that it does not consider the specific geometry of the building being calculated. Buildings these days are complex and will continue to be designed that way as technology improves. The Eurocode still, however, only gives guidance for standard shapes with the only variables based on geometry used are the height and width of the building. The verification study (Section 4.2) showed that the FSI analysis was able to capture the unique loadings induced by the geometry of the building being analysed while the EN values were more general. EN1991-1-4 is aware of its shortcomings in that area which is why clause 1.5 allows the use of physical wind tunnel tests. However, performing these tests is a highly specialised field requiring experts who can appropriately set up the scaled model and the wind tunnel. It is also a time consuming and very expensive process making it impractical for most projects to do repeated tests in an early design phase and even then, it would be a trial and error process as to what shape is better.

What kinds of geometries are more suitable for dealing with high wind loads? What geometries should be avoided?

The shape of a building, particularly a high-rise, can have a big effect on the level of wind-induced response. The strategies given in section 2.2.4 can help reduce the static and dynamic loads due to the wind. Tactics like softening corners, tapering, and varying cross-sectional shape can help particularly with delaying flow separation and thus vortex shedding. This is an advantage for more complex geometry buildings. The sharp corners of traditional rectangular shaped towers induce flow separation that can have negative dynamics effects on the tower. In the optimisation tests (Section 3.4) the geometries that arose as a result of the optimisation algorithm were usually more rounded and smoother. In the case of the Absolute Tower, the wind load was reduced by twisting the tower so that the long axis of the cross-section was parallel to the wind in addition, to expanding the short axis a bit. The Nanchang tower optimisation produced a more rounded tower at the top as opposed to the concave geometry of the original design. In general, the shape should allow wind to flow smoothly around it as easily as possible, however, with the increasing complexity of building shapes the rule of thumb design strategies may not always be the best or only option. For example, in the Nanchang Tower optimisation, it was expected that the bottom portion should be rounded and smoothed out for better performance, but it was shown that the sharper contracted plan shape also produced optimal results. The advantage of the developed FSIO method is that you also have exact numerical results for many different options so you know exactly how well they perform relative to others so a user can choose what is preferred. While these rule-of-thumb strategies may be a good starting point it is more valuable to have numerical indicators of performance especially in situations where wind reactions are critical such as supertall towers.

What responses (deflections, vibrations, reaction forces) do building structures give to wind loading?

Wind flow induces pressure on a building surface. This occurs as positive pressure (pushing the building) at the front where the flow impinges the façade and also as negative pressure (pulling the building) on the rear due to drag. These forces cause deflection in the direction of flow but also base overturning moments and thus stresses in the structure. The other effect is in the crosswind direction perpendicular to the air flow which causes dynamic swaying motions that can be uncomfortable for building occupants. The higher the building the more pronounced these effects can be as seen by the analyses of the 135m Ardmore residence versus the 170m high Absolute Tower versus the 303m high Nanchang Tower. The magnitude of wind forces is much greater in the Nanchang Tower. At such heights, lateral stability trumps vertical stability as the chief structural problem. Attention must be paid to add stiffness to the building structure to reduce such horizontal motion but geometry can play an even bigger role to avoid motions in the first place.

How can Computational Fluid Dynamics (CFD) be used to analyse the effects of wind on a building? How efficient is it compared to current calculation methods? How accurate is it compared to current calculation methods?

CFD simulates the wind and its impact on an obstruction. This includes the pressure induced on the building surface. Thus, as seen in the many analyses run in this thesis, it can be used to determine the effect of wind on a structure in a computational method rather than having to simulate in a physical wind tunnel test. While it is a complicated field and takes knowledge to set up the simulation properly with regards to parameters like meshing, turbulence model, etc. it has the benefit of being able to do this within a computer program making it versatile especially as a parametric tool like Butterfly within Grasshopper. CFD is still, however, a time-consuming process with a high computational cost. For the optimisation tests, the time for each iteration was about 1.5 to 2 hours which was only achieved by using a coarse mesh setting with 2000 max iterations on a fairly powerful computer. Even with this time reduction over the previous CFD simulations in section 3.2.2 and 3.2.3, each optimisation took 1 – 2 days to run. While it may not seem efficient, it is when compared to the cost and total time for

setting up and performing physical wind tunnel tests with the added benefit of being able to intelligently optimise using a computational algorithm instead of relying solely on educated guessing and trial and error to improve performance after a wind tunnel experiment. These benefits also apply over the Eurocode calculations with the added advantage of greater individualised accuracy for the geometry.

The validation study (Section 3.2.2) showed how CFD compares to a wind tunnel test. There is some discrepancy in values obtained with the parameters of mesh size and turbulence model playing the largest role in determining the level of accuracy. A finer mesh can make it much more accurate, but this must be weighed against the time it takes to complete the simulation. For this thesis, a balance had to be struck between these two objectives. It has been determined that for early-stage design the absolute accuracy is less important compared to the precision of results and time taken. The precision of the Butterfly simulations as shown by the results of the mesh size sensitivity analysis (Figure 3-16) allows it to be used confidently in a repetitive application like optimisation where, at least for early-stage design, relative improvements in objective fitness is deemed more important than absolute accuracy to guide geometric strategies for reducing wind-induced responses. It is, however, quite accurate when compared to EN calculations when the relevant factors are applied as seen in section 4. One can conclude that it is even more accurate when looking at the specific geometries of the buildings as the results showed the effects of each building's geometry on the induced wind load.

How can CFD, structural analysis, and optimisation be incorporated into a single, easy to use and efficient, computational process?

The development portion of the thesis focused on answering this question. The result was a computational method that could simulate wind effects on the structure of a parametrically defined geometry and optimise said geometry to reduce those effects. This shows the power of computational processes today and how existing tools can be made more useful by combining them. This is already done in other industries so it logical that the building industry should also step forward and use the technological power available to improve current workflow.

In this thesis, the method was implemented in Grasshopper. This was successful not only because of the host of plugins available but also the visual scripting method and parametric nature of the tool. This allowed the widely different plugins of Butterfly, Karamba, and Opossum to be combined in a single procedure. Moreover, the parametric nature makes it so that it is versatile so that almost any complex high-rise geometry could be input. Much scripting, in Grasshopper and with custom C# and Python script components, had to be implemented in order to make this as flexible as possible. Settings like mesh size were parametrised and the usability of the whole tool was improved to the point where only two toggles are required for operation. While it is still not perfect with errors arising particularly because of Butterfly meshing it is able to be easily used by someone with some knowledge of Rhinoceros and Grasshopper.

Its implementation within Rhino/Grasshopper, a tool already widely used in the building industry, further adds to its value and versatility. However, by following the steps and guidelines detailed here this method could be applied in other software as well. Grasshopper, while incredibly useful and easy to use, does have its limitations. Using other software for the CFD, FEA, and optimisation components and combining them together into this method may present opportunities for integrating functionality or performance improvements that are not achievable in Grasshopper.

How can having wind load analysis in an early design phase improve building performance?

The optimisation runs (Section 3.4) showed the reduction in needed structural material as a result of knowing the wind-induced responses on the structure and optimising the geometry to lessen the impact. In a building project, such a reduction in material saves cost but also lowers the carbon footprint of the project which are valuable objectives to achieve. The FSI analysis of a building in an early stage can also tell if the wind-induced deflection or stresses are too great and thus the geometry can be manipulated to improve it. This is more efficient than waiting until a design is in a more finalised stage and then do the analysis using Eurocode or wind tunnel methods and having to increase the number or size of structural elements to resist reactions. This, however, creates the situation where the optimised geometry of the building may have strayed too far from the original architectural intent of the building which a designer and/or client may not accept. It is useful to have options rather than a single optimum. Thus, the method is useful in producing a list of input variables and their resulting performance from which the stakeholders can choose from. The sub-optimal solutions are still valuable as many aspects of the building have to be addressed not just wind loading.

5.2 Further Improvements

While the development and testing of the FSIO method were successful there are still a number of areas that can be further improved upon within the development of the Grasshopper based tool itself and the research process.

- The FSIO tool works by analysing the structural response due to static wind loads. However, as shown in the research, dynamic loads are arguably more important for very tall high-rises. This tool could be further improved by implementing a method for analysing the dynamic effects of wind and using that as an optimisation objective.

- Only one wind direction was looked at in this study however, the ability to analyse the impact of multiple wind directions would be much more valuable as this can have a big effect on the performance.

- The tool could be further validated by using a case study of a complex geometry high-rise building for which physical wind tunnel tests have been performed. One could then analyse the building with the FSIO tool to determine the accuracy and then optimise the building’s geometry to ascertain how much improvement could be made over the original.

- Model-based optimisation, specifically RBFOpt in Opossum, was chosen as the optimisation algorithm due to its favourable reviews in benchmark studies and its ability to converge an optimisation problem in a relatively small number of iterations. However, it is not known for sure whether this is truly the best for this specific case especially as there was little found research on CFD optimisation. In further work, a benchmark study could be performed using different algorithms to determine which is best for this method.

6 BIBLIOGRAPHY

ANDERSON, J. D. 1995. *Computational fluid dynamics : the basic with applications,* New York, McGraw-Hill.

AYNSLEY, R. M. 1999. Shape and Flow: The Essence of Architectural Aerodynamics. *Architectural Science Review,* 42, 69-74.

AYNSLEY, R. M., MELBOURNE, W. H. & VICKERY, B. J. 1977. *Architectural aerodynamics,* London, Applied Science Publishers.

BERNARDINI, E., SPENCE, S. M. J., WEI, D. & KAREEM, A. 2015. Aerodynamic shape optimization of civil structures: A CFD-enabled Kriging-based approach. *Journal of Wind Engineering and Industrial Aerodynamics,* 144, 154-164.

BLOCKEN, B. 2014. 50 years of Computational Wind Engineering: Past, present and future. *Journal of Wind Engineering and Industrial Aerodynamics,* 129, 69-102.

BUNGARTZ, H. J. & SCHÄFER, M. 2006. Fluid-structure interaction : modelling, simulation, optimisation. Berlin: Springer-Verlag.

CHRONIS, A., TURNER, A. & TSIGKARI, M. 2011. Generative fluid dynamics: integration of fast fluid dynamics and genetic algorithms for wind loading optimization of a free form surface. *Proceedings of the 2011 Symposium on Simulation for Architecture and Urban Design.* Boston, Massachusetts: Society for Computer Simulation International.

CLANNACHAN, G., LIM, J., BICANIC, N., TAYLOR, I. & J. SCANLON, T. 2009. *Practical Application of CFD for Wind Loading on Tall Buildings*.

COCHRAN, L. & ASCE. COMMITTEE ON STRUCTURAL WIND, E. 2012. Wind issues in the design of buildings. Reston, Va.: American Society of Civil Engineers.

CONN, A. R., SCHEINBERG, K. & VICENTE, L. N. 2009. *Introduction to derivative-free optimization*, Siam.

COOK, N. 2007. *Designers' Guide to EN 1991-1-4 Eurocode 1 - Actions on Structures, General Actions, Part 1-4: Wind Actions*, ICE Publishing.

DONALDYTONG 2012. Burj Khalifa. *In:* KHALIFA.JPG, B. (ed.). commons: Wikipedia.

EKICI, B., CUBUKCUOGLU, C., TURRIN, M. & SARIYILDIZ, I. S. 2019. Performative computational architecture using swarm and evolutionary optimisation: A review. *Building and Environment,* 147, 356-371.

EVINS, R. 2013. A review of computational optimisation methods applied to sustainable building design. *Renewable and Sustainable Energy Reviews,* 22, 230-245.

FEBLOWITZ, J. C. 2010. Confusing The Wind: The Burj Khalifa, Mother Nature, and the Modern Skyscraper. *Inquiries Journal/Student Pulse* [Online], 2. Available: http://www.inquiriesjournal.com/a?id=124.

FRANKE, J., HELLSTEN, A., SCHLÜNZEN, H. & CARISSIMO, B. 2007. *Best Practice Guideline for the CFD Simulation of Flows in the Urban Environment*.

FRANSOS, D. & LO GIUDICE, A. 2015. *On the use of computational simulation in the determination of wind loads on structures: design experiences and food for thought*.

GREGSON, S. 2018. Nelder-Mead Optimisation Component in Grasshopper. London, UK.

HOLMES, J. D. 2007. *Wind loading of structures,* New York, Taylor and Francis.

HOSCH, W. 2018. *Navier-Stokes Equation* [Online]. Encyclopædia Britannica: Encyclopædia Britannica, inc. Available: https://www.britannica.com/science/Navier-Stokes-equation [Accessed December 20 2018].

HUANG, S., LI, Q. S. & XU, S. 2007. Numerical evaluation of wind effects on a tall steel building by CFD. *Journal of Constructional Steel Research,* 63, 612-627.

ILUNGA, G. & LEITÃO, A. 2018. Derivative-free Methods for Structural Optimization.

IRWIN, P. A. 2009. Wind engineering challenges of the new generation of super-tall buildings. *Journal of Wind Engineering and Industrial Aerodynamics,* 97, 328-334.

MAHER, M. L., POON, J. & BOULANGER, S. 1996. Formalising Design Exploration as Co-Evolution. Springer US.

MELBOURNE, W. H. 1980. Comparison of measurements on the CAARC standard tall building model in simulated model wind flows. *Journal of Wind Engineering and Industrial Aerodynamics,* 6, 73-88.

MELILLO, J. M., RICHMOND, T. & YOHE, G. W. 2014. *Climate Change Impacts in the United States: The Third National Climate Assessment* [Online]. Washington: U.S. Global Change Research Program. Available: http://nca2014.globalchange.gov/ [Accessed December 31 2018].

MENDIS, P., NGO, T., HARITOS, N., HIRA, A., SAMALI, B. & CHEUNG, J. 2007. *Wind loading on tall buildings.*

MENG, F.-Q., HE, B.-J., ZHU, J., ZHAO, D.-X., DARKO, A. & ZHAO, Z.-Q. 2018. Sensitivity analysis of wind pressure coefficients on CAARC standard tall buildings in CFD simulations. *Journal of Building Engineering,* 16, 146-158.

MENTER, F. R. 2009. Review of the shear-stress transport turbulence model experience from an industrial perspective. *International Journal of Computational Fluid Dynamics,* 23, 305-316.

MOHOTTI, D., MENDIS, P. & NGO, T. 2014. *APPLICATION OF COMPUTATIONAL FLUID DYNAMICS (CFD) IN PREDICTING THE WIND LOADS ON TALL BUILDINGS-A CASE STUDY.*

NEN, N. S. I. 2005. Eurocode 1: Actions on structures - Part 1-4: General actions - Wind actions. *NEN-EN1994-1-4:2005.* NEN.

NEN, N. S. I. 2011. National Annex to NEN-EN 1991-1-4+A1+C2: Eurocode 1: Actions on structures - Part 1-4: General actions - Wind actions. *NEN-EN 1991-1-4+A1+C2.* NEN.

OXMAN, R. 2006. *Theory and design in the first digital age*.

PREISINGER, C. 2013. Linking structure and parametric geometry. *Architectural Design,* 83, 110-113.

RAHMAN, M. M., KARIM, M. M. & ALIM, M. A. 2007. Numerical investigation of unsteady flow past a circular cylinder using 2-D finite volume method. *Journal of Naval Architecture and Marine Engineering,* 4, 27-42.

RIOS, L. M. & SAHINIDIS, N. V. J. J. O. G. O. 2013. Derivative-free optimization: a review of algorithms and comparison of software implementations. 56, 1247-1293.

SHIH, T. H., LIOU, W., SHABBIR, A., YANG, Z. & ZHU, J. 1994. *A New k-(Eddy Viscosity Model for High Reynolds Number Turbulent Flows - Model Development and Validation*.

SIMIU, E. & SCANLAN, R. H. 1996. *Wind effects on structures : fundamentals and applications to design,* New York, John Wiley.

SIMON, H. A. 1973. The structure of ill structured problems. *Artificial Intelligence,* 4, 181-201.

SMITS, D. R. J. A. D. S. P. G. D. A. J. 2018. *Drag of Blunt and Streamlined Bodies* [Online]. eFluids. Available: http://www.efluids.com/efluids/bicycle/bicycle_pages/blunt.jsp [Accessed 11/12 2018].

STAM, J. 1999. Stable fluids. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques.* ACM Press/Addison-Wesley Publishing Co.

SUN, D., OWEN, J. S. & WRIGHT, N. G. 2009. Application of the k–ω turbulence model for a wind-induced vibration study of 2D bluff bodies. *Journal of Wind Engineering and Industrial Aerodynamics,* 97, 77-87.

TAMURA, T., NOZAWA, K. & KONDO, K. 2008. AIJ guide for numerical prediction of wind loads on buildings. *Journal of Wind Engineering and Industrial Aerodynamics,* 96, 1974-1984.

TARANATH, B. S. 2012. Structural analysis and design of tall buildings : steel and composite construction. Boca Raton, Fla. [Washington, D.C.]: CRC Press; International Code Council.

THÉVENIN, D. & JANIGA, G. B. 2008. Optimization and computational fluid dynamics. Berlin: Springer Verlag.

TURRIN, M., VON BUELOW, P. & STOUFFS, R. 2011. Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. *Advanced Engineering Informatics,* 25, 656-675.

VAN HOOFF, T., BLOCKEN, B. & VAN HARTEN, M. 2011. 3D CFD simulations of wind flow and wind-driven rain shelter in sports stadia: Influence of stadium geometry. *Building and Environment,* 46, 22-37.

VONGSINGHA, P. 2015. *Adaptive Façade for Windload Reduction in High-rise.* MSc., TU Delft.

WAIBEL, C., BYSTRICKY, L., KUBILAY, A., EVINS, R. & CARMELIET, J. 2017. Validation of Grasshopper-based Fast Fluid Dynamics for Air Flow around Buildings in Early Design Stage.

WAIBEL, C., WORTMANN, T., EVINS, R. & CARMELIET, J. 2019. Building energy optimization: An extensive benchmark of global search algorithms. *Energy and Buildings,* 187, 218-240.

WENDT, J. F., ANDERSON, J. D., JR. & VON KARMAN INSTITUTE FOR FLUID, D. 2009. Computational fluid dynamics : an introduction. 3rd ed. ed. Berlin ;: Springer.

WORTMANN, T. 2017. *Opossum: Introducing and Evaluating a Model-based Optimization Tool for Grasshopper.*

WORTMANN, T. 2018. Genetic evolution vs. function approximation: Benchmarking algorithms for architectural design optimization. *Journal of Computational Design and Engineering*.

WORTMANN, T., COSTA, A., NANNICINI, G. & SCHROEPFER, T. 2015. *Advantages of surrogate models for architectural design optimization.*

WORTMANN, T. & NANNICINI, G. 2016. *Black-box optimisation methods for architectural design*.

WORTMANN, T., WAIBEL, C., NANNICINI, G., EVINS, R., SCHROEPFER, T. & CARMELIET, J. 2017. *Are Genetic Algorithms Really the Best Choice in Building Energy Optimization?*

YAKHOT, V., ORSZAG, S. A., THANGAM, S., GATSKI, T. B. & SPEZIALE, C. G. 1992. Development of turbulence models for shear flows by a double expansion technique. 4**,** 1510-1520.

YANG, D., REN, S., TURRIN, M., SARIYILDIZ, S. & SUN, Y. 2018. Multi-disciplinary and multi-objective optimization problem re-formulation in computational design exploration: A case of conceptual sports building design. *Automation in Construction,* 92**,** 242-269.

YANG, D., SUN, Y., DI STEFANO, D., TURRIN, M. & SARIYILDIZ, S. Impacts of problem scale and sampling strategy on surrogate model accuracy: An application of surrogate-based optimization in building design. 2016 2016. IEEE.

ZUO, W. & CHEN, Q. 2009. Real-time or faster-than-real-time simulation of airflow in buildings. *Indoor air,* 19**,** 33-44.

6.1 Table of Figures

7 APPENDIX

7.1 Appendix 1 – Navier-Stokes equations

This appendix lists the full sets of equations for describing fluid motion (Wendt et al., 2009).

7.1.1 Euler Equations (Inviscid Flow)

Inviscid flow is a flow where the dissipative, transport phenomena of viscosity, mass diffusion and thermal conductivity are neglected (Wendt et al., 2009).

Continuity equation
(Non-conservation form)

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \vec{V} = 0$$

(Conservation form)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0$$

Momentum equations
(Non-conservation form)

$$x\text{-component}: \quad \rho \frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \rho f_x$$

$$y\text{-component}: \quad \rho \frac{Dv}{Dt} = -\frac{\partial p}{\partial y} + \rho f_y$$

$$z\text{-component}: \quad \rho \frac{Dw}{Dt} = -\frac{\partial p}{\partial z} + \rho f_z$$

(Conservation form)

$$x\text{-component}: \quad \frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \vec{V}) = -\frac{\partial p}{\partial x} + \rho f_x$$

$$y\text{-component}: \quad \frac{\partial(\rho v)}{\partial t} + \nabla \cdot (\rho v \vec{V}) = -\frac{\partial p}{\partial y} + \rho f_y$$

$$z\text{-component}: \quad \frac{\partial(\rho w)}{\partial t} + \nabla \cdot (\rho w \vec{V}) = -\frac{\partial p}{\partial z} + \rho f_z$$

Energy equation
(Non-conservation form)

$$\rho \frac{D}{Dt}\left(e + \frac{V^2}{2}\right) = p\dot{q} - \frac{\partial(up)}{\partial x} - \frac{\partial(vp)}{\partial y} - \frac{\partial(wp)}{\partial z} + \rho \vec{f} \cdot \vec{V}$$

(Conservation form)

$$\frac{\partial}{\partial t}\left[\rho\left(e + \frac{V^2}{2}\right)\right] + \nabla \cdot \left[\rho\left(e + \frac{V^2}{2}\right)\vec{V}\right] = \rho\dot{q} - \frac{\partial(up)}{\partial x} - \frac{\partial(vp)}{\partial y} - \frac{\partial(wp)}{\partial z} + \rho \vec{f} \cdot \vec{V}$$

7.1.2 Navier-Stokes Equations (Viscous Flow)

These equations describe viscous flow, i.e. flow that includes the dissipative transport phenomena of viscosity and thermal conduction. The additional transport phenomena for mass diffusion is not included as we are describing a homogenous, nonreactive gas (Wendt et al., 2009).

Continuity equations
(Non-conservation form—Eq. (2.18))

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \vec{V} = 0$$

(Conservation form—Eq. (2.27))

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0$$

Momentum equations
(Non-conservation form—Eqs. (2.36a–c))

$$x\text{-component}: \quad \rho \frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x$$

$$y\text{-component}: \quad \rho \frac{Dv}{Dt} = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y$$

$$z\text{-component}: \quad \rho \frac{Dw}{Dt} = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z$$

(Conservation form—Eqs. (2.42a–c))

$$x\text{-component}: \quad \frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \vec{V}) = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x$$

$$y\text{-component}: \quad \frac{\partial(\rho v)}{\partial t} + \nabla \cdot (\rho v \vec{V}) = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y$$

$$z\text{-component}: \quad \frac{\partial(\rho w)}{\partial t} + \nabla \cdot (\rho w \vec{V}) = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z$$

Energy equation
(Non-conservation form—Eq. (2.52))

$$\rho\frac{D}{Dt}\left(e+\frac{V^2}{2}\right)=\rho\dot{q}+\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)$$
$$-\frac{\partial(up)}{\partial x}-\frac{\partial(vp)}{\partial y}-\frac{\partial(wp)}{\partial z}+\frac{\partial(u\tau_{xx})}{\partial x}$$
$$+\frac{\partial(u\tau_{yx})}{\partial y}+\frac{\partial(u\tau_{zx})}{\partial z}+\frac{\partial(v\tau_{xy})}{\partial x}+\frac{\partial(v\tau_{yy})}{\partial y}$$
$$+\frac{\partial(v\tau_{zy})}{\partial z}+\frac{\partial(w\tau_{xz})}{\partial x}+\frac{\partial(w\tau_{yz})}{\partial y}+\frac{\partial(w\tau_{zz})}{\partial z}+\rho\vec{f}\cdot\vec{V}$$

(Conservation form—Eq. (2.64))

$$\frac{\partial}{\partial t}\left[\rho\left(e+\frac{V^2}{2}\right)\right]+\nabla\cdot\left[\rho\left(e+\frac{V^2}{2}\vec{V}\right)\right]$$
$$=\rho\dot{q}+\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)$$
$$+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)-\frac{\partial(up)}{\partial x}-\frac{\partial(vp)}{\partial y}-\frac{\partial(wp)}{\partial z}+\frac{\partial(u\tau_{xx})}{\partial x}$$
$$+\frac{\partial(u\tau_{yx})}{\partial y}+\frac{\partial(u\tau_{zx})}{\partial z}+\frac{\partial(v\tau_{xy})}{\partial x}+\frac{\partial(v\tau_{yy})}{\partial y}$$
$$+\frac{\partial(v\tau_{zy})}{\partial z}+\frac{\partial(w\tau_{xz})}{\partial x}+\frac{\partial(w\tau_{yz})}{\partial y}+\frac{\partial(w\tau_{zz})}{\partial z}+\rho\vec{f}\cdot\vec{V}$$

7.2 Appendix 2 – CFD validation

7.2.1 CFD Validation results

Test			TJ(D)	BF_1	BF_2	BF_3	FFD
Turbulence model			N/A	kEpsilon	realizableKE	RNGkEpsilon	v = 0.1
Cell size (m)			N/A	0.98755	0.98755	0.98755	5
no. of cells			N/A	615 120	615 120	616 120	1 728 000
Iterations			N/A	30000	30000	30000	600
Time			N/A	41.7h	37.4h	42.6h	9.2h
Pressure coefficient, Cp	Front	1	0.61	0.83	1.75	0.62	0.21
		2	0.87	0.97	1.84	0.81	0.30
		3	0.89	0.99	1.83	0.84	0.31
		4	0.89	0.97	1.75	0.81	0.30
		5	0.63	0.83	-1.76	0.62	0.19
	Left	6	-0.84	-1.12	-1.23	-0.71	-0.23
		7	-0.87	-0.89	-1.07	-0.72	-0.15
		8	-0.89	-0.66	-1.08	-0.68	-0.12
		9	-0.89	-0.53	-1.57	-0.58	-0.10
		10	-0.94	-0.43	-4.32	-0.57	-0.09
	Rear	11	-0.71	-0.38	-3.85	-0.38	-0.09
		12	-0.68	-0.36	-3.78	-0.35	-0.09
		13	-0.66	-0.35	-3.85	-0.34	-0.09
		14	-0.66	-0.36	-4.32	-0.35	-0.09
		15	-0.72	-0.38	-1.54	-0.38	-0.09
	Right	16	-0.92	-0.43	-1.07	-0.56	-0.09
		17	-0.86	-0.52	-1.05	-0.59	-0.10
		18	-0.83	-0.65	-1.05	-0.68	-0.11
		19	-0.83	-0.87	-1.20	-0.73	-0.14
		20	-0.80	-1.19	-1.73	-0.73	-0.23

7.2.2 Cp of turbulence model sensitivity analysis

From Meng et al. (2018) courtesy of co-author Baojie He.

NO.	Standard	RNG	Realizable	SST	BSL	TJ
1	0.64203	0.62405	0.63719	0.63694	0.58418	0.56343
2	0.86740	0.86104	0.86992	0.86719	0.91843	0.79758
3	0.90867	0.90675	0.91301	0.90807	0.96502	0.85394
4	0.86497	0.86535	0.87009	0.86424	0.91862	0.80929
5	0.63260	0.63569	0.63846	0.62679	0.58465	0.57070
6	-0.74235	-0.69798	-0.72661	-0.71705	-0.85334	-0.71434
7	-0.77215	-0.71447	-0.75695	-0.78587	-0.86121	-0.69435
8	-0.76761	-0.70998	-0.77649	-0.80605	-0.90704	-0.77940
9	-0.70157	-0.61904	-0.72372	-0.69233	-0.80108	-0.74728
10	-0.70472	-0.59103	-0.69386	-0.71559	-0.62045	-0.80001
11	-0.59608	-0.54242	-0.58018	-0.53124	-0.47155	-0.53354
12	-0.59200	-0.55271	-0.55210	-0.51010	-0.46453	-0.48526
13	-0.59372	-0.55831	-0.53055	-0.50280	-0.46287	-0.46526
14	-0.58985	-0.55684	-0.54639	-0.49055	-0.46459	-0.47758
15	-0.56329	-0.55188	-0.57858	-0.49106	-0.47147	-0.53031
16	-0.62959	-0.65635	-0.67693	-0.65882	-0.62179	-0.77697
17	-0.65807	-0.67288	-0.71664	-0.62373	-0.80337	-0.75293
18	-0.73689	-0.76660	-0.76871	-0.68804	-0.90974	-0.73697
19	-0.73556	-0.76476	-0.74908	-0.69438	-0.86563	-0.70081
20	-0.68510	-0.74364	-0.74132	-0.66100	-0.85862	-0.68082

7.3 Appendix 3 – Sensitivity analysis

7.3.1 Sensitivity analysis results - Number of iterations - Butterfly

Test			RNG_30k	RNG_10k	RNG_5k
Turbulence model			RNGkEpsilon	RNGkEpsilon	RNGkEpsilon
Cell size (m)			0.987552	0.987552	0.987552
no. of cells			615 120	615 120	615 120
Iterations			30000	10000	5000
Time			42.6h	15.7h	6.95h
Pressure coefficient, Cp	Front	1	0.62	0.62	0.62
		2	0.81	0.81	0.81
		3	0.84	0.84	0.84
		4	0.81	0.81	0.81
		5	0.62	0.62	0.62
	Left	6	-0.71	-0.71	-0.71
		7	-0.72	-0.72	-0.72
		8	-0.68	-0.68	-0.68
		9	-0.58	-0.58	-0.58
		10	-0.57	-0.57	-0.57
	Rear	11	-0.38	-0.38	-0.38
		12	-0.35	-0.35	-0.35
		13	-0.34	-0.34	-0.34
		14	-0.35	-0.35	-0.35
		15	-0.38	-0.38	-0.38
	Right	16	-0.56	-0.57	-0.57
		17	-0.59	-0.59	-0.59
		18	-0.68	-0.69	-0.68
		19	-0.73	-0.73	-0.73
		20	-0.73	-0.73	-0.73

7.3.2 Sensitivity analysis results - Number of iterations – GH Wind

Test			FFD_600	FFD_400
Turbulence model			v = 0.1	v = 0.1
Cell size (m)			5	5
no. of cells			1 728 000	1 728 000
Iterations			600	400
Time			9.2h	5.7h
Pressure coefficient, Cp	Front	1	0.62	0.62
		2	0.81	0.81
		3	0.84	0.84
		4	0.81	0.81
		5	0.62	0.62
	Left	6	-0.71	-0.71
		7	-0.72	-0.72
		8	-0.68	-0.68
		9	-0.58	-0.58
		10	-0.57	-0.57
	Rear	11	-0.38	-0.38
		12	-0.35	-0.35
		13	-0.34	-0.34
		14	-0.35	-0.35
		15	-0.38	-0.38
	Right	16	-0.56	-0.57
		17	-0.59	-0.59
		18	-0.68	-0.69
		19	-0.73	-0.73
		20	-0.73	-0.73

7.3.3 Sensitivity analysis – Mesh size – Butterfly

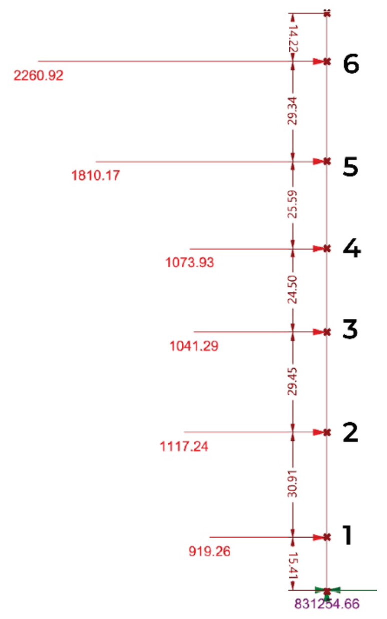
		Test	MAD_1	MAD_2	MAD_3	MAD_4	MAD_5
		Turbulence model	RNGkEpsilon	RNGkEpsilon	RNGkEpsilon	RNGkEpsilon	RNGkEpsilon
		Resolution	Coarse	Medium	Fine	Super Fine	XXFine
		Cell size (m)	4.18	2.96	2.09	1.48	1.08
		no. of cells	176545	236050	346647	525640	732422
		Time	5.7h	6.3h	8.6h	14.5h	20.3h
		Iterations	10000	10000	10000	10000	10000
Cp at pressure tap	**Front**	**0**	0.04	0.06	0.01	0.01	0.01
		1	0.29	0.27	0.19	0.18	0.23
		2	0.16	0.14	-0.21	-0.29	-0.29
		3	0.09	0.01	-0.32	-0.35	-0.36
		4	0.47	0.49	0.36	0.29	0.39
		5	0.17	0.18	0.18	0.18	0.17
		6	0.38	0.40	0.46	0.43	0.42
		7	0.65	0.65	0.58	0.59	0.62
		8	0.71	0.78	0.80	0.79	0.79
		9	0.86	0.91	0.92	0.92	0.94
		10	0.04	0.09	0.09	0.05	0.05
		11	0.07	-0.05	-0.06	-0.04	-0.04
		12	-0.1	-0.25	-0.49	-0.50	-0.39
		13	0.53	0.56	0.60	0.55	0.45
		14	0.5	0.51	0.37	0.33	0.38
	Rear	**15**	-0.38	-0.37	-0.35	-0.31	-0.31
		16	-0.37	-0.36	-0.33	-0.29	-0.29
		17	-0.39	-0.56	-0.59	-0.49	-0.62
		18	-0.59	-0.55	-0.53	-0.53	-0.51
		19	-0.49	-0.48	-0.48	-0.45	-0.49
		20	-0.36	-0.34	-0.32	-0.28	-0.28
		21	-0.36	-0.35	-0.31	-0.28	-0.29
		22	-0.39	-0.38	-0.32	-0.23	-0.18
		23	-0.58	-0.55	-0.55	-0.52	-0.53
		24	-0.45	-0.40	-0.40	-0.37	-0.38
		25	-0.38	-0.40	-0.35	-0.31	-0.30
		26	-0.38	-0.37	-0.35	-0.31	-0.32
		27	-0.45	-0.49	-0.65	-0.64	-0.71
		28	-0.58	-0.53	-0.54	-0.50	-0.50
		29	-0.5	-0.46	-0.42	-0.41	-0.44

7.3.4 Sensitivity analysis – Mesh size – GH Wind

		Test	MAD_6	MAD_7	MAD_8
		Turbulence model	FFD	FFD	FFD
		Resolution	Medium	Fine	SuperFine
		Cell size (m)	10	8	6
		no. of cells	390830	1423254	1824912
		Time	3.4h	7.1h	11.8h
		Iterations	400	400	400
Cp at pressure tap	**Front**	**0**	-0.01	0.05	-0.02
		1	0.02	0.06	-0.24
		2	0.00	0.07	-0.21
		3	-0.05	0.10	-0.05
		4	-0.02	0.21	-0.06
		5	0.06	0.07	-0.04
		6	0.11	0.21	-0.10
		7	0.03	0.25	-0.11
		8	0.18	0.23	-0.12
		9	0.18	0.25	-0.16
		10	0.00	0.00	-0.15
		11	-0.02	-0.01	-0.26
		12	-0.02	-0.05	-0.35
		13	0.05	0.13	-0.31
		14	0.01	0.10	-0.41
	Rear	**15**	-0.01	-0.05	-0.08
		16	0.00	-0.03	0.05
		17	-0.01	-0.03	-0.03
		18	-0.04	-0.03	-0.43
		19	-0.05	-0.03	-0.39
		20	0.00	-0.03	-0.08
		21	0.00	-0.02	0.02
		22	0.00	-0.02	-0.09
		23	-0.01	-0.03	-0.21
		24	-0.04	-0.03	-0.28
		25	-0.01	-0.05	-0.06
		26	-0.01	-0.03	0.00
		27	-0.02	-0.03	-0.13
		28	-0.03	-0.03	-0.22
		29	-0.05	-0.04	-0.17

7.4 Appendix 4 – Hand calculations

Hand calculations for verifying FEA



	Loads	
Node	**Forces[kN]**	**Moments [kNm]**
1	919.26	-333.55
2	1117.24	3946.49
3	1041.29	11350.61
4	1073.93	-7067.46
5	1810.17	-5958.61
6	2260.92	1780.07

	d between pts [m]	**d from base [m]**
1	15.41	15.41
2	30.91	46.32
3	29.45	75.77
4	24.50	100.27
5	25.59	125.86
6	29.34	155.20
7	14.22	169.42
	Total height	169.42

	Deflection [m]
1	0.0036
2	0.0368
3	0.0859
4	0.1464
5	0.3644
6	0.6390
Total	1.2761

Core Dimensions [m]	
Height	8.60
Width	8.00
Thickness	0.40
E [kN/m2] =	3.60E+07
I [m4] =	139.3061

My [kNm]	831220.6
Mz [kNm]	3717.56
Max stress [MPa]	25.66

7.5 Appendix 5 – Eurocode calculation

Wind force (F_w) calculation for Absolute Tower using EN1991-1-4

EN = NEN-EN1991-1-4:2005

EN-NA = Netherlands National Annex to NEN-EN1991-1-4+A1+C2

Basic wind velocity, v_b	= 30 m/s
Building height, h	= 170m
Building width, b	= 40.13m
Height above ground, z	= 76.29m

$$F_w = c_s c_d \cdot c_f \cdot q_p(z_e) \cdot A_{ref}$$

F_w = 927.85 N/m^2

Where:

$q_p(z)$ =	Peak velocity pressure at height z (eq. 1)
c_f =	Force coefficient (eq. 2)
$c_s c_d$ =	Structural factor (eq. 3)
A_{ref} =	Reference area of structure = 1m^2

$$q_p(z) = [1 + 7 \cdot I_v(z)] \cdot \frac{1}{2} \cdot \rho \cdot v_m^2(z) \quad \text{(eq. 1)}$$

$q_p(z)$ = 1487.124 N/m^2

Where:

$I_v(z)$ =	Turbulence intensity at height z (eq. 1.1)
ρ =	Air density = 1.225 kg/m^3
v_m^2 =	Mean wind velocity at height z (eq. 1.2)

$$I_v(z) = \frac{k_I}{c_0(z) \cdot \ln\left(\frac{z}{z_0}\right)} \quad \text{(eq. 1.1)}$$

$I_v(z)$ = 0.2307

Where:

k_I =	Turbulence factor = 1.0 (EN 4.4)
c_o =	Orography factor = 1.0 (EN-NA A.3)
z =	height above ground
z_0 =	Roughness length = 1m

$$v_m(z) = c_r(z) \cdot c_o(z) \cdot v_b \quad \text{(eq. 1.2)}$$

$v_m(z)$ = 30.47 m/s

Where:

$c_r(z)$ =	Roughness factor
v_b =	Basic wind velocity

$$c_r(z) = k_r \cdot \ln\left(\frac{z}{z_0}\right) \quad \text{(eq. 1.2.1)}$$

$c_r(z)$ = 1.0157

Where:

k_r =	Terrain factor
z =	height above ground
z_0 =	Roughness length

$k_r = 0.19 \cdot \left(\frac{z_0}{z_{0,II}}\right)^{0.07}$ (eq. 1.2.2)

$\boldsymbol{k_r}$ = 0.2343

Where:

z_0 =	Roughness length
$z_{0,II}$ =	Roughness length for terrain category II = 0.05m

---

$c_f = c_{f,0} \cdot \psi_\lambda$ (eq. 2)

$\boldsymbol{c_f}$= 0.5890

Where:

$c_{f,0}$ =	Force coefficient without free-end flow for a circular cylinder (EN 7.9.2 – figure 7.28 using Re (eq. 2.1))
ψ_λ =	End-effect factor = 0.68 (EN 7.13)

$Re = \frac{b \cdot v(z_e)}{\nu}$ (eq. 2.1)

$\boldsymbol{Re}$ = 1.32E+08

Where:

Re =	Reynold's number
b =	Building width
$v(z_e)$ =	Peak wind velocity at height z
ν =	Kinematic viscosity of air = 1.5E-6 m^2/s

$v = \sqrt{\frac{2 \cdot q_p}{\rho}}$ (eq. 2.1.1)

$\boldsymbol{v}$ = 49.2743 m/s

Where:

$q_p(z)$ = Peak velocity pressure at height z

ρ = Air density = 1.225 kg/m^3

$c_s c_d = \frac{1+2 \cdot k_p \cdot I_v(z_s)\sqrt{B^2+R^2}}{1+7 \cdot I_v(z_s)}$ (eq. 3)

$\boldsymbol{c_s c_d}$ = 1.0593

Where:

R^2 =	Resonance response factor
B^2 =	Background factor
k_p =	Peak factor

$Iv(z_s)$ = Turbulence intensity at reference height for structural factor $z_s = 0.6 \cdot h$

$R^2 = \frac{\pi^2}{2 \cdot \delta} \cdot S_L(z_s, n_{1,x}) \cdot K_s(n_{1,x})$ (eq. 3.1)

$\boldsymbol{R^2}$ = 0.9227

Where:

δ =	Total logarithmic decrement of damping
S_L =	Wind power spectral density function at reference height z_s at the natural frequency of the building, $n_{1,x}$
K_s =	Size reduction function at natural frequency, $n_{1,x}$

$$\delta = \delta_s + \delta_a + \delta_d$$ (eq. 3.1.1)

δ = 0.0851

Where:

δ_s = Logarithmic decrement of structural damping = 0.08 (EN F.5 – Table F.2)

δ_a = Logarithmic decrement of aerodynamic damping

δ_d = Logarithmic decrement of damping due to special devices = 0

$$\delta_a = \frac{c_f \cdot \rho \cdot v_m(z_s)}{2 \cdot n_1 \cdot \mu_e}$$ (eq. 3.1.1.1)

δ_a = 0.0051

Where:

c_f = Force coefficient

$v_m(z_s)$ = Mean wind speed at z_s

n_1 = Natural frequency of building = 46/h (EN F.2)

μ_e = Equivalent mass per unit area = 347.4 kg/m^3 · b (Vongsingha, 2015)

$$S_L(z,n) = \frac{6.8 \cdot f_L(z,n)}{\left(1+10.2 \cdot f_L(z,n)\right)^{5/3}}$$ (eq. 3.1.2)

$S_L(z,n)$ = 0.1015

Where:

$f_L(z,n)$ = Non-dimensional frequency determined by natural frequency $n_{1,x}$

$$f_L(z,n) = \frac{n \cdot L(z)}{v_m(z)}$$ (eq. 3.2.1)

$f_L(z,n)$ = 1.3926

Where:

n = Natural frequency of the building

$L(z)$ = Turbulence length scale at height z

$V_m(z)$ = Mean wind speed at height z

$$L(z) = L_t \cdot \left(\frac{z}{z_t}\right)^{\alpha}$$ (eq. 3.2.1.1)

$L(z)$ = 157.284

Where:

L_t = Reference length scale = 300m

z_t = Reference height = 200m

$\alpha = 0.06 + 0.05 \cdot \ln(z_0)$

$$K_s(n) = \frac{1}{1+\sqrt{\left(G_y \cdot \phi_y\right)^2 + (G_z \cdot \phi_z)^2 + \left(\frac{2}{\pi} \cdot G_y \cdot \phi_y \cdot G_z \cdot \phi_z\right)^2}}$$ (eq. 3.1.3)

$K_s(n)$ = 0.1569

Where:

Gy = 5/18 (EN C.2 – Table C.1)

Gz = ½ (ENC.2 – Table C.1)

$\phi_y = \frac{c_y \cdot b \cdot n}{v_m(z_s)}$ & $\phi_z = \frac{c_z \cdot h \cdot n}{v_m(z_s)}$

$c_y = c_z$ = Decay constants = 11.5 (EN C.2)

$$B^2 = \frac{1}{1+\frac{3}{2}\cdot\sqrt{\left(\frac{b}{L(z_S)}\right)^2+\left(\frac{h}{L(z_S)}\right)^2+\left(\frac{b}{L(z_S)}\cdot\frac{h}{L(z_S)}\right)^2}} \qquad \text{(eq. 3.2)}$$

B^2 = 0.4166

Where:

$L(z_s)$ = Turbulence length scale at reference height z_s

$$k_p = \sqrt{2\cdot\ln(\nu\cdot T)}+\frac{0.6}{\sqrt{2\cdot\ln(\nu\cdot T)}} \qquad \text{(eq. 3.3)}$$

k_p = 3.3223

Where:

ν =	Up-crossing frequency
T =	Averaging time for the mean wind velocity = 600 s

$$\nu = n_{1,x}\sqrt{\frac{R^2}{B^2+R^2}} \qquad \text{(eq. 3.3.1)}$$

ν = 0.2239

Where:

$n_{1,x}$ =	Natural frequency of the building
R^2 =	Resonance response factor
B^2 =	Background factor

7.6 Appendix 6 - Scripts

Custom Python and C# scripts used in the FSIO tool.

7.6.1 Blockage ratio calculation

```python
"""Provides a scripting component.
    Inputs:
        WTpoints: Corner Points from Wind Tunnel Component
        Geo: The geometry
    Output:
        BRatio: Blockage Ratio"""

__author__ = "ErronEstrado"
__version__ = "2019.01.21"

import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import math

def CreateTunnelBox(pts):
    ptStart = pts[0]
    ptEnd = pts[6]
    box = rg.BoundingBox(ptStart, ptEnd)
    return box.ToBrep()

def TunnelBoxIntersection(box, bldg):
    bBox = bldg.GetBoundingBox(False)
    BrepBox = bBox.ToBrep()
    props = rg.VolumeMassProperties.Compute(BrepBox)
    centerPt = props.Centroid
    pln = rg.Plane(centerPt, rg.Vector3d.ZAxis, rg.Vector3d.XAxis)
    result = rg.Intersect.Intersection.BrepPlane(box, pln, 0.001)[1]
    crv = rg.Curve.JoinCurves(result)
    return crv

def BuildingIntersection(bldg):
    bBox = bldg.GetBoundingBox(False)
    BrepBox = bBox.ToBrep()
    props = rg.VolumeMassProperties.Compute(BrepBox)
    centerPt = props.Centroid
    pln = rg.Plane(centerPt, rg.Vector3d.ZAxis, rg.Vector3d.XAxis)
    result = rg.Intersect.Intersection.BrepPlane(bldg, pln, 0.001)[1]
    crv = rg.Curve.JoinCurves(result)
    return crv

def ComputeBlockageRatio(tunnel, bldg):
    bldgArea = rg.AreaMassProperties.Compute(bldg).Area
    TunnelArea = rg.AreaMassProperties.Compute(tunnel).Area
    result = bldgArea/TunnelArea
    return result

TunnelBox = CreateTunnelBox(WTpoints)
TunnelCrv = TunnelBoxIntersection(TunnelBox, Geo)
BldgCrv = BuildingIntersection(Geo)

BRatio = ComputeBlockageRatio(TunnelCrv, BldgCrv)
```

7.6.2 Cell size selector

```python
"""Provides a scripting component.
    Inputs:
        _geo: The geometry
        _quality: Mesh quality
        0 = Coarse
        1 = Medium
        2 = Fine
        3 = SuperFine
        4 = XXFine
    Output:
        cellSize_: Cell size"""

__author__ = "ErronEstrado"
__version__ = "2019.01.25"

import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import math

bBox = _geo.GetBoundingBox(True)
Box = rg.Box(bBox)

dim = Box.X.Length
if Box.Y.Length < dim:
    dim = Box.Y.Length

base = 10
n = math.sqrt(2)

def quality(x):
    return{
        0 : base,
        1 : base * n,
        2 : base * n * n,
        3 : base * n * n * n,
        4 : base * n * n * n * n
    }.get(x,10)

div = quality(_quality)
size = dim / div
cellSize_ = format(size, '.2f')
```

7.6.3 Horizontal plane to visualize velocity vectors

```python
"""Creates horizontal plane to visualize velocity vectors.
    Inputs:
        _WTpoints: Corner points of wind tunnel box
        _geo: The geometry
        _height: Height to make plane
    Output:
        srf_: The output surface"""

__author__ = "Erron Estrado"
__version__ = "2019.01.28"

import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
from copy import copy

def CreateTunnelBox(pts):
    ptStart = pts[0]
    ptEnd = pts[6]
    box = rg.BoundingBox(ptStart, ptEnd)
    return box.ToBrep()

def GetSurface(brep, z, bldg):
    face = brep.Faces[4].ToBrep()
    srf = copy(face)
    move = rg.Vector3d(0, 0, z)
    srf.Translate(move)
    centerPt = rg.VolumeMassProperties.Compute(bldg).Centroid
    centerPt.Z = z
    centerPln = rg.Plane(centerPt, rg.Vector3d.ZAxis)
    centerScale = rg.Transform.Scale(centerPln, 0.3, 0.3, 1)
    srf.Transform(centerScale)
    return srf

def IntersectSurface(bldg, srf):
    cutter = copy(bldg)
    cutSrf = srf.Split(cutter, 0.001)
    return cutSrf[0]

bldgBox = _geo.GetBoundingBox(True)
centerPt = bldgBox.Center
bldgPln = rg.Plane(centerPt, rg.Vector3d.ZAxis)

scaling = rg.Transform.Scale(bldgPln, 1.1, 1.1, 1.0)
scaledBldg = copy(_geo)
scaledBldg.Transform(scaling)

box = CreateTunnelBox(_WTpoints)
pln = GetSurface(box, _height, _geo)
srf_ = IntersectSurface(scaledBldg, pln)
```

7.6.4 Vertical plane to visualize velocity vectors

```python
"""Creates vertical plane to visualize velocity vectors.
    Inputs:
        _WTpoints: Corner points of wind tunnel box
        _geo: The geometry
    Output:
        srf_: The output surface"""

__author__ = "Erron Estrado"
__version__ = "2019.01.28"

import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
from copy import copy

def CreateTunnelBox(pts):
    ptStart = pts[0]
    ptEnd = pts[6]
    box = rg.BoundingBox(ptStart, ptEnd)
    return box.ToBrep()

def GetSurface(brep, x, bldg):
    face = brep.Faces[3].ToBrep()
    srf = copy(face)
    move = rg.Vector3d(x, 0, 0)
    srf.Translate(move)
    centerPt = rg.VolumeMassProperties.Compute(bldg).Centroid
    #centerPt.X = x
    centerPt.Z = 0.0
    scalePln = rg.Plane(centerPt, rg.Vector3d.XAxis)
    scaling = rg.Transform.Scale(scalePln, 0.3, 0.6, 1)
    srf.Transform(scaling)
    return srf

def IntersectSurface(bldg, srf):
    cutter = copy(bldg)
    centerPt = rg.VolumeMassProperties.Compute(cutter).Centroid
    centerPt.Z = 0.0
    scalePln = rg.Plane(centerPt, rg.Vector3d.ZAxis)
    scaling = rg.Transform.Scale(scalePln, 1.1, 1.1, 1.01)
    cutter.Transform(scaling)
    cutSrf = srf.Split(cutter, 0.001)

    srfVel = cutSrf[0]
    if rg.AreaMassProperties.Compute(cutSrf[1]).Area > rg.AreaMassProperties.Compute(cutSrf[0]).Area:
        srfVel = cutSrf[1]
    return srfVel

boxBrep = CreateTunnelBox(_WTpoints)
box = rg.Box(rg.Plane.WorldXY, boxBrep)
position = box.X.Length / 2
surface = GetSurface(boxBrep, position, _geo)

srf_ = IntersectSurface(_geo, surface)
```

7.6.5 Get façade surface from building geometry

```csharp
using System;
using System.Collections;
using System.Collections.Generic;

using Rhino;
using Rhino.Geometry;

using Grasshopper;
using Grasshopper.Kernel;
using Grasshopper.Kernel.Data;
using Grasshopper.Kernel.Types;



/// <summary>
/// This class will be instantiated on demand by the Script component.
/// </summary>
public class Script_Instance : GH_ScriptInstance
{
#region Utility functions
  /// <summary>Print a String to the [Out] Parameter of the Script component.</summary>
  /// <param name="text">String to print.</param>
  private void Print(string text) { /* Implementation hidden. */ }
  /// <summary>Print a formatted String to the [Out] Parameter of the Script component.</summary>
  /// <param name="format">String format.</param>
  /// <param name="args">Formatting parameters.</param>
  private void Print(string format, params object[] args)  /* Implementation hidden. */ }impl
  /// <summary>Print useful information about an object instance to the [Out] Parameter of the Script component. </summary>
  /// <param name="obj">Object instance to parse.</param>
  private void Reflect(object obj) { /* Implementation hidden. */ }
  /// <summary>Print the signatures of all the overloads of a specific method to the [Out] Parameter of the Script component. </summary>
  /// <param name="obj">Object instance to parse.</param>
  private void Reflect(object obj, string method_name) { /* Implementation hidden. */ }
#endregion

#region Members
  /// <summary>Gets the current Rhino document.</summary>
  private readonly RhinoDoc RhinoDocument;
  /// <summary>Gets the Grasshopper document that owns this script.</summary>
  private readonly GH_Document GrasshopperDocument;
  /// <summary>Gets the Grasshopper script component that owns this script.</summary>
  private readonly IGH_Component Component;
  /// <summary>
  /// Gets the current iteration count. The first call to RunScript() is associated with Iteration==0.
  /// Any subsequent call within the same solution will increment the Iteration count.
  /// </summary>
  private readonly int Iteration;
#endregion

  /// <summary>
  /// This procedure contains the user code. Input parameters are provided as regular arguments,
  /// Output parameters as ref arguments. You don't have to assign output parameters,
  /// they will have a default value.
  /// </summary>
  private void RunScript(Brep _geo, ref object srf_)
  {
    BoundingBox bBox = _geo.GetBoundingBox(true);
    double height = bBox.Max.Z;
    List<Brep> surfaces = new List<Brep>();
    List<AreaMassProperties> props = new List<AreaMassProperties>();

    foreach (BrepFace face in _geo.Faces)
    {
      Brep faceSrf = face.ToBrep();
      Point3d centre = AreaMassProperties.Compute(faceSrf).Centroid;
      if (centre.Z > 0.1 && centre.Z < height - 0.1)
      {
        surfaces.Add(faceSrf);
      }
    }

    Brep[] facade = Brep.JoinBreps(surfaces, 0.01);

    Brep flippedBrep = FlipBrep(facade[0]);

    srf_ = ExplodeBrep(flippedBrep);

  }

  // <Custom additional code>

  static Brep FlipBrep(Brep geo)
  {
    Box bBox = new Box(geo.GetBoundingBox(true));
    double width = bBox.X.Length;
    if (bBox.Y.Length < width)
      width = bBox.Y.Length;

    Surface srf = geo.Faces[0].ToNurbsSurface();

    Interval dom = new Interval(0, 1);

    srf.SetDomain(0, dom);
    srf.SetDomain(1, dom);

    Vector3d normal = srf.NormalAt(0.5, 0.5);
    normal *= width * 0.2;
    Point3d pt = srf.PointAt(0.5, 0.5);
    pt += normal;

           Brep cappedGeo = geo.CapPlanarHoles(0.01);

           if (cappedGeo.IsPointInside(pt, 0.01, false))
           {
             geo.Flip();
           }

           return geo;
         }

         static List<Surface> ExplodeBrep(Brep geo)
         {
           List<Surface> faces = new List<Surface>();

           foreach (Surface srf in geo.Faces)
           {
             srf.ToBrep();
             faces.Add(srf);
           }
           return faces;
         }
         // </Custom additional code>
       }
```

7.6.6 Vertical data grouping for FSI translation

```
using System;
using System.Collections;
using System.Collections.Generic;

using Rhino;
using Rhino.Geometry;

using Grasshopper;
using Grasshopper.Kernel;
using Grasshopper.Kernel.Data;
using Grasshopper.Kernel.Types;



/// <summary>
/// This class will be instantiated on demand by the Script component.
/// </summary>
public class Script_Instance : GH_ScriptInstance
{
#region Utility functions
  /// <summary>Print a String to the [Out] Parameter of the Script component.</summary>
  /// <param name="text">String to print.</param>
  private void Print(string text) { /* Implementation hidden. */ }
  /// <summary>Print a formatted String to the [Out] Parameter of the Script component.</summary>
  /// <param name="format">String format.</param>
  /// <param name="args">Formatting parameters.</param>
  private void Print(string format, params object[] args) { /* Implementation hidden. */ }
  /// <summary>Print useful information about an object instance to the [Out] Parameter of the Script component. </summary>
  /// <param name="obj">Object instance to parse.</param>
  private void Reflect(object obj) { /* Implementation hidden. */ }
  /// <summary>Print the signatures of all the overloads of a specific method to the [Out] Parameter of the Script component. </summary>
  /// <param name="obj">Object instance to parse.</param>
  private void Reflect(object obj, string method_name) { /* Implementation hidden. */ }
#endregion

#region Members
  /// <summary>Gets the current Rhino document.</summary>
  private readonly RhinoDoc RhinoDocument;
  /// <summary>Gets the Grasshopper document that owns this script.</summary>
  private readonly GH_Document GrasshopperDocument;
  /// <summary>Gets the Grasshopper script component that owns this script.</summary>
  private readonly IGH_Component Component;
  /// <summary>
  /// Gets the current iteration count. The first call to RunScript() is associated with Iteration==0.
  /// Any subsequent call within the same solution will increment the Iteration count.
  /// </summary>
  private readonly int Iteration;
#endregion

  /// <summary>
  /// This procedure contains the user code. Input parameters are provided as regular arguments,
  /// Output parameters as ref arguments. You don't have to assign output parameters,
  /// they will have a default value.
  /// </summary>
  private void RunScript(DataTree<System.Object> tree, int zDiv, ref object newTree)
  {

    IList<GH_Path> paths = tree.Paths;
    GH_Path lastBranch = paths[paths.Count - 1];
    int lenZ = lastBranch.Indices[0];
    int lenX = lastBranch.Indices[1];

    int zGrouping = (lenZ + 1) / zDiv;

    newTree = Grouping(tree, lenX, lenZ, zGrouping);

  }

  // <Custom additional code>
  public DataTree<object> Grouping(DataTree<object> input, int xDim, int yDim, int grouping)
  {
    DataTree<object> grouped = new DataTree<object>();

    for (int i = 0; i <= xDim; i++)
    {
      int ind = 0;
      int counter = 0;

      for (int j = 0; j <= yDim; j++)
      {
        int[] getPath = {j, i};
        int[] setPath = {ind, i};
        object item = input.Branch(new GH_Path(getPath))[0];
        grouped.Add(item, new GH_Path(setPath));
        counter++;
        if (counter > grouping)
        {
          ind++;
          counter = 0;
        }
      }
    }
    return grouped;
  }


  //Return a BoundingBox that contains all the geometry you are about to draw.
  public override BoundingBox ClippingBox
  {
          get
          {
            return BoundingBox.Empty;
          }
        }

        //Draw all meshes in this method.
        public override void DrawViewportMeshes(IGH_PreviewArgs args)
        {
        }

        //Draw all wires and points in this method.
        public override void DrawViewportWires(IGH_PreviewArgs args)
        {
        }

        // </Custom additional code>
      }
```