Tampere University

Taneli Koukonen

AN EXAMINATION OF A QUADCOPTER DRONE FLOW STRUCTURE AND NOISE USING QUANTUM LATTICE BOLTZ-MANN METHOD

Master of Science Thesis
Faculty of Engineering and Natural Sciences
Examiners: Dr. Raoul Yondo Mine
Prof. Kari Koskinen
July 2025

ABSTRACT

Taneli Koukonen: An examination of a quadcopter drone flow structure and noise using quantum lattice boltzmann method

Master of Science Thesis

Tampere University

Master's Degree Program in Mechanical Engineering

July 2025

The drone industry has shown great development in recent years. To maintain this development, drones need to be studied further. Development must also occur in the ways and methods of developing drones. In this work, we will examine the airflow and acoustics around the DJI Phantom 3 quadcopter drone.

The work investigates the differences between a drone in forward movement and a drone in stationary hovering. Vortices and other aerodynamic effects are studied from these two cases. Flying a quadcopter drone invariably produces sound, and this can be reduced using various methods. To utilize these methods, it is necessary to investigate how and why sound is generated. Ansys Fluent has been utilized for sound and flow analysis.

The theoretical part of the work introduces traditional RANS Navier-Stokes method and different models in it. The theoretical part also introduces a new QLBM simulation method that utilizes quantum computing. It is noted that this simulation method is quite new and is not widely used for large or complex simulations.

Simulations have been performed for a full drone using traditional RANS simulation in Ansys Fluent. It is noted that the most energy-intensive vortices are formed at the blade tips but also near the fuselage as the blade passes it. The aeroacoustics is affected by the periodic sound from the propeller rotation and the broadband sound from the background noise. A moving drone produces higher sound pressure levels than a stationary one, except for the sound in the direction directly opposite to the movement.

At the start of this research, it was planned to use QLBM for higher velocities, but due to the limits of QLBM, it was used only for very low speeds. Simulations made using the quantum emulator are compared with the traditional method performed on an airfoil. It is noted that results are similar and that there is a large difference in simulation times, which is due to using a quantum emulator. However, a direct conclusion cannot be made that the methods are equally accurate on a larger scale, since the simulations were made at very low flow speeds.

Keywords: Quadcopter, drone, aerodynamics, aeroacoustics, CFD, QLBM

TIIVISTELMÄ

Taneli Koukonen: Tutkielma neliroottori dronen virtausrakenteesta ja äänestä hyödyntyäen kvantti lattice Boltzmann menetelmää

Diplomityö

Tampereen yliopisto

Konetekniikan diplomi-insinöörin tutkinto-ohjelma

Heinäkuu 2025

Drone-teollisuus on osoittanut voimakasta kehitystä viime vuosina. Tämän kehityksen ylläpitämiseksi droneja on tutkittava. Kehitystä on tapahduttava myös tavoissa ja menetelmissä tutkia droneja. Tässä työssä tarkastellaan DJI Phantom 3 neliroottori dronen ympärillä tapahtuvaa ilmavirtausta ja akustiikkaa.

Työssä tutkitaan eroja eteenpäin liikkuvan ja paikallaan leijuvan dronen välillä. Pyörteitä ja muita aerodynaamisia vaikutuksia tutkitaan näiden kahden tapauksen välillä. Nelikopterin lennättäminen tuottaa väistämättä ääntä, mutta sitä voidaan pienentää useilla menetelmillä. Näiden menetelmien hyödyntäminen edellyttää ensin äänen tutkimista, miten ja miksi ääni syntyy. Ansys Fluentia on käytetty ääni- ja virtaus analyyseihin.

Työn teoreettisessa osassa esitellään perinteinen RANS Navier-Stokes menetelmä ja siihen sisältyvät mallit. Teoreettisessa osassa esitellään myös uusi QLBM-simulointimenetelmä, joka hyödyntää kvanttilaskentaa. On huomattava, että tämä simulointimenetelmä on melko uusi eikä sitä käytetä vielä laajemmin suurissa tai monimutkaisissa simulaatioissa.

Simuloinnit on suoritettu kokonaiselle dronelle perinteisellä RANS-simuloinnilla. Huomataan että energia tiheimmät pyörteet muodostuvat siiven kärjistä mutta myös lähellä runkoa siiven ohittaessa sen. Aeroakustiikkaan vaikuttavat jaksollinen ääni propellin pyörinnästä ja jatkuva ääni taustamelusta. Liikkuva drone tuottaa korkeampia äänipainetasoja kuin paikallaan leijuva drone, lukuun ottamatta liikkeen vastakkaisessa suunnassa.

Tutkimuksen alussa tarkoitus oli käyttää QLBM:ää suuremmilla nopeuksilla, mutta QLBM:n rajoitusten vuoksi sitä käytettiin vain hyvin pienillä nopeuksilla. Kvanttiemulaattorilla tehtyjä simulaatioita verrataan perinteiseen simulointiin 2D siivelle. Todetaan, että tulokset ovat samankaltaisia ja että, simulaatio ajoissa on suuria eroja, jotka selittyvät emulaattorin käytöllä. Suoraa johtopäätöstä siitä, että menetelmät ovat yhtä tarkkoja suuremmassa mittakaavassa, ei kuitenkaan voida tehdä, koska simulaatiot tehtiin hyvin pienillä virtausnopeuksilla.

Avainsanat: Neliroottori, drone, aerodynamiikka, aeroakustiikka, CFD, QLBM

USE OF ARTIFICIAL INTELLIGENCE IN THIS THESIS

In this thesis, artificial intelligence applications have been used:

☒ No

☐ Yes

I am aware that I am fully responsible for the entire content of my thesis, including the parts where AI has been used, and I accept responsibility for any potential violations of ethical guidelines.

PREFACE

This master thesis was conducted out of interest in drones and computational fluid dynamic simulations. The work included quantum simulation, a new field with little prior knowledge. This was not easy work to do, and many new things were learned.

Special thanks to supervisor Raoul for helping to find a subject for my thesis and with help during the work. But also, for teaching and sharing a lot of knowledge during courses. It was interesting to add the quantum simulations to the work, so thanks for the company and Valtteri for giving the opportunity and help for this.

Finally, thanks to friends and family for the support during this work and through my studies. Taking breaks from my studies to focus on something completely different made a significant difference, allowing me to return with refreshed energy.

Tampere, 28.7.2025

Taneli Koukonen

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AOA	Angle of attack
BPF	Blade passing frequency
BSL	Baseline
CFD	Computational fluid dynamics
FDM	Finite difference method
FEM	Finite element method
FVM	Finite volume method
FW&H	Ffowcs Williams & Hawking
HPC	High performance computing
ISA	International standard atmosphere
LBE	Lattice Boltzmann equation
LBGK	Lattice Bhatnagar-Gross-Krook
LBM	Lattice Boltzmann method
LES	Large-eddy simulation
LGA	Lattice gas automata
MRF	Multi reference frame
MTOM	Maximum take-off mass
NS	Navier stokes
NSE	Navier stokes equation
OSPL	Overall sound pressure level
PISO	Pressure-implicit with splitting of operators
QLBM	Quantum lattice Boltzmann method
RANS	Reynold-average Navier-Stokes
SDK	Software development kit
SIMPLE	Semi-implicit method for pressure-linked equations
SIMPLEC	Semi-implicit method for pressure-linked equations coupled
SPL	Sound pressure level
SRT	Single relaxation time
SST	Shear-Stress transport
UAV	Unmanned aerial vehicle
URF	Under-relaxation factor
VTOL	Vertical take-off and landing

1 INTRODUCTION

This master's thesis explores drone airflow fields and aeronautics from two distinct perspectives. The first one is a widely used classical CFD method (Versteeg and Malalasekera, 2007), and the second is a newer method to solve NSE using QLBM (Budinski, 2022; Shinde et al., 2025).

Simulating airflow over and around different objects has become more feasible and faster in recent years. The development of HPCs in recent years has opened the possibility to simulate complex cases using CFD that would not have been possible with basic computers. In this work simulating airflow around a quadcopter using a mesh with a large number of elements takes quite a lot of simulation time even with HPC. Many companies offer the possibility to use HPC. One of these kinds of companies is CSC, located in Finland. One of the HPC clusters they have is the Mahti supercomputer. This supercomputer is used to solve CFD case in this study.

Quantum computers might be the next step simulating more complex fluid flow cases more efficiently. There is a lot of research done on quantum computers and different applications with those. Quanscient is a company that is developing a simulation platform that uses QLBM. In this work their proof of concept is used to simulate airflow over an airfoil.

The drone industry has shown rapid development in the last decade, and it does not show signs of slowing down. Studying the flow field and aeroacoustics of drones opens the possibility to develop drones even further. For the traditional CFD problem, the DJI Phantom 3 quadcopter is studied. This is quite an old model of DJI drone, but it does not influence this study because there is no major effect on what kind of quadcopter is used. With traditional CFD, two cases are studied: one where the drone is in stationary hovering and one where the drone is in forward movement. QLBM is the alternative simulation method for classical CFD, in this study airflow over an airfoil is simulated with it.

Work is divided into three parts. The first part will be devoted to conducting a literature review about CFD simulations, LBM, and QLBM. Different solving algorithms and turbulence models are studied so that the most sufficient can be chosen to be used in this

work. In the scope of this thesis, research is done with LBE and QLBM, and the basics of these are represented in this work. This work also explains some fundamentals of quantum computers, which are necessary to understand QLBM.

The second part consists of traditional simulations of aerodynamics and aeroacoustics with Ansys Fluent using Mahti supercomputer for full drone. QLBM simulations are made with an open source qiskit quantum emulator. With quantum simulation, there are still some limitations due to the noisy nature of early-stage quantum devices. QLBM simulation is limited to the 2D case, where simulations are conducted for airfoil at 95% spanwise location. This location was chosen based on the fact that in 3D simulations vortices form near the tip. During simulations it was found that only very low velocities can be simulated, therefore the formation of vortices can't be seen.

The final part involves analysis of the results from both simulations. Where the two classical simulation cases are compared and evaluated. QLBM simulation is compared to traditional CFD simulation with similar airfoil simulation.

The main research problem in this study is to investigate ways in which flow and noise around a drone, in hover and forward movement, can be simulated with traditional CFD. And how can QLBM be used at the moment, and what are the requirements.

The research problem has been divided into three sub-questions. With these research questions, the problem to solve can be better understood. The questions are

1. What affects the aerodynamics and aeroacoustics of a quadcopter?
2. How can the QLBM be used at the moment simulating aerodynamic and aeroacoustics?
3. How does the QLBM compare to CFD simulations, and what are the limitations of current quantum computing?

Aerodynamics and aeroacoustics are studied briefly in the literature review part. This section introduces the aerodynamic features of the quadcopter. Main categories for aeroacoustics noise sources are found to be periodic and broad band and what are the main source for these. Simulations gives good understanding about what affects the aerodynamics and aeroacoustics. Simulating the stationary hovering case and the forward movement gives an understanding of what the differences in aerodynamics and aeroacoustics are between those two cases. The major source for vortices is the propeller tips, where the higher velocity on the upper side of the wing meets the lower velocity on the lower side of the wing. At the trailing edge, these velocity differences combine and form vorticities. Aeroacoustics simulation shows the locations for higher sound pressure levels

around the drone. It is noted that forward motion increases the overall sound pressure levels. This same phenomenon is also seen in other studies. Vortices forming and shedding from propeller tips are a major source of noise, along with periodic noise.

QLBM is studied in literature review and with simulations. Different studies shows that there is great potential using QLBM for CFD simulations. And that lots of research is done in this field. At the moment the possibility to simulate CFD problems is in 2D, but as it is seen during the QLMB simulations, it is still very limited and could only be used with low Reynolds number flows. Using a higher Reynolds number increases the number of needed qubits in simulation but also added some boundary errors. One of the limitations for using higher velocity is the simulation time as quantum emulator is used. Emulators simulate quantum computers and as the velocity is increased the number of qubits needed for the simulation is increased. Due to the late state when QLBM simulations was able to be started it was decided to limit simulations for lower velocities.

The QLBM proof of concept used in this study proved to be able to simulate similar results as traditional CFD with very low velocities. Aerodynamic results, pressure and velocity fields are shown around airfoil for both simulation methods and results are compared. Using low velocity and Reynolds number the flow field is laminar and therefore much simpler to model. It is noted that for future simulations higher velocity should be used to get better comparison. As mentioned, QLBM simulations are limited to 2D cases, indicating that simulating complex geometries is still quite far away. To effectively simulate complex geometries, a significantly larger number of qubits is required, making it difficult or even impossible for emulators to handle it. It has been demonstrated that representing a large number of qubits in classical bits brings challenges as the number of bits gets to quadrillions or higher. Quantum computers with more qubits are continuously being developed. However, simulating complex geometries in 3D would need quantum computers larger than those currently available.

2 LITERATURE REVIEW

This chapter provides a literature review of drones. Different simulation methods are explained, those that are currently in use and those that are not yet widely used.

2.1 Drones

Unmanned aerial vehicles (UAVs) commonly drones, were developed at the beginning for military use, just like aircrafts were developed for military use at the beginning. Interest in using drones for different applications has grown in the past years. Aerial photography is one of the many examples where it is much cheaper to take photos or videos of different places with a drone than it would be with a helicopter.

The drone industry has shown quite rapid growth in the drone market, starting from the early 2010s to the end of the decade. In the 2020s, the growth rate has decreased but is estimated to be growing at a decent rate to the end of the 2020s (Nonami, 2025). In recent years, drones used by militaries have also grown and will grow more in the coming years due to the investments done in military defense.

2.1.1 Flying vehicles

Flying machines are divided into heavier-than-air and lighter-than-air vehicles. Lighter-than-air vehicles float in the air, and altitude is maintained by having lighter gas inside than the surrounding air is. Hot air balloons serve as an example of this type of machine, controlling the density inside by heating the air. Heavier-than-air vehicles need some kind of other method to produce lift, rotating propellers or wings. Airplanes and helicopters are examples of these kinds of vehicles. Nowadays, heavier-than-air is the most common type of flying vehicle, and these are subdivided into more specific types. Airplanes are the type of fixed-wing vehicle that produces lift via pressure difference on the wing, where average pressure on the upper side of the wing is lower than it is on the underside. Pressure difference is achieved with different wing designs and airflow passing around that wing. Rotary-wing vehicles produce lift by rotating propellers where airflow passes around the propeller, generating a pressure difference at the propeller. These kinds of vehicles can stay in a stationary hovering position.

The most common types of flying vehicles are airplanes. Fixed-wing airplanes are used in commercial travel and transporting goods but also in military defense. Airplanes have

long-range capabilities with high-speed operation. Gliders are fixed-wing airplanes that have no engine. Those need another airplane to get to the initial speed and altitude. These are mostly in sport use. Helicopters are one type of rotary-wing vehicle. Some of the applications are commercial transport, military defense, or medical needs. With its vertical takeoff and hovering near the ground, it can operate in challenging places and in remote locations where airplanes cannot land. Drones are similar to helicopters in terms of lift production and are subdivided into multiple categories.

2.1.2 Drone types

There are different methods to categorize drones. For instance, one can classify drones based on specific design features, power sources, size, or their intended purpose. Some of the most common categories used to categorize drones are single-rotor Figure 1c and multi-rotor Figure 1a with vertical take-off, fixed-wing Figure 1b with horizontal take-off, and hybrid VTOL (vertical take-off and landing) drones Figure 1d with the capability of vertical take-off but operating after take-off like fixed-wing.

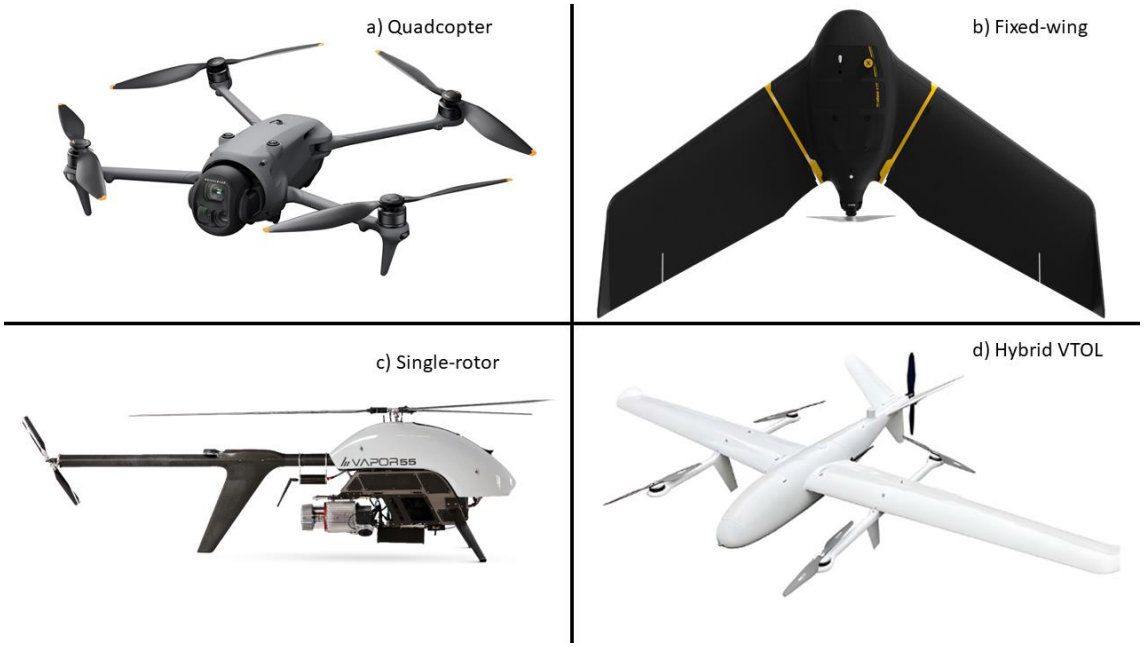

Figure 1. *Drone types a) DJI Mavic 4 Pro (DJI Official, 2025a) b) senseFly eBee X (AgEagle Aerial Systems Inc., 2025) c) Vapor 55 UAV Helicopter (Skyline UAV, 2025) d) GAOTek VTOL 101 (GAOTek, 2025).*

In these four categories, each type of drone has some advantages and disadvantages, as shown by (Kovalev et al., 2019). Single-rotor drones have one main rotor to produce lift and can be described as miniature helicopters. Even though it is called a single-rotor drone, it also has a second rotor in the back to control rotation around the main rotor. These types of drones are used in high-payload applications. Multi-rotor drones have multiple propellers and motors to produce a lift. These drones take off vertically and are

controlled by changing the rotational speed of the propellers relative to each other. Photographers, inspectors, and recreational flyers utilize these drones. This type of drone is light and simple to use, but with its lightness comes the disadvantage that it's sensitive to wind.

A fixed-wing drone is a type of drone that has wings to produce lift and propellers to produce thrust. It can have one or multiple propellers, and the location of the propellers is not specified. These drones find their use in mapping, surveying, and other monitoring applications. Used also in military applications.

A hybrid (VTOL) drone combines the fixed-wing and multi-rotor drone features. It has propellers to produce upwards thrust but also wings to produce lift when gaining enough speed. Transition between these two modes happens after takeoff or before landing.

Other categories for drones are the size, for example, nano, small, medium, large, and heavy-lift drones. Or by use case, and these use cases can be agriculture, delivery, inspection, photography, racing, and surveying drones.

Using drones in different applications is regulated in the EU area. When drone weight is over 250 g or it has a camera, the operator needs to register the drone (EU2019/945, 2020). In the delegated regulation (EU2019/945, 2020) drones are divided into seven different categories, one of the criteria is maximum take-off mass (MTOM). When looking at MTOM, the category's drones under 250 g fit in category C0, drones under 900 g in C1, and drones under 4 kg in C2. But the MTOM is not the only factor to determine the category. Categories C1 and C2 have a maximum decibel limit that depends on MTOM.

2.1.3 Quadcopter

A quadcopter is a multi-rotor drone that has 4 propellers. Two of these propellers rotate clockwise and two counterclockwise. With this arrangement, the drone's maneuverability and control are achievable. Accelerating the motor counterclockwise causes clockwise counterforce to the drone. Accelerating two motors clockwise and two counterclockwise, opposing forces cancel each other out, and each propeller generates an equal amount of thrust.

The drone rotates around three axes, referred to as yaw, pitch, and roll. Yaw can be achieved by increasing or decreasing the two motors that rotate in the same direction. These changes will also cause drones to reduce or increase overall thrust. By accelerating two motors on the same side roll is achieved. Pitch is achieved by accelerating two motors on the front or back.

The DJI Phantom 3 quadcopter was chosen for this investigation. This quadcopter was released in April 2015 (DJI Official, 2015). Different variations have been released after the first version with different camera options and with slight improvements. For this study, the different camera options and the shape of the camera are not relevant. Introducing the mesh and geometry later, the camera and landing gear are removed.

Some relevant specs of the drone are introduced. The specs are gathered from the technical data sheet and from the DJI website (DJI Official, 2025b). Operating weight is 1,216 g. The battery is a 4S LiPo with a capacity of 4480 mAh and a voltage of 15.2 V. Motors are 920 KV, where KV value means RPM/Volt. From that, the maximum RPM could be calculated, but a better estimate for the maximum RPM of the motor and propeller is the manufacturer given value of 1250 rad/s or 12000 RPM (Paz et al., 2020).

2.2 Aerodynamic noise

Aerodynamic noise is the sound produced when a solid object or moving fluid interacts with the surrounding air. Sound that is generated by rotating propellers is separated into vortex and rotational components. The periodic or commonly rotational noise describes all the sounds that have the same frequencies as the blade passing frequency (BPF). Broad band noise describes sounds produced by fluctuating pressure fields caused by vortices that are shed from the trailing edge and tips of the propeller, as well as by turbulence present in the airflow. (Marte and Kurtz, 1970, p. 3)

Aerodynamic noises generated by rotating propellers are shown in Figure 2, where noise is subdivided into multiple categories. With drones some of the components has significantly smaller role for noise generation when comparing to helicopters where rotational speeds are much higher.

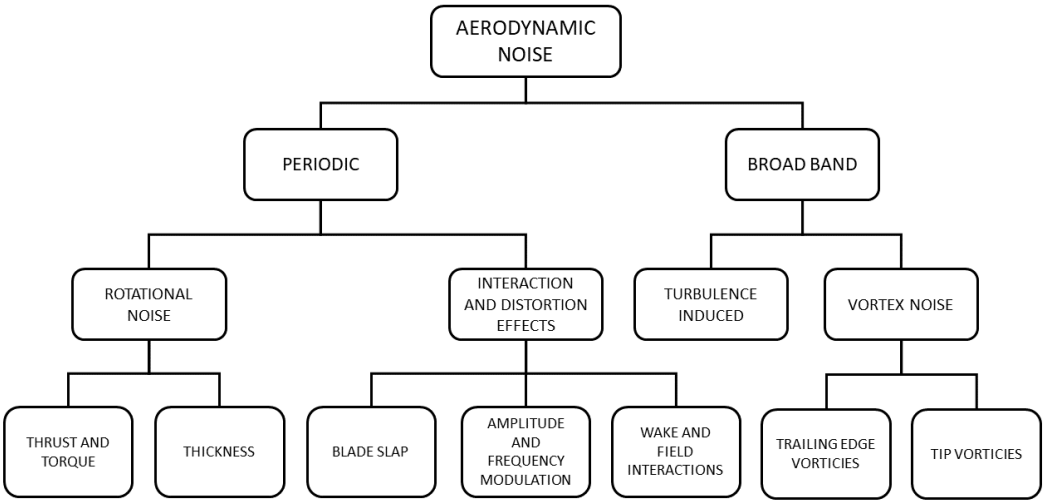

Figure 2. *Sources of aerodynamic noise are shown by (Marte and Kurtz, 1970, p. 3).*

Two of the main categories are periodic and broadband noise and these are explained in the next chapters.

2.2.1 Periodic noise

Rotating propellers distribute pressure when they're moving relative to the surrounding air. Pressure distribution around the propeller blade causes a force on the blade. This force vector can be resolved into a thrust vector and a torque vector. The area that the rotating propeller covers is called a disc. On the disc, a rotating field appears as an oscillating pressure at a fixed point. The frequency of this oscillating pressure is the frequency that the blade passes a fixed point. This value can be calculated from the rotational frequency of the propeller multiplied by the number of blades on the propeller. (Marte and Kurtz, 1970, pp. 3–4)

The propeller generates a thickness noise when it moves air near the disc. The rate of the mass that the propeller moves depends on the blade profile, the angle of the profile relative to the moved air, and the speed at that point.

Noise generated by interaction and distortion effects is periodic depending on the rotational speed of the propeller. Blade slap is an impulsive type of noise characterized by high-amplitude and periodic sound components along with vortex noise generated by sudden fluctuating forces acting on the propeller. The kinds of forces include stalling and unstalling, shock waves, and blade-vortex interaction. These conditions occur when blades pass through tip vortexes or unsteady wakes. Shock waves may occur at the tip of the propeller, where the velocity is the highest. This phenomenon arises when the

propeller's rotating speed approaches the speed of sound, a condition typically not seen in drone propellers. (Marte and Kurtz, 1970, p. 4)

Amplitude and frequency modulation are noise generated due to the changes in blade loading and incoming flow conditions. The modulation of noise characteristics, intensity, and frequency creates complex acoustic footprints.

Wake and field interaction is noise generated by rotors. Wake distorts airflow entering the neighboring rotor, leading to unsteady loading and extra periodic noise components.

2.2.2 Broad band noise

Turbulence-induced noise, also called broad band noise, arises from unsteady random flows. At turbulent boundary layers, rotor blades generate random pressure fluctuations that radiate noise. Turbulence shed from a rotor passes through another rotor that adds broadband noise, which is called turbulent wake interaction. (Marte and Kurtz, 1970, pp. 4–5)

Formation and shedding of vortices past the propeller at the flow field are described as vortex noise introduced by (Marte and Kurtz, 1970, p. 4). With a rotating propeller, velocity varies along the spanwise location. Vortices are shed to the vortex street, which depends on the cylinder's diameter and the flow velocity.

Tip vortices are generated at the tip of the blade that has a strong pressure gradient with air wrapping around the blade. Shedding off these vortices is a major noise contribution, particularly at higher rotational speeds. Also, at the trailing edge of the propeller, small-scale vortices are forming, contributing further to the broad band noise. Vortex shedding frequency depends on local flow velocity as well as the geometry of the blade. (Marte and Kurtz, 1970, pp. 2–4)

2.3 Aerodynamic features

A quadcopter generates a lift by rotating propellers. These propellers resemble revolving wings. The angle of attack of the drone propeller airfoil at some spanwise location is almost constant. Accelerating the propeller causes the blade to twist, which affects the angle of attack (AOA), but this twist is not significant. With a higher angle of attack, lift increases up to a critical point where flow separates from the airfoil and it stalls. With a rotating propeller, the oncoming flow is perpendicular to the rotating axis, and therefore the angle of attack is equal to the blade pitch. A higher pitch generates more lift, as with the AOA, but also increases drag and power consumption. The generation of lift also

influences the speed of the rotating propeller and the characteristics of the air, including density. (Anderson, 2016, pp. 20 & 512–514)

Controlling drone movement is based on pitch, roll, and yaw. Pitch rotation is controlled by increasing the rotational velocity of the drone's back propellers and decreasing the front propellers relative to the back propellers and vice versa. Forward movement is achieved by pitching the drone's nose down and slightly increasing lift to maintain altitude. Higher rotational speed on the back propeller will tilt the drone forward and generate forward movement. Roll controls tilt from left to right and is similar to pitch difference in that the rotation axis is in a different direction, and therefore movement is also perpendicular to pitch rotation. Yaw rotates the drone around an axis upwards from center. Quadcopters have four propellers diagonally opposite to each other. Two propellers rotate clockwise and the other two rotate counterclockwise to cancel net torque. To generate yaw movement, the rotational speed of the propellers that are rotating in the same direction is increased, and the other two are decreased. With this, constant lift is maintained and yaw movement is achieved. (Stephan, 2025, pp. 15–17)

The lift produced from the propellers affects the proximity of the ground. In this study, the drone is set to be close to the ground, which simulates takeoff, landing, or moving close to the ground. Ground effect increases the overall lift generated by the propellers and decreases the drag. Ground effect is studied by (Paz et al., 2020) where the stationary case and moving over an obstacle were studied. The stationary case shows a lift force increase of over 7% 10 cm above the ground, and when the DJI Phantom 3 drone is over 50 cm above ground, the ground effect is negligible.

2.4 Computational Fluid dynamics

Computational fluid dynamics (CFD) is described as a method to predict fluid flow in different applications. These vary from small to large and from simple to very complex. The smallest and simplest cases can be calculated with home laptops. The need for computing power increases as the shapes become more complex and the number of equations solved during simulation increases. In recent years, the use of HPC (high-performance computing) has grown, and the use of CFD in more complex geometries has become feasible. Also, the use of HPC in simpler cases makes CFD computing much faster.

Fluid flow is modeled using mathematical equations. These equations are then solved numerically, and the result is an approximate solution to the given problem. The equations representing the CFD problem are called governing equations.

The mathematical governing equations that form the CFD are based on the principle that a fluid's physical properties remain conserved. Conservation laws of physics in the fluid are summarized with three laws. Mass conservation means that the total mass remains constant. Momentum conservation, as described by Newton's second law, means that any change in a particle's momentum is caused by the total net force affecting it. Energy conservation means that the change in energy of a fluid particle is the sum of the work done to that particle and the total head added (first law of thermodynamics). (Versteeg and Malalasekera, 2007, p. 10-24)

Mass increase rate in an element is the net mass change in the element relative to time. The rate of increase is shown in equation (1)

$$\frac{\partial}{\partial t}(\rho\delta x\delta y\delta z) = \frac{\partial \rho}{\partial t}\delta x\delta y\delta z \tag{1}$$

where element volume is multiplied by the partial derivative of density with respect to time. Equating the rate of mass increase to the element faces, the equation (2) is obtained

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} = 0 \tag{2}$$

where velocity components are added. Equation (2) is commonly rewritten in more compact vector notation

$$\frac{\partial \rho}{\partial t} + \mathrm{div}(\rho \mathbf{u}) = 0\,. \tag{3}$$

Equation (3) is the mass conservation equation for compressible three-dimensional fluids. The first term is the change of density in time, and the second term is the group of velocity equations that determines the net mass flow out of the system.

The momentum equation is divided into three components. These components are commonly called *x*-, *y*-, and *z*-components. Setting the *x*-momentum equal to the forces on *x*-direction, equation (4) is obtained

$$\rho\frac{Du}{Dt} = \frac{\partial(-p + \tau_{xx})}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + S_{Mx} \tag{4}$$

and *y*-component

$$\rho \frac{Dv}{Dt} = \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \left(-p + \tau_{yy}\right)}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + S_{My} \tag{5}$$

and *z*-component

$$\rho \frac{Dw}{Dt} = \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial (-p + \tau_{zz})}{\partial z} + S_{Mz}. \tag{6}$$

In the momentum equations, last S_M term is the source term to different directions, where surface stress effects are considered. Partial derivates have a term τ that is the stress component in different directions along the element surface.

The governing equations include unknown terms, which represent the components of viscous stress. These components are expressed by the functions of deformation rate or local strain rate. And the local deformation rate is composed of the volumetric deformation rate and the linear deformation rate. Newton's law of viscosity in three dimensions contains nine components of viscous stress, of which six are independent. By substitution of shear stress into the momentum components, the Navier-Stokes (NS) equations are yielded. Writing these equations in more useful form equations are obtained in the x-direction

$$\rho \frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \mathrm{div}(\mu \,\mathrm{grad}\, u) + S_{Mx} \tag{7}$$

and to y-direction

$$\rho \frac{Dv}{Dt} = -\frac{\partial p}{\partial y} + \mathrm{div}(\mu \,\mathrm{grad}\, v) + S_{My} \tag{8}$$

and to z-direction

$$\rho \frac{Dw}{Dt} = -\frac{\partial p}{\partial z} + \mathrm{div}(\mu \,\mathrm{grad}\, w) + S_{Mz}. \tag{9}$$

These NS equations (7), (8) and (9) are used to discretize the model.

2.4.1 Methodology

Solving a CFD problem starts with generating the geometry of the object that is being investigated. Geometry should not be more complex than is necessary. In this study, geometry and mesh are the same as those used by (Paz et al., 2020) and are therefore not modeled for this study.

When examining the fluid surrounding the geometry, it's important to consider the distance from the geometry at which the flow analysis is required. Making the fluid domain too small gives inaccurate results, and making it too large takes unnecessary computing power, and therefore it takes longer to solve the problem. For the size of the fluid domain, there are multiple factors affecting it, depending on what type of geometry and problem are examined. In cases where walls or other objects are not affecting the flow field, boundaries need to be set far enough. And in studies where ground effect or some other object effect on the flow field boundaries are set to walls and distances corresponding to the desired value. The outlet side of the flow is most often set to be further away from geometry since vortices and other interesting flow phenomena appear after the object. This kind of approach is used when the object is only examined in flow mainly in one direction. In this study, the flow domain is cylinder-shaped, so the inlet and outlet sides are equally as far from the drone. This approach gives the freedom to change flow direction to any direction and the outlet side being still far enough. Generating a fluid domain, it is also important to name areas that are important for solving the problem. These areas include the inlets and outlets of the flow, as well as geometric regions. The more accurate the naming is, setting boundary conditions and analyzing different areas, later becomes easier.

Governing equations that are solved need to be defined depending on how the problem is solved. The flow can be set to be inviscid, which simplifies the equations, or viscous flow in most real-world cases. Also, the turbulent model must be determined. Another important part is to determine the boundary conditions inlets, outlets and walls of the geometry. Starting with the simulations, residuals are determined. That show if the solutions settle to a constant value or not. The number of iterations that the equations are iterated is chosen, and after that, the simulation stops if the residuals are not achieved. The final step, known as postprocessing, displays the desired results for examination and for evaluating these based on their quality.

Spatial discretization of the NS (Navier-Stokes) equations starts with numerical approximation of the equations (7), (8), and (9). Three of the main categories for spatial discretization are the Finite Difference Method (FDM), the Finite Volume Method (FVM), and the Finite Element Method (FEM). These are all based on grids to discretize the governing equations. The grid types are structured and unstructured grids. With a structured grid, every grid point is identified by indexing them, for example, (i, j, k) and neighbor grid points (i, j+1, k). Unstructured grids are different in that grid points have no particular order, and therefore neighboring points or grids cannot be directly identified with indexes. One grid can be 8 and its neighbor 125. A structured grid has the advantage

that the indexing represents linear address space to the computer memory. This technique allows easy and quick access to neighboring grids, adding or subtracting from index value. This works well for simple geometries but becomes difficult when the geometry is more complex. The space can be divided into multiple much simpler parts to avoid one complex grid that is hard to generate. Unstructured grid advantages are that the grid can be generated more automatically even for complex geometries. The disadvantage is that solving with an unstructured grid reduces computational efficiency. (Blazek, 2015, pp. 433–454)

FDM is one of the earliest techniques to numerically solve differential equations. This method works by directly applying discretization to the differential form of the governing equations used. The key idea is that derivative flow variables are approximated by Taylor series expansion. The advantage of this method is its simplicity and ability to easily obtain high-order approximations. The method uses a structured grid, and therefore applications where this approach can be used are restricted to the fact that structured grids are mostly used on simpler geometries. Curvilinear coordinates cannot directly apply to FDM. Instead, one must use the Cartesian coordinate system and transform the governing equations to it. (Blazek, 2015, pp. 53–54)

FVM applies the conservation laws directly by using the integral expressions of the NS and Euler equations. The discretization of the governing equations begins by establishing control volumes throughout the entire flow domain. Approximation to the surface integral in equation (10)

$$\frac{\partial}{\partial t}\int_{\Omega}\vec{W}\mathrm{d}\Omega + \oint_{\partial\Omega}\left(\vec{F}_{\mathrm{c}} - \vec{F}_{\mathrm{v}}\right)\mathrm{d}S = \int_{\Omega}\vec{Q}\mathrm{d}\Omega \tag{10}$$

is made within each control volume, and where fluxes on individual faces are summed together. Control volume shape and position can be defined with respect to the grid with some different possibilities. Two of these possibilities are cell-centered, where the control volume grid lines are in line with the grid and variables are stored in the center. Second one is cell vertex, where the center of the control volume is located at the grid point and variables are therefore stored at grid points. With FVM, the main advantage is that there is no need to make any transformation between coordinate systems, and it can be used with structured and unstructured meshes easily. And makes this suitable for complex geometries. The numerical scheme conserves mass, momentum, and energy, enabling the computation of weak solutions from the governing equations. However, the entropy condition must still be satisfied when solving the Euler equations. The entropy condition

prevents unphysical features, like the expansion of shocks. FVM is very popular and widely used because of its properties. (Blazek, 2015, p. 53)

With FEM, the physical phase is divided into discrete elements, and an unstructured grid must be used. The type and requirements of the problem determine the needed number of points used at boundaries and/or inside the element itself. Degrees of freedom are calculated with the number of points used and multiplying it by the number of unknowns. Shape functions are defined and represent variations in the solutions. The FEM transforms the governing equations into an equivalent integral form. FEM's integral formulation and the fact that it uses an unstructured grid make it an attractive method for complex geometries. And this model is used in some aeroacoustics problems but also with non-Newtonian fluids. (Blazek, 2015, pp. 53–54)

In this work CFD simulation are run on Ansys Fluent where spatial discretization method is FVM. Mesh is constructed from structured grid which allows quicker access to neighboring grid elements. This makes the computations slightly quicker with the disadvantage of challenging mesh construction.

2.4.2 Solution algorithms

Solving Navier-Stokes equations in a pressure-based approach, continuity and momentum equations need to be satisfied. In a pressure-based solver, pressure is one of the unknowns rather than density, which is used in a density-based approach. Pressure-based is used commonly at low speed, and density-based is used at high Mach number (Ansys, 2025a, p. 967). As the pressure is unknown, it must be ensured that the pressure field is consistent with the velocity field, so the mass is conserved. This is ensured with different solution algorithms like SIMPLE, SIMPLEC, PISO, or COUPLED in an iterative process.

SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) starts with an initial guess of velocity and pressure that the user defines. Solve the momentum equations where the velocity field is calculated from pressure. The pressure correction equation is built based on continuity requirements. This ensures that that velocity field is corrected to satisfy mass conservation. Pressure and velocity fields are adjusted with the corrections. These steps are repeated until the case is converged or user-specified criteria are satisfied. (Versteeg and Malalasekera, 2007, p. 186-189)

SIMPLEC (SIMPLE-Consistent) is based on the SIMPLE method with some differences. It uses fewer approximations in the velocity correction step. Larger under-relaxation factors can be used because the correction is formulated differently, and therefore convergence can be achieved in fewer iterations. (Versteeg and Malalasekera, 2007, p. 193)

PISO (Pressure-Implicit with Splitting of Operators) starts with prediction to solve momentum equations using the current pressure field to get the initial velocity field. From that pressure correction, equations are solved based on continuity to adjust velocity. There is also an additional correction pass to further refine velocity and pressure consistency. Steps are repeated. PISO uses at least two pressure-correction loops per iteration. This improves consistency between velocity and pressure without excessive under-relaxation. This method is effective with transient time-dependent problems, achieving good pressure-velocity coupling in fewer iterations. With the steady state method, additional correction is unnecessary because correction is done either way in the next iteration. (Versteeg and Malalasekera, 2007, pp. 193–195)

COUPLED solver forms and solves a combined system of equations for pressure and velocity in a single matrix system instead of separately solving the momentum equations and correction equations. This means better stability and, in some cases, faster convergence. This method requires more computational resources, like memory, because the matrix solved in each iteration is significantly bigger. (Ansys, 2025a, pp. 995–996)

Underrelaxation is used to adjust the correction made to the guessed value. URF (underrelaxation factor) has some initial value, and these values depend on the chosen algorithm. Every iteration correction value is calculated and is multiplied by the chosen URF value. This product is added to the previous iteration corrected value. Using a URF value of 0.5 corrects the guessed value by half of the calculated value. (Versteeg and Malalasekera, 2007, p. 189)

For the simulations in Ansys Fluent it is chosen to use COUPLED for steady state simulation as it seems to convergence faster with the disadvantage of using more memory. Memory use is not found to be problem in HPC. For transient state simulation PISO is chosen at the beginning to be used, as it adds an extra pressure correction for each iteration. With testing it found that PISO does not converge the solution correctly even with lowered URF values. SIMPLEC is changed to be used where larger URF values can be used. One of the reasons that PISO does not work even with lowered URF values is that if too large time step size is used.

2.4.3 Turbulence models

Difficulties solving the governing equations occur when flow is viscous or turbulent. Turbulent flow occurs when the critical Reynolds number is achieved in the flow. In this state, flow behavior changes, and it becomes random and chaotic.

Direct numerical simulation (DNS) is a method to simulate turbulence directly from Navier-Stokes equations depending on time but is used in rather simple flow cases and at low Reynolds numbers. For sufficient spatial resolution, the needed number of grid points is quite high around $Re^{9/4}$, and also the CPU time is around Re^3. (Versteeg and Malalasekera, 2007, pp. 112–113) Therefore, this model is not used in this study.

Large-eddy simulation (LES) is an approximation model, and the idea with it is to resolve accurately the large eddies. The LES requirement for grid points is smaller than the DNS model and becomes feasible at larger Reynolds numbers. But it is still computationally very demanding. LES is suitable for studies of separated unsteady flows, aerodynamic noise, flow control strategies, or large-scale mixing. (Versteeg and Malalasekera, 2007, p. 98) The LES model is suitable for this study but was chosen not to be used.

Reynolds-averaged Navier-Stokes equations (RANS) are based on flow variables decomposing into mean and fluctuating parts. When the density is not constant, we apply density-weighted or Favre decomposition to the velocity components. This approach prevents governing equations from becoming unnecessarily complicated due to the density fluctuations. Solutions of RANS require modeling heat flux and Reynolds stresses. This method's advantages over LES include the ability to use a coarser grid and the assumption of a stationary mean solution. Compute turbulent flows using RANS equations. Reynolds stresses, scalar transport terms, and systems of mean flow equations need to be predicted. This is done with turbulence models. The most common turbulence models and the number of extra transport equations needed for each model are the mixing length model with zero extra equations, the Spalart-Allmaras model with one, the, $k-\epsilon$, $k-\omega$ and algebraic stress models with two extra equations, and the Reynolds stress model with seven extra equations (Versteeg and Malalasekera, 2007, p. 67-93).

The next parts explain the differences and determine which turbulence model is best for this study. With the mixing length model, an assumption needs to be made, which is that kinetic turbulent viscosity can be expressed by the product of two components that are turbulent length scale and turbulent velocity scale. Turbulence kinetic energy is mostly stored in large eddies. There is a strong connection between the mean flow and large eddies. If this assumption is accepted, the link between the mean flow and the largest

eddies is characterized by scale length. This works well for simple two-dimensional turbulent flows and is not, therefore, suitable to this case. (Versteeg and Malalasekera, 2007, pp. 69–72)

The Spalart-Allmaras model contains one equation to solve the modeled transport equation. The model has been shown to be suitable for boundary layers that are affected by pressure gradients. The algebraic stress model is a simplified version of the Reynolds stress model without solving the full scale of its transport equations. (Versteeg and Malalasekera, 2007, pp. 89–90)

The $\mathrm{k}-\epsilon$ model contains two transport equations that are solved. This model has three different forms that are standard, RNG, and realizable. Different models are shown in the Ansys theory guide (Ansys, 2025a, pp. 49–60). With these models' flow needs to be considered fully turbulent and is therefore only valid for flows that are fully turbulent. This kind of assumption is not valid at the boundary layers for the flow. Differences in these models are the methods used to calculate turbulent viscosity and Prandtl numbers that are used to govern turbulent diffusion terms k and ϵ. With different turbulence models in the ϵ equation generation and destruction terms have some differences.

The RNG model and standard models are similar, but with the RNG model, some additional adjustments are made. To improve accuracy at quickly strained flows, an additional term to the equation is added. Also, the effect of swirl is included, and for the Prandtl numbers, an analytical formula is added rather than using some constant value as in the standard model. The model is also more suitable for low Reynolds number cases due to the fact that for effective viscosity differential formula derived analytically is used. However, the effectiveness of this depends on the near-wall treatment chosen. These additions from standard to RNG make this model more accurate in a wider range of flows. The realizable model has two important differences from the standard model. With this model, turbulent viscosity has an alternative formulation, and transport equations are derived by mean square vorticity fluctuating in exact equations. The name realizable comes from the fact that mathematical constraints on the Reynolds stresses are satisfied. (Ansys, 2025a, pp. 51–56)

The $\mathrm{k}-\omega$ model also consists of three different models standard, BSL, and SST and is shown in the Ansys theory guide (Ansys, 2025a, pp. 60–70). The major differences between BSL and SST to the standard model are the models used in the inner and outer boundary layers. Where standard model uses only the $\mathrm{k}-\omega$ model. BSL and SST combines the standard and the $\mathrm{k}-\epsilon$ model. With the SST model, a modified version of tur-

bulent viscosity formulation is factored in for transport effects. The standard model problem is the freestream sensitivity that can have a significant effect on the solution. The standard model is an empirical model that is based on transport equations.

The BSL model consists of a formulation more accurate of $k-\omega$ model at the near-wall regions and $k-\varepsilon$ in the far field of the walls. Refinements to the standard model are adding the standard model and the $k-\varepsilon$ model together. This model also includes in ω equation damped cross diffusion derivative term, and this has different modeling constants. Transport equations are

$$\frac{\partial}{\partial t}(\rho k)+\frac{\partial}{\partial x_i}(\rho k u_i)=\frac{\partial}{\partial x_j}\left(\Gamma_k \frac{\partial k}{\partial x_j}\right)+G_k-Y_k+S_k+G_b \tag{11}$$

and

$$\frac{\partial}{\partial t}(\rho \omega)+\frac{\partial}{\partial x_i}(\rho \omega u_i)=\frac{\partial}{\partial x_j}\left(\Gamma_\omega \frac{\partial \omega}{\partial x_j}\right)+G_\omega-Y_\omega+D_\omega+S_\omega+G_{\omega b}. \tag{12}$$

In these equations G_k is the term of generation of turbulence kinetic energy, G_ω is generation of ω. Γ_k and Γ_ω is the effective rate of diffusion of the terms *k* and ω. Y_k and Y_ω are the dissipation terms in k and ω due to turbulence. These terms are calculated. User defines the source terms by the use case which are S_k and S_ω. G_k and $G_{\omega b}$ are buoyancy terms and D_ω is cross-diffusion term.

The SST model includes all the same additions as the BSL model with added turbulence shear stress. This addition makes the model much more accurate than BSL and standard models for a wider range of different types of flows. Transport equations are the same as in the BSL model.

Considering these different models to simulate flow around a quadcopter, it seems that to get the most accurate results with RANS, the SST $k-\omega$ model should be used. With this, flow can be simulated in the far field and near walls.

2.5 Lattice Boltzmann method

The Lattice Boltzmann method (LBM) describes the dynamics of gas in mesoscopic scale and lead to fluid dynamic equations in macroscale (Krüger et al., 2016, pp. 61–63). This represents the density of particles that have a velocity function $c_i=(c_{ix},c_{iy},c_{iz})$ at a time *t* and with position *x*. Position *x*, where f_i is defined, is positioned into a square lattice that is defined in space, lattice spacing Δx determines the distance between lattices sites. Time also has an important role in f_i, where timestep Δt separates different

moments. Discrete velocities with weighting coefficients form velocity sets $\{c_i, w_i\}$. Velocity sets are represented by notation D*d*Q*q*, where D*d* is the dimension of the case (1D, 2D, or 3D) and Q*q* is the number of velocity sets used in the case. For example, the three-dimensional case with 19 sets of velocities is marked as D3Q19. LBE is found from the Boltzmann equation by discretizing it in velocity and physical space and also in time, as shown by (Zhaoli and Chang, 2013, p. 16)

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i\delta_t, t + \delta_t) - f_i(\boldsymbol{x}, t) = \Omega_i\big(f(\boldsymbol{x}, t)\big). \tag{13}$$

This equation (13) expresses that particle $f_i(x, t)$ move at some velocity c_i to its neighboring point $x + c_i\delta_t$ by timestep $t + \delta_t$. Particles are also affected by collisions expressed by collision operator Ω_i.

Instead of solving Navier Stokes equations directly, fluid density is simulated with streaming and collision process.

2.5.1 History

Evolution of the LBM starts from the lattice gas automata (LGA), which is an artificial microscopic model used to model gases. Later it was discovered that from the Boltzmann equation with some discretization, LBE could also be derived. From one perspective, LBE can be described as a fluid model, and from another perspective, as a special numerical method for solving the Boltzmann equation. Both viewpoints show that LBE is a technique that differs significantly from traditional CFD methods. (Zhaoli and Chang, 2013, p. 10)

LGA is considered the precursor of LBE, which simulates fluid flows using simple fluid models. Simulated particles are collected to model fluid on a regular lattice with specific symmetry properties. These particles undergo collision and streaming, following rules that adhere to fundamental physics laws. The core idea with LGA is that macroscopic fluid behavior is essentially the statistical collective result of the micro-dynamics of individual fluid molecules, and it is independent of the detailed characteristics of those molecules. This implies that it is possible to simulate large-scale fluid flows using a simplified micro-level model that archives the necessary physical principles. (Zhaoli and Chang, 2013, pp. 10–16)

Evolution from LGA to LBE started with the need to reduce the statistical noise shown by (Zhaoli and Chang, 2013, p. 16). Boolean variable n_i was replaced with real-valued distribution function f_i and the collision rule was kept in the same form. This new equation

effectively removes the statistical noise, although the collision operator is still rather complicated. The collision operator is approximated by a linearized one, assuming f_i is close to its equilibrium.

2.5.2 Mathematical framework

Understanding the LBE starts from equation (13), where the left-hand side's first term shows that the particle moves to its neighboring point. The second term shows the particle's original point relative to location and time, the Ω_i term is the collision term. This collision operator is quite complicated, and it can be approximated by a linearized one where the assumption is made that f_i is close to its equilibrium. Equation (13) is divided into collision (14) and streaming (15) parts,

$$f_i^{\star}(\boldsymbol{x}, t) = f_i(\boldsymbol{x}, t) + \Omega_i\big(f(\boldsymbol{x}, t)\big) \tag{14}$$

and

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i\delta_t, t + \delta_t) = f_i^{\star}(\boldsymbol{x}, t). \tag{15}$$

In these equations, the $f_i^{\star}(\boldsymbol{x}, t)$ term shows the post-collision population. With streaming equation (15), $f_i^{\star}(\boldsymbol{x}, t)$ populations stream along the direction determined by that model to the next lattice and become $f_i(\boldsymbol{x} + \boldsymbol{c}_i\delta_t, t + \delta_t)$. (Krüger et al., 2016, pp. 101–103)

One of the methods to model collision is the Lattice Bhatnagar-Gross-Krook (LBGK), which is also called single relaxation time (SRT) and can be used for low Mach number Navier-Stokes simulations, as shown by (Zhaoli and Chang, 2013, pp. 16–19). The linearized collision operator, denoted as Ω_i, takes a form

$$\Omega_i(f) = K_{ij}\left(f_j - f_j^{(eq)}\right), \tag{16}$$

in which $K_{ij} = \partial\Omega_i/\partial f_i$ represents the collision matrix. Using a linearized collision operator by the fact that $\Omega_i\big(f_i^{eq}\big) = 0$ simplifies the model. The collision matrix is defined with $K = \tau^{-1}I$, and the collision equation is therefore expressed as

$$\Omega_i(f) = -\frac{1}{\tau}\Big[f_i - f_i^{(eq)}\Big]. \tag{17}$$

The equation relaxes particles toward equilibrium f_i^{eq} at a rate that is determined by relaxation time τ. For incompressible and weakly compressible fluids, equilibrium for equation (17) is given by equation

$$f_i^{(eq)}(\boldsymbol{x},t) = w(\boldsymbol{c}_i)\rho\left(1+\frac{\boldsymbol{c}_i\cdot\boldsymbol{u}}{RT}+\frac{(\boldsymbol{c}_i\cdot\boldsymbol{u})^2}{2(RT)^2}-\frac{u^2}{2RT}\right)+O(\boldsymbol{u}^3). \quad (18)$$

Where lattice velocity is c_i, u is velocity, weighting factor w_i, *R* is the specific gas constant, and *T* is temperature. The last term is the error correction term, and due to this, only a weakly compressible regime can be calculated (Krüger et al., 2016, p. 144). With compressible flows, the energy equation needs to be solved as well. Equation (18) cannot be directly applied to compressible flows at high Mach numbers and therefore needs some more refinements that are not discussed here. (Zhaoli and Chang, 2013, pp. 198–199)

In Table 1 are shown parameters for different models where lattice speed c is set to 1.

Model	Lattice vector c_i	weight w_i	c_s^2
D1Q3	$0,\pm1$	$2/3\,,1/6$	1/3
D1Q5	$0,\pm1,\pm2$	$6/12\,,2/12,1/12$	1
D2Q7	$(0,0)$, $\left(\pm\,1/2\,,\pm\sqrt{3}/2\right)$	$1/2\,,1/12$	1/4
D2Q9	$(0,0),(\pm1,0)$, $(0,\pm1),(\pm1,\pm1)$	$4/9\,,1/9\,,1/72$	1/3
D3Q15	$(0,0,0)$, $(\pm1,0,0),(0,\pm1,0),(0,0,\pm1)$, $(\pm1,\pm1,\pm1)$	$2/9\,,1/9\,,1/72$	1/3
D3Q19	$(0,0,0)$, $(\pm1,0,0),(0,\pm1,0),(0,0,\pm1)$, $(\pm1,\pm1,0),(\pm1,0,\pm1),(0,\pm1,\pm1)$	$1/3\,,1/18\,,1/36$	1/3

Table 1. *Parameters for DdQq models (Zhaoli and Chang, 2013, p. 20).*

Figure 3 shows the airfoil shape of the drone propeller, and around that are visualized nodes and two different lattices, D2Q5 and D2Q9. In these models, the middle lattice points of zero represents no velocity condition.

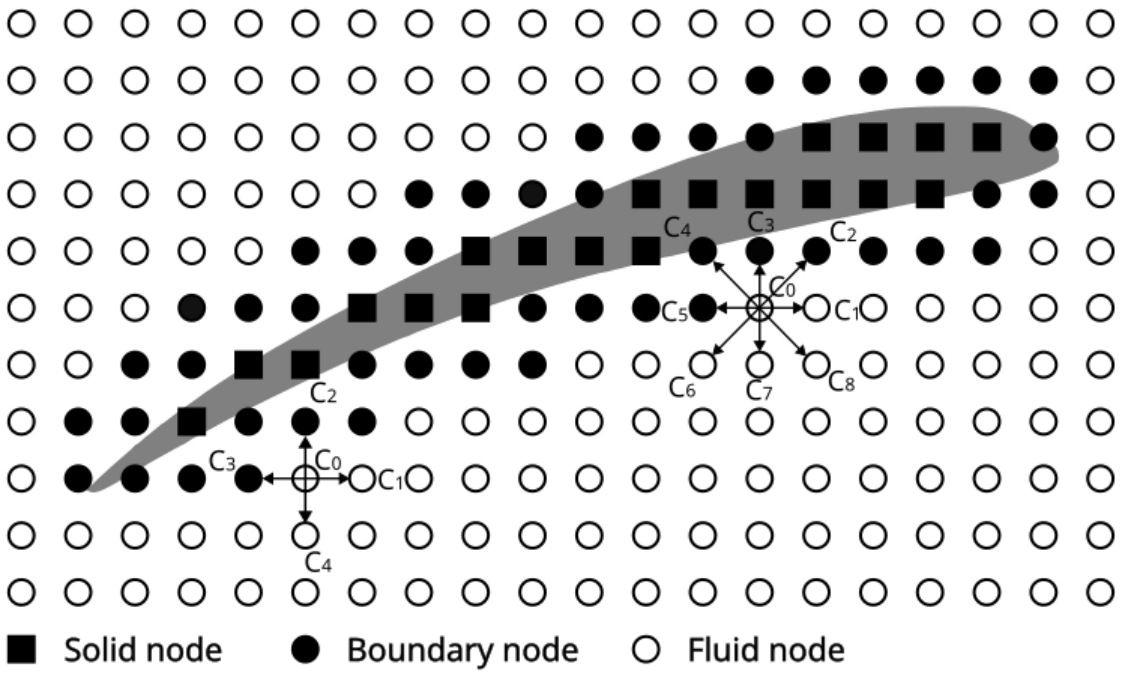

Figure 3. *Fluid, solid and boundary nodes around airfoil.*

Figure 4 shows collision and streaming near the wall. Two cases are visualized: one where the collision happens at the wall and one where it happens a couple of lattice nodes away from the wall. After collision, streaming is represented on the right side.

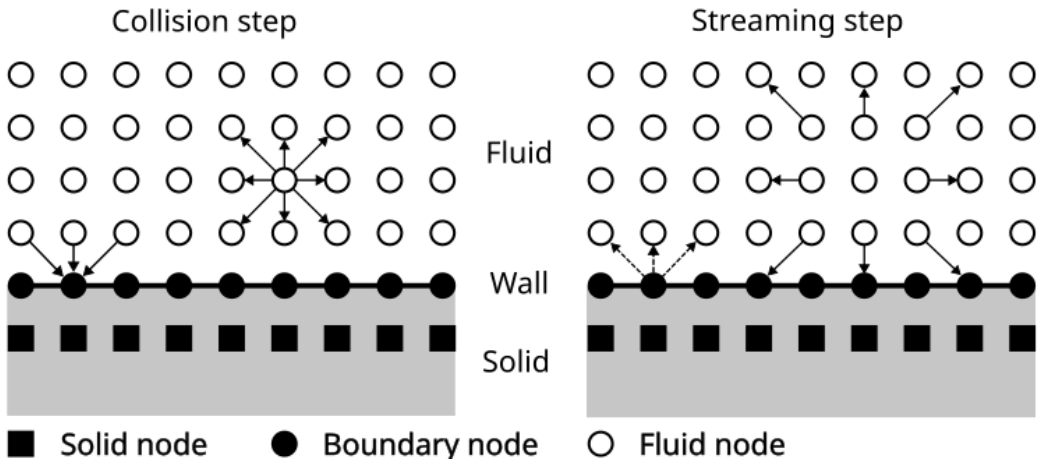

Figure 4. *Collision and streaming visualized.*

In Figure 4, particles at fluid nodes will stream to their neighboring nodes, which are indicated by solid arrows. Streaming from boundary nodes to fluid nodes, shown by dashed arrows, cannot be described directly by LBE as it can be in fluid node to fluid node. Boundary conditions need to be taken into account when propagating from a solid object into a fluid. (Krüger et al., 2016, pp. 164–166)

In collision step arrows show the direction of collision to node. Streaming step is the result of collision.

Fluid nodes are the nodes where LBM is applied and are connected to boundaries or other fluid nodes. Solid nodes are inside the solid object, and at these nodes LBM is not solved. These types of nodes are linked to boundaries or other solid nodes. Boundary nodes link these two nodes together and require special treatment. A boundary node is linked to at least one solid node and one fluid node. (Krüger et al., 2016, p. 165)

2.5.3 Numerical and computational aspects

Initial conditions are set at the beginning of the simulation and are used to achieve steady-state flow. They are more relevant in complex flows. Some of the initial conditions that are set before simulation are, for example, setting density to a constant value and setting velocity to a value that is near the steady solution velocity. Guessing the value of velocity in some cases or at some part of the case is hard, and therefore the best guess at the time is chosen. This step is done to minimize the simulation time in transient time. (Succi, 2018, pp. 271–272)

To determine boundary conditions, we need to introduce two basic classes of boundary conditions: grid-aligned and grid-cutting. Whit grid-aligned, the physical boundary aligns with the grid coordinates. These kinds of boundaries need to be straight lines or planes. With grid-cutting, the boundary can take any shape and does not need to be straight lines or a plane. With this, the boundary cuts through the lattice. These kinds of boundary conditions are more accurate but are harder to implement. Names of the different class boundary conditions are periodic, no-slip, free-slip, frictional slip, sliding walls, and open inlets/outlets. (Succi, 2018, pp. 273–274)

Periodic boundary conditions separate the bulk phenomena and the actual boundaries and are used in cases where surface effects have a negligible role. With a two-dimensional case, square boxes with lattice sites are created where discrete fluid is represented with as many matrices as there are velocity vectors in the model. Model D2Q7 contains 7 matrices. To model interaction between these NxM lattice sites, it is specified that the population at some of the sides that face inward is equal to the opposite side that faces outward. The no-slip boundary condition is defined as the condition where, at the solid surface, velocity is zero. Collision steps at the boundary undergo collision normally. The streaming step is defined as any population leaving the domain being reversed along the same link, and the net velocity is zero at the boundary. (Succi, 2018, pp. 274–278)

Free-slip boundaries allow the fluid to move without friction on the wall and prevent penetration. The velocity component normal to the wall is zero. A frictional slip boundary is a condition between no-slip and free slip that allows some tangential velocity at the boundary with some frictional resistance. (Krüger et al., 2016, pp. 279–280)

Open boundaries such as inlets and outlets define where the flow enters or exits the domain. With the inlet flow, the equilibrium population is constantly replenished with the desired density and velocity. Outlet flow is more complex. For the outlet, another boundary condition is applied, that is a porous plug. The main idea is that particles reaching the outlet should slow down. Otherwise, a backwards propagating disturbance is triggered. (Krüger et al., 2016, p. 284)

Parallelization employs domain decomposition, where the computational domain is divided into multiple subdomains, and each is handled by a different computing core. Communication between subdomains occurs at the boundaries, exchanging distribution functions through layers. This type of parallelization is commonly used in many applications, but the LBM is well suited for this type of computing due to its local and explicit collision and streaming algorithm.

2.6 Quantum computing

Quantum computing is based on the principles of quantum mechanics, primarily superposition and entanglement. Information processing operates on principles that are fundamentally different from those used in classical computing. Quantum bits, or qubits for short (Nielsen and Chunag, 2010, p. 13), are the basis of quantum computing, as bits are the basis for classical computing. With classical computer bits having states 0 and 1, quantum bits have similar states $|0\rangle$ and $|1\rangle$. Notation around one and zero is called Dirac notation. Notation around one and zero is called Dirac notation. The difference between qubits and bits is that a qubit's state can be other than the two represented states. Combining these states is called superposition and can be expressed by equation (19)

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle\,. \tag{19}$$

In this equation, α and β are complex numbers corresponding to each state. To link multiple qubits together are called separable if it is the tensor product of qubit states or entangled if it can’t be expressed as a tensor product of qubit states, which are crucial for quantum algorithms (Nielsen and Chunag, 2010, p. 571-573). When qubits are in an entanglement state, they are connected in such a way that properties are combined, and

states are shared. Quantum computations are processed through quantum gates, and information extraction involves quantum measurements where quantum states are collapsed probabilistically (Nielsen and Chunag, 2010, p. 188-189). Quantum computing has problems, and one of them is decoherence. This phenomenon means that qubits lose their quantum properties, and therefore quantum error correction is essential (Nielsen and Chunag, 2010, p. 435).

Comparing classical bits to quantum bits as an example, three bits have eight possible states: 000, 001, 010, 100, 011, 101, 110, and 111. Three qubits have eight computational basis states: $|000\rangle$, $|001\rangle$, $|010\rangle$, $|100\rangle$, $|011\rangle$, $|101\rangle$, $|110\rangle$, and $|111\rangle$. With bits, states are local, but with qubits, those coefficients are not local but global. This means that three qubits are represented with state vector (20)

$$|\psi\rangle = a|000\rangle + b|001\rangle + c|010\rangle + d|100\rangle + e|011\rangle + f|101\rangle + g|110\rangle + h|111\rangle. \quad (20)$$

So, the number of possible combinations is 2^n, for both bits and qubits but the difference comes from the fact that qubits are in superposition. This means that it can be in all states at the same time. Because coefficients are global, it means that coefficients cannot be derived from local states. Global nature makes the system as a whole, and each qubit is not an independent unit anymore. (Nielsen and Chunag, 2010, p. 16)

Scaling from three bits and qubits to 50 bits and qubits, the exponential effect of qubits can be seen. With 50 bits, information is represented with 50 zeros and ones and can store the information exactly once at a time. With 50 qubits, superpositions of all 2^{50} states can be represented simultaneously. Comparing these in theory 50 qubits can represent over a quadrillion (10^{15}) classical states.

Sometimes the Quantum Lattice Boltzmann Method QLBM enables the simulation of quantum systems governed by the Dirac equation and other fundamental quantum mechanical principles, extending the classical LBM to include quantum effects. However, within the scope of this thesis, QLBM means transferring the LBM into the quantum computational setting. In other words, quantum algorithm implementation of the LBM. This concept is explained in the next chapter.

2.6.1 Quantum Lattice Boltzmann method

At the basics of QLBM, it follows a similar algorithm structure to LBM but is executed in the quantum domain. This process is divided into four key stages: encoding, collision,

propagation, and also the calculation of macroscopic variables shown by (Budinski, 2021). Here, as an example, the D2Q5 lattice is used.

State encoding is a step where the initial fluid variable $\Phi(x,0)$ is encoded into a quantum state vector. This involves preparing a superposition state across qubits in such a way that the amplitude of each basis state corresponds to a fluid variable. Efficient state preparation is non-trivial for a lattice with *M* sites. One must initialize a superposition of *M* basis states with appropriate amplitudes. The encoding step in the quantum state is shown by the equation

$$|\psi_0\rangle = |0\rangle_a \sum_{k=0}^{8M^2-1} \phi_{\alpha,k}/\|\phi\||k\rangle_q. \tag{21}$$

State initialization cost scales with the qubits that are required in the *q* register and is calculated by $n = \log_2(5 * (M * M)) + 1$, where the additional qubit is for the collision step.

Collision step with the LBM, interactions of particles are computed at each lattice site. And with QLBM, collisions are implemented by a quantum gate that operates on the quantum register representing each site's distribution. With the collision step, equation (21) becomes the form of equation (22) shown by (Budinski, 2021)

$$|\psi_1\rangle = |0\rangle_a \sum_{k=0}^{8M^2-1} a_{k,k}\, \phi_{\alpha,k}/\|\phi\||k\rangle_q + |1_{\phi}^{\perp}\rangle_{aq}. \tag{22}$$

In this equation, $a_{k,k}$ is the diagonal operator for entries of eight-block and with this distribution, functions are represented by the quantum sub-states corresponding to each block.

The propagation operation (streaming) step in LBM shifts post-collision distributions to neighboring lattice sites along their respective directions. In QLBM, propagation is implemented by shifting quantum amplitudes between basis states corresponding to the neighboring lattice positions. The quantum state is re-indexed to affect particle movement, which is done by quantum gate operations. With the D2Q5 model, there are four distribution functions that are shifted in four directions and velocity vectors defined by equation (23)

$$e_\alpha = \begin{cases} (0,0), & \alpha = 0 \\ (\pm e_x, 0), & \alpha = 1,2 \\ (0, \pm e_y), & \alpha = 3,4 \end{cases}. \tag{23}$$

Where one out of the five is a rest particle. For other models' velocity vectors are shown in Table 1.

These links from equation (23) are implemented into a quantum circuit with two operators, *R* and *L*. With these operators' the quantum state of the entire system now becomes in form

$$|\psi_2\rangle = |0\rangle_a LR \left(\sum_{k=0}^{8M^2-1} a_{k,k}\, \phi_{\alpha,k} / \|\phi\| |k\rangle_q \right) + |1_{\phi}^{\perp}\rangle_{aq}. \tag{24}$$

This propagation process does not affect the last term of the equation.

Macroscopic variables based on equation (25) are extracted with addition between quantum sub-states that correspond with respect to the specific distribution function.

$$\phi(x,t) = \sum_{\alpha} f_\alpha(x,t) \tag{25}$$

This is achieved by altering the quantum sub-states that must be introduced between the existing states. By applying post-selection of the state and performing normalization, for the next timestep the desired value of the variable $\Phi(x,y)$ is obtained. (Budinski, 2021)

With these steps, QLBM replicates the stream and collision cycle of the LBM on a quantum platform. Algorithmic parity with classical LBM means that fluid equation recovery can be maintained, but implementing this in a quantum platform constrains some challenges. The quantum circuit depth grows poly-logarithmically with the number of lattice points (Budinski, 2021), which increase the potential for significant speed-up compared to traditional CFD.

QLBM is studied constantly, and several enhancements and lots of progress have been made. It is shown by (Georgescu et al., 2025) that QLBM research has taken three different directions that are studied currently. Some of the research studies focus on implementing separate streaming and collision in QLBM, without Linear Combinations of unitaries, one of these studies is shown by (Schalkers and Möller, 2024). In other studies it is shown that computing the collision step can be practically linearized, which includes

the design of applications for the advection-diffusion equation (Wawrzyniak et al., 2025; Xu et al., 2025). These are a few of the studies that show that QLBM is ongoing research.

2.6.2 Hardware and emulators

Quantum computers are relatively new. In Finland, the first quantum computer was finished in 2021 and has 5 qubits, and the second was built in 2023 and has 20 qubits (VTT, 2023). The third quantum computer was taken into use in March of 2025 and was the largest superconducting quantum computer in Europe at the time, containing 50 qubits (VTT, 2025). Quantum computers are constantly developed and tested and have shown great potential in different applications.

Quantum computers can be categorized based on the technology used to implement and control qubits, and there are several different types of quantum computers. VTT's 50-qubit computer is a superconducting quantum computer, which is the most common type of quantum computer at the time. Superconducting circuits are utilized by quantum processors. A superconductor has a special feature where resistivity drops near zero at some temperature. This temperature depends on what material is used but is very low, and therefore these types of quantum computers need constant cooling. Trapped ion quantum processors use individual atoms as qubits, and these are manipulated by electromagnets. These types of computers also need to be cooled to extremely low temperatures to reduce external noise. The photonic quantum processor uses photons as its qubits and is based on quantum entanglement. Neutral atom quantum processors use neutral atoms trapped in a vacuum chamber. Atoms are cooled to low temperatures to reduce movement, and those can be manipulated by lasers. Topological qubits are based on topological materials that are insulated inside and conducting on the surface. Topological qubits are much more resistant to noise than traditional qubits. (Chander, 2024)

Quantum emulators are developed to simulate quantum computers and are used for research purposes. Emulators run on classical computers, and with these, the number of qubits is limited to the memory of the classical computer. One of the simulators is the Intel Quantum SDK system and is explained by (Shinde et al., 2025). Other emulators are Matrix Product Emulators Qiskit-MPS, Quimb-MPS, QMatchaTea-MPS, and MIMIQ-MPS these are commercial emulators. Tensor Network emulator Quimb-TN, Decision Diagrams emulator MQT-DDS and Pyqrack. These are emulators that are tested and benchmarked by (Leonteva et al., 2025) in their study. Emulator that is used in this study is Qiskit Aer 0.17.1 which is an open-source SDK emulator. It is run on 16 core m6g.4xlarge machine from Amazon web services.

3 CFD SIMULATION

In these simulations, a DJI Phantom 3 drone is used. This study concentrates more on the simulations part, and therefore already meshed model is used. The same mesh is used in a study by (Paz et al., 2020). Mesh and geometry are simplified versions of the DJI Phantom 3 drone, only containing the fuselage, motors, and propellers. In other words, landing gear, camera mount, and camera are removed from the model seen in Figure 5.

3.1 Mesh and Geometry data

The Phantom 3 drone has four propellers that have a diameter of 240 mm. Propeller characteristics are shown in Figure 5. In the same picture, the original drone geometry is also shown, as is the simplified model.

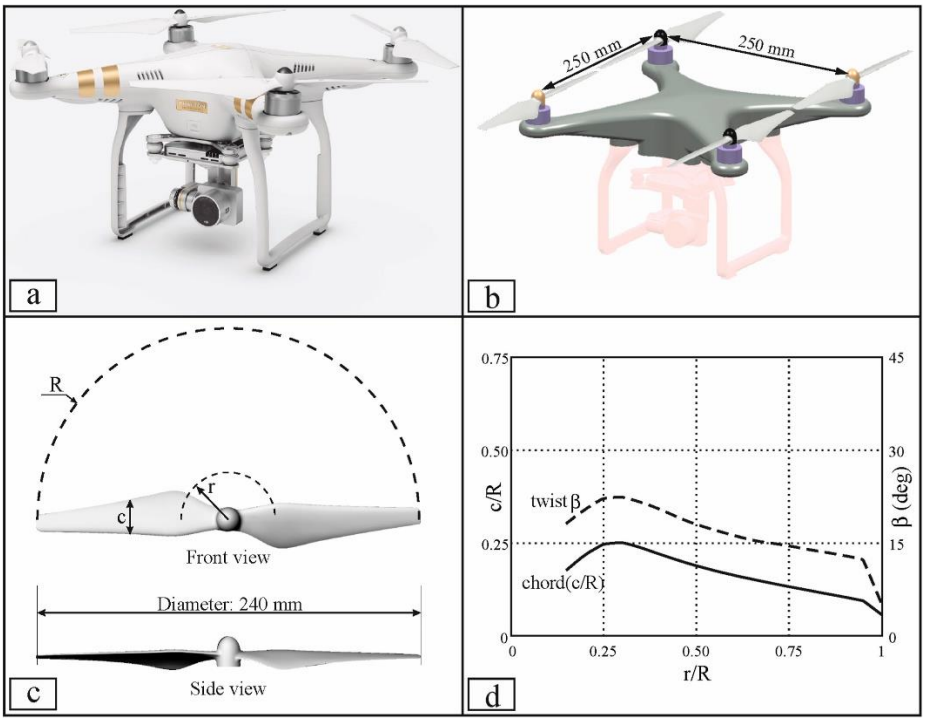

Figure 5. *Drone measurements a) original drone b) modeled drone c) propeller d) blade twist and chord ratio made by (Paz et al., 2021).*

Figure 5d shows the twist and width of the chord along the diameter from center to tip. Two of the propellers rotate counterclockwise and two clockwise propellers are placed in such a way that pairs are on opposite sides of the cross shaped fuselage. Propellers

maximum rotating speed is around 1250 rad/s (Paz et al., 2021). Maximum rotation speed depends on the charge of the drone's battery.

The mesh is constructed from approximately 25 million elements. The mesh around the drone is constructed so that the distance from the ground to the center of the propellers is 192 mm. Some of the key elements of the mesh are an aspect ratio of 25 in the first layers, around the model prismatic layers are set to 10, and the maximum 2D size is set to be 1.5 mm. At the propeller's surface, 2D cell size is reduced to 0.3 mm. Propellers have 160,000 surface cells, and each rotating domain contains 3,900,000 elements. (Paz et al., 2020)

Mesh at different locations is introduced below. Cross sections show where mesh is denser and at what location it is coarser.

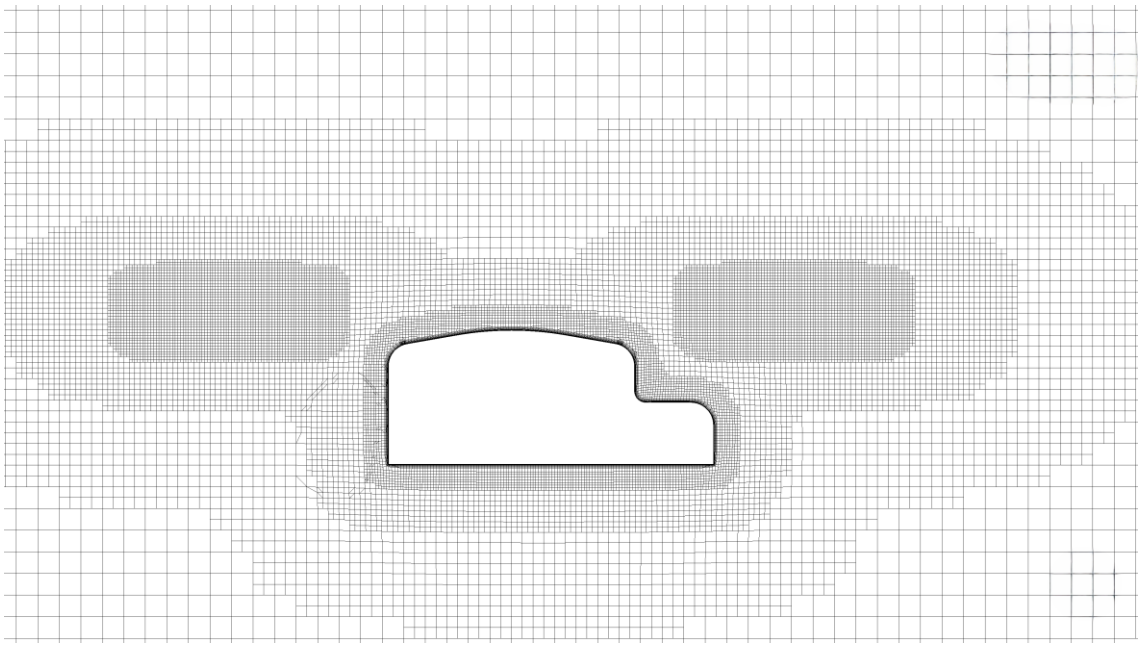

Figure 6. *Cross section at the middle.*

In the Figure 6 cross-section, the quadcopter mesh can be seen. The front side of the drone is seen in the figure on the left side and the back on the right side. Mesh gets denser near the frame of the quadcopter but also near areas where propellers are rotating.

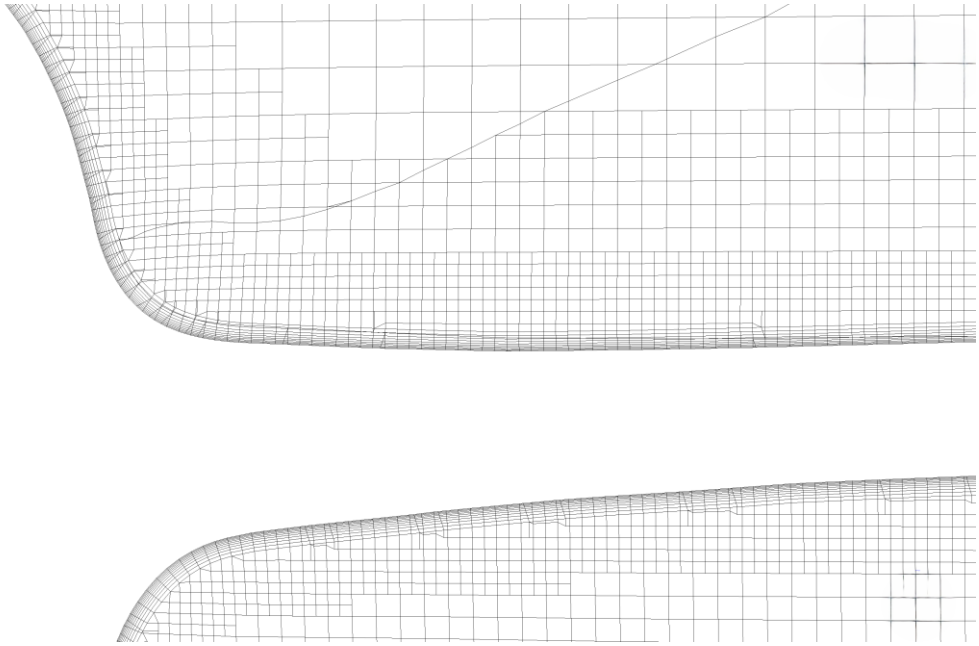

Figure 7. *Cross section of propeller mesh near center.*

Figure 7 shows a closer view of the mesh near the propeller. Inflation is also seen near the wall. These figures also show the size and shape of mesh elements in different locations. Mesh is structured mesh that contains 3D hexahedra elements and 2D quadrilaterals.

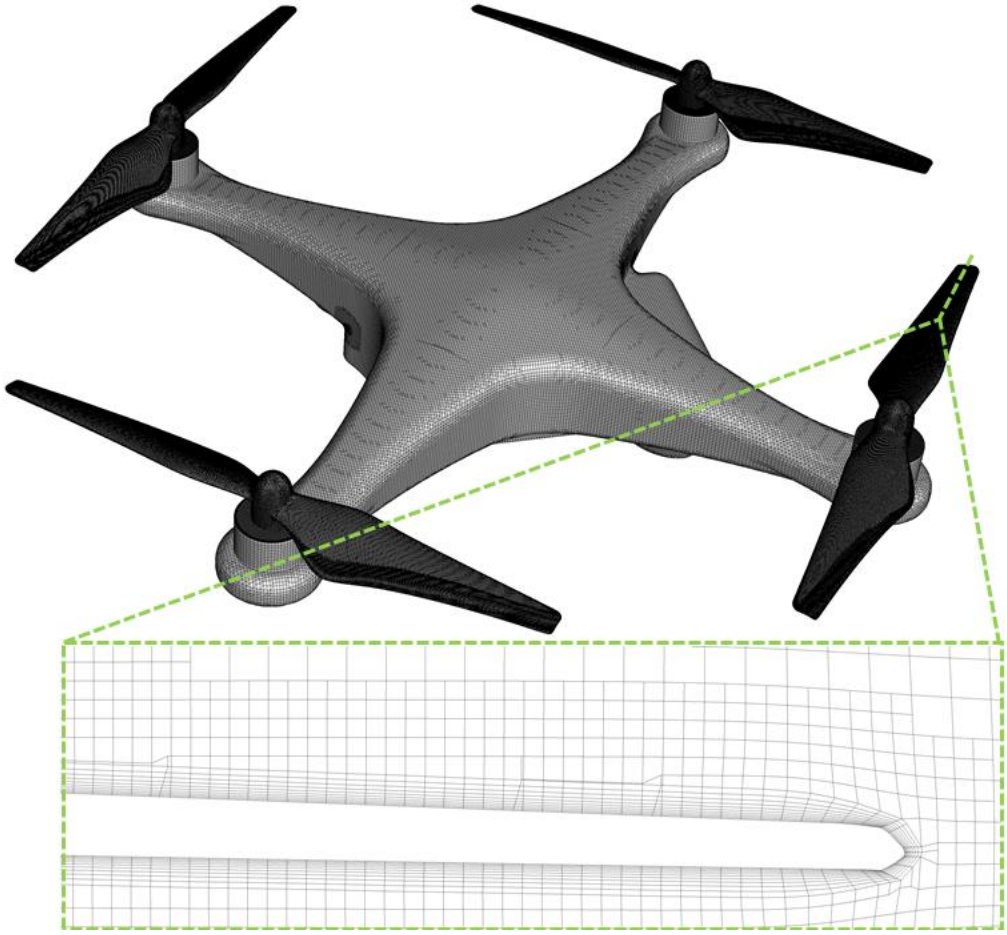

Figure 8. *Whole drone surface mesh and cross section on tip of the propeller.*

Smaller 2D mesh size used at propellers is seen from Figure 8, where surface mesh is displayed. Cross section of the blade shows the inflation near the tip.

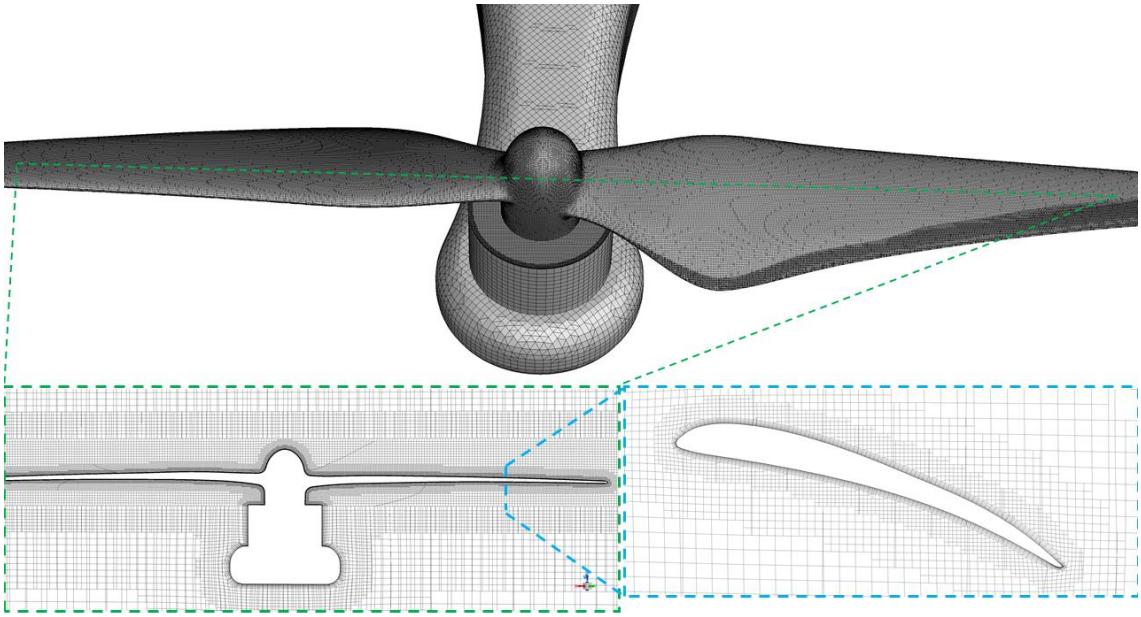

Figure 9. *Propeller and airfoil mesh.*

Figure 9 shows the surface mesh of the propeller and part of the frame where the motor and propeller are attached. Cross sections at the middle of the propeller and cross sections of the airfoil are seen. The mesh near the propeller is critical to maintain quality, because the velocities are higher in that area.

3.2 Simulation tools and techniques

Settings for simulations are changed in Ansys Fluent preprocessing at the beginning. In general settings, the steady and transient time method is changed between different simulations. The turbulence model chosen for simulations is k-omega SST to yield more accurate results near the wall and far from the wall. Model constants are set to the original values. Fluid material is air at ISA conditions, where the density is 1.225 kg/m^3. The propeller's motion is simulated with frame motion in steady state, and with transient time, the sliding mesh is changed. The propeller's speed is set to 1,000 rad/s, which is approximately 80% of the maximum rotational speed. Boundary conditions that are set are walls for quadcopter surfaces with no-slip conditions, and the ground is also set to wall. The outer lateral surface and upper surface of the flow domain are set to pressure outlets when simulating a quadcopter at a stationary state. These pressure outlets are changed to velocity inlets when simulating drone movement. With frame motion, the mesh interface between rotating and stationary domains is determined. The solution method

schemes that are used are PISO and SIMPLEC. The pseudo-time method is set up with steady-state time.

Calculations are done in CSC servers using the Mahti supercomputer. Using an HPC cluster for simulations reduces the simulation time significantly. With local or virtual computers, the number of cores is limited. With Mahti, the number of CPU cores is much larger than it is reasonable or possible to use with CFD simulations. With large partitions, the number of nodes is up to 700, and each node contains up to 128 cores running at 2.6 GHz. Memory at Mahti is up to 256 GiB per core. Mahti HPC does not have interactive user interface that could be used in these simulations, and therefore the pre- and post-processing must be done with the local or virtual computer. After the simulations are finished, the output data is imported to the local computer for post-processing.

Different core amounts is tested based on the rule for using Mahti that by doubling the core amount, computational time must increase over 1.5 times (CSC, 2024). According to this rule, HPC does not utilize more resources than what is considered efficient. Increasing the core amount from 64 to 128 decreased computational time almost by half. A core amount of 256 is also tested, and the computational time is efficient enough to run with this. The problem with running Ansys most of the time with 256 cores is the limitation of the license used by CSC. With testing, it is found that the most suitable number of cores is 128. A license error occurred most of the time when using 256 cores.

Starting simulations in the Mahti HPC cluster, three files are needed at the first simulation time. When simulations are continued from previous simulations, four files are needed. These files are a case and data file, a journal file, and a batch job file. The case file contains the mesh and all settings that are changed and set in Ansys on the local computer via the graphical interface. At the beginning, it is easiest to set all the basic settings using a graphical interface. In the situation where the same case file is used for multiple simulations, setting contours, vectors, and other similar objects before starting the simulations makes post-processing easier. Setting these before simulations removes the need to set them for each simulated case individually in post-processing. A data file is used when a simulation is run at least once before, and simulation data is needed for the next simulation. For instance, where transition from a steady to a transient time is made.

The batch job script contains information about the job that is done in Mahti. The essential information is included in the batch job script and consists of simulation time, the number of cores used, and the command to start Ansys Fluent. After Fluent is started inside Mahti, it starts reading the journal file straight away.

The Fluent console reads the journal file line by line and acts on the commands. At the beginning, the case and data file are loaded. After these are loaded case can be changed. Most of the commands that can be used are listed in (Ansys, 2025b). Commands listed by Ansys contain only the main commands to change settings but not the settings inside that command, and due to that, every command that is used in the journal needs to be verified in local Ansys. Although some commands are simpler, like starting calculations on a steady-state case, “solve/iterate 1000”. Some commands need an ID number for zones or areas. One of these kinds of commands is the changing from MRF to sliding mesh “mesh/modify-zones/mrf-to-sliding-mesh 2”. IDs need to be checked in advance. The command to turn on the FW-H is one of the examples where there are multiple settings changed by one command: “define/models/acoustics/ffowcs-williams yes yes no 1.225 340 2e-05”. To generate a journal, it can be recorded directly from local Fluent or by writing by hand. When writing it from Fluent, the commands also need to be written on the console. One of the time-consuming parts is getting the journal file correct. If there are some errors in commands, Fluent stops. Therefore, when iteration is started and completed, there should be a command to save data, and auto-saving should be set when simulations are running. This approach ensures that if an error occurs in the journal, already calculated data is saved and simulations can be continued from that. Fluent itself makes some assumptions in the console, if commands are “yes” and “no”, but only “y” is written, it reads it as “yes”. The final command in the journal file should be to close Fluent. When Fluent is closed, the batch job also ends. In the case where Fluent is not closed properly, Mahti keeps using allocated resources until the time limit set in the batch job is achieved, even if Fluent isn’t doing anything. Also, if the time set in the batch job runs out before the simulation is completed, the job is ended.

3.3 Aerodynamics

In this chapter, steady state and transient simulations are explained, and simulation results are shown. Steady state simulations are crucial to get an initial solution that is then used for a transient solution.

3.3.1 Steady state

Steady state simulation is used to determine initial values for the transient simulation. Simulation is started by choosing the steady state method. For the viscous model, k-omega SST is chosen to simulate flow near walls but also further away from them. With the steady state model, the MRF approach needs to be chosen. The pressure-velocity coupling scheme is chosen to be COUPELD to obtain more accurate results at the cost

of computing time. Least squares cell-based is set for the spatial discretization gradient, and for pressure, second order is selected. Second-order upwind is set for momentum, turbulent kinetic energy (k), and specific dissipation range (omega). The pseudo time method is turned on. URFs are chosen to be default values. To check if the solution is convergent, it was monitored with force in the *z* direction with a value of 1e-5. Other variables like drag, momentum and force to different directions is also examined and that these supports the convergence of the solution.

Simulations are calculated with 128 CPUs and run in one node. Starting with the stationary case solution, it converged at 4,100 iterations with the chosen convergence value and took 11h 12min. To ensure that the solution is converged, residuals and aerodynamic features like lift coefficient, drag coefficient, and force need to be checked to see that they settle towards some constant value. If the values fluctuate or start to fluctuate after some iterations, the solution is not converged. Net mass flow from the outlet/inlet is checked to be small enough so there is not more flow into the domain than there is out or vice versa.

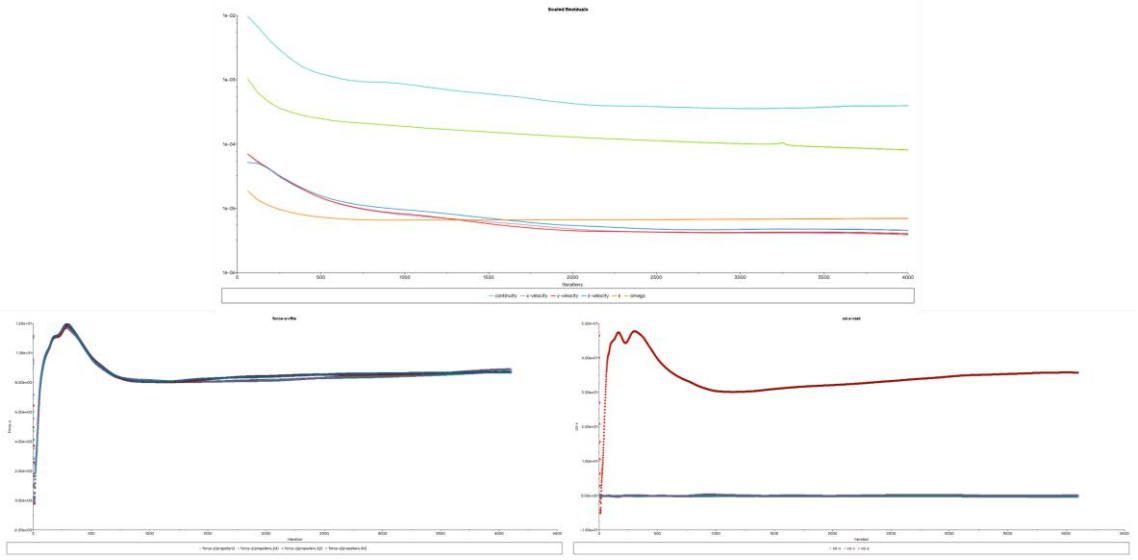

Figure 10. *Stationary simulation residuals, force to z direction and drag coefficient.*

From Figure 10 it is seen that residuals, on the top, sets towards a constant value. In bottom right force from each propeller settles towards a constant value, and the same is true for the drag coefficient in every three dimensions and is seen in bottom right of the figure. Net mass flow is -6.84e-06. It is also noted that the significant change in values happens with the first 1,000 iterations. Because the only use for a steady state solution in this simulation is to determine initial values for transient simulations, the 1,000th iteration solution data is satisfactory enough.

Forward motion was simulated with the same settings, where the only difference is that forward velocity is applied. Simulation time is around 12 h 16 min with the 4,500 iterations, and after the 1,000th iteration, the solution provides satisfactory enough results to start transient simulations. Residuals and the force or drag coefficient are not presented here. It is noted that there are some minor differences in those compared to the stationary case. The differences in the force and drag plots indicate that there is greater variation among the propellers.

3.3.2 Transient simulation

The transient case is started by solving it with a steady state. This approach reduces computational time by calculating more accurate initial values for the transient. From a steady solution 1,000th iteration is used as it was shown it gives good starting point to transient simulation. A fully converged solution is not used as it is unnecessary. From Figure 10, force and drag start to settle and supply adequate initial values to start the transient time solving.

With transient time simulations, MRF (multi-reference frame) motion is changed to sliding mesh. The pressure-velocity coupling scheme is SIMPLEC. Using transient time, one important parameter that needs to be set is the timestep size. For this particular case, the timestep size is calculated based on the rotational speed of the propellers. Using too large a timestep size, the solution becomes unstable, and with too small a timestep size, it takes longer to calculate one rotation of the propeller. There are different methods to determine timestep size, from the courant number (Blazek, 2015, pp. 353–354) or from the degree of rotation per timestep. Propellers should rotate 1-3 degrees each timestep (Kelecy, 2021; Mak and Salim, 2022). Timestep size was chosen to correspond to a 2-degree revolution per timestep based on testing different timestep sizes. The timestep size is $\pi/90000$ s, given that the propeller rotates at a speed of 1,000 rad/s and each timestep corresponds to a 2° rotation. The timestep size is rounded to 0.0000349 s, which corresponds to a propeller rotation of around 2°.

Two different cases are simulated: one where the drone is at a stationary state and one with forward movement. With the stationary case, the default URF was found to be adequate using the SIMPLEC algorithm. The PISO algorithm was also tested. There were found to be convergence problems where the solution diverged after around 1,000 timesteps. Multiple URF values were tested, decreasing those by 0.1 at a time. Due to the fact that a 1,000 timestep takes 30 h of simulating time, the decision was made to use the SIMPLEC algorithm.

Transient simulations were set to run for an overall 3,960 timesteps that correspond to 22 full revolutions and a flow time of 0.138204 s. Simulations were run in two parts: first to achieve a fully developed flow field for the transient case and second to enable acoustic sampling.

The same output data is used for analyzing aeroacoustics and aerodynamic results.

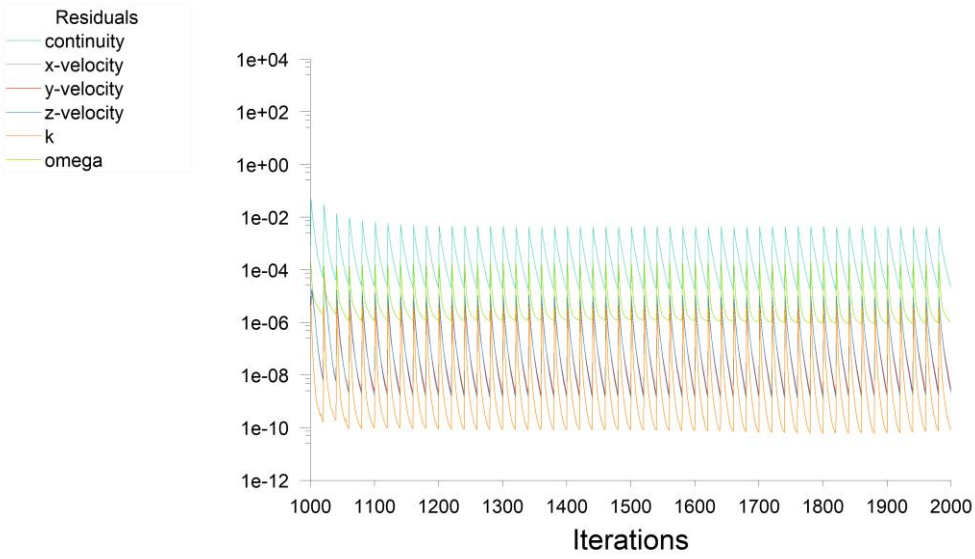

Figure 11. *Transient case residuals for first 1,000 iteration.*

With transient simulation, the solution does not converge towards the end of the simulation but with every timestep as seen in Figure 11. Each timestep contains 20 iterations. These parameters can be changed, but it is found that it is better to change the timestep size than it would be to increase or decrease the maximum number of iterations per timestep. Residuals seem to fluctuate evenly, and at the end of each timestep, residuals drop below 1e-6, except for omega residuals that drop down to 1e-5. This shows that the solution is converging properly at each timestep. From these transient residuals, it cannot be seen directly when the whole solution is converged, only that the solution is converging at each timestep. To see when the simulation is converged, the lift coefficient, moment coefficient, drag coefficient, and force at the propellers are analyzed. These values are saved to text files from the simulations and plotted with Matlab. There are overall 13 figures that are plotted and from those four that have major role when analyzing the convergence. The flow field is converged around 6 propeller rotations or 1,080 transient iterations.

Some of the simulation stats are introduced in Table 2. Number of iterations or timesteps and simulation time it took for those iterations. Also, time led to convergence is shown for steady and transient.

Iterations steady state for initial	1,000
Simulation time for steady	2h40min
Converge iterations for steady	4,100
Converge time for steady	11h12min
Convergence transient timesteps	1,080
Converge transient time	26h42min
Timesteps transient no acoustic	1,200
Simulation time no acoustic	29h42min
Timesteps transient with acoustic	2,760
Simulation time with acoustic	68h19min
Overall iterations	80,200
Overall simulation time	100h 41min
Maximum memory used	151.22
Billing units consumed	15,200

Table 2. *Stationary simulation statistics.*

When the memory used for simulation is known, the maximum allocated memory is limited slightly above that to reduce the used billing units. Billing units are currency inside Mahti that can be bought, and with those used resources are monitored. Figures that are shown here show the final simulated timestep, and the propeller orientation is the same as shown in Figure 8.

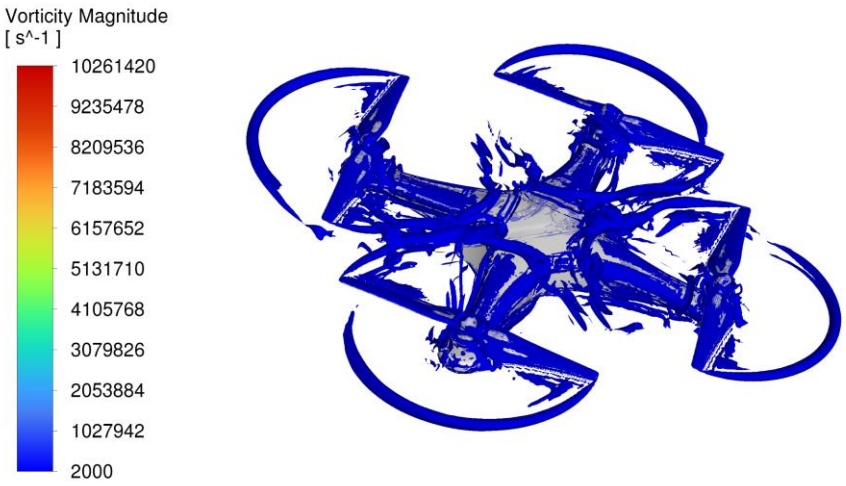

Figure 12. *Iso-surface of Q criterion colored by vorticity magnitude from 2,000 to maximum for hovering.*

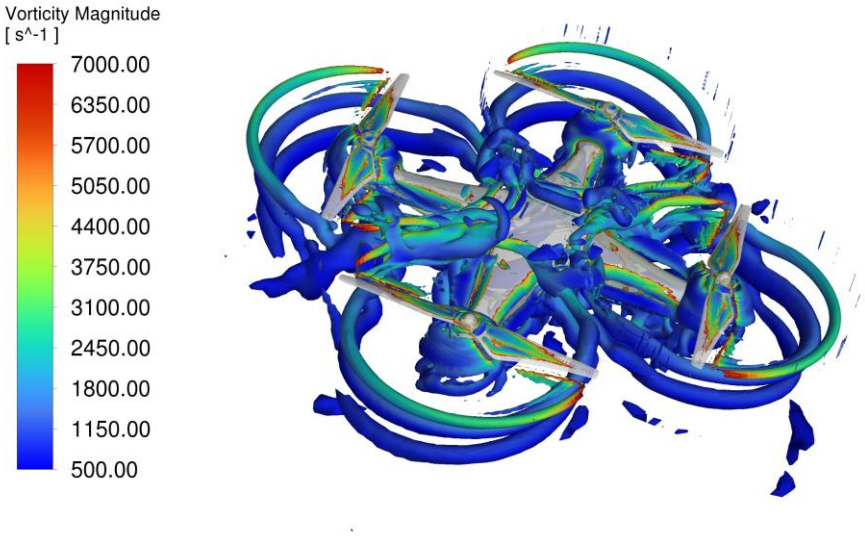

Figure 13. *Iso-surface of Q criterion colored by vorticity magnitude from 500 to 7,000 for hovering.*

Q criterion normalized iso-surfaces colored by vorticity magnitude can be seen in Figure 12 and Figure 13, where maximum and minimum values are limited and are seen on the right side of the figures. Maximum vortices are seen to form near the propeller tip and at the trailing edge.

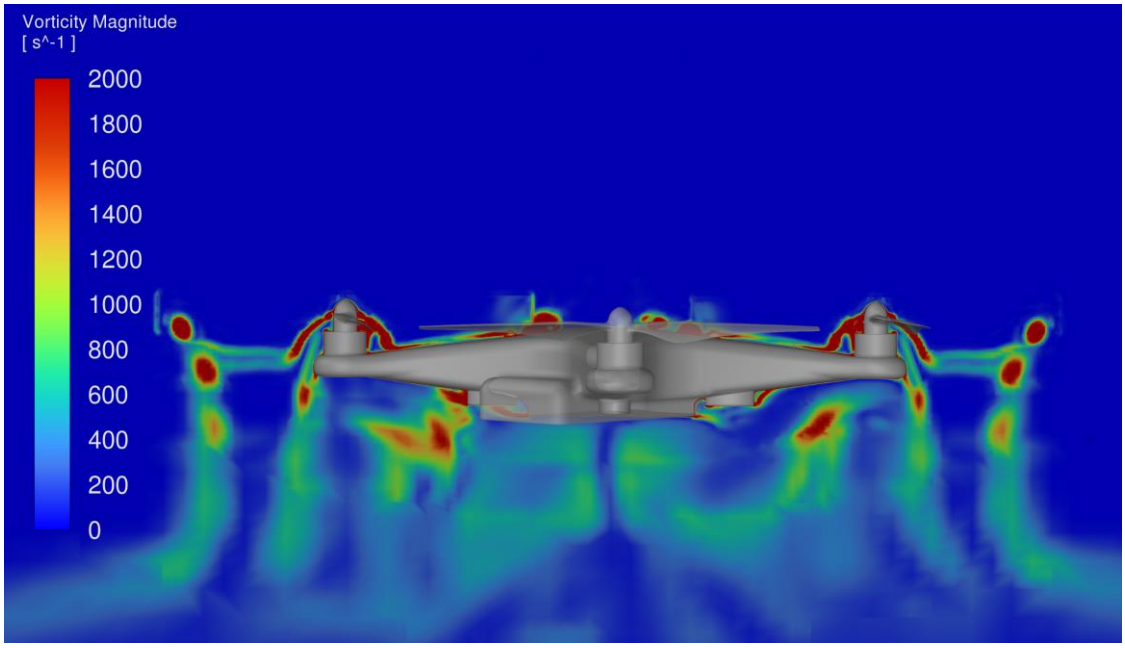

Figure 14. *2D visualization of vorticity magnitude from back left propeller to front right propeller for hovering.*

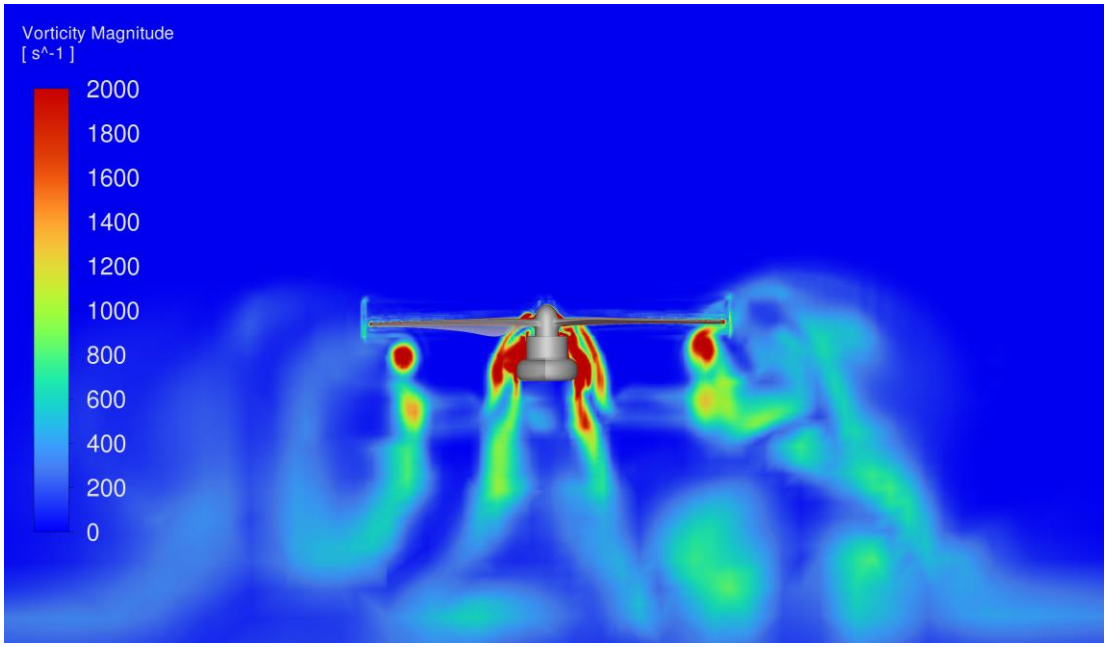

Figure 15. *2D visualization of vorticity magnitude for single propeller for hovering.*

2D cross section from vorticity magnitudes are shown in Figure 14 and Figure 15. Where the first one's cross section is from propeller center to diagonal propeller center, and in the second figure, the propeller cross section is shown. The maximum value is limited to 2,000. Values above that are shown as red, corresponding to the value 2,000 seen on the left side of the figure. Maximum vortices form from the tip of the propeller and from the center of the propeller.

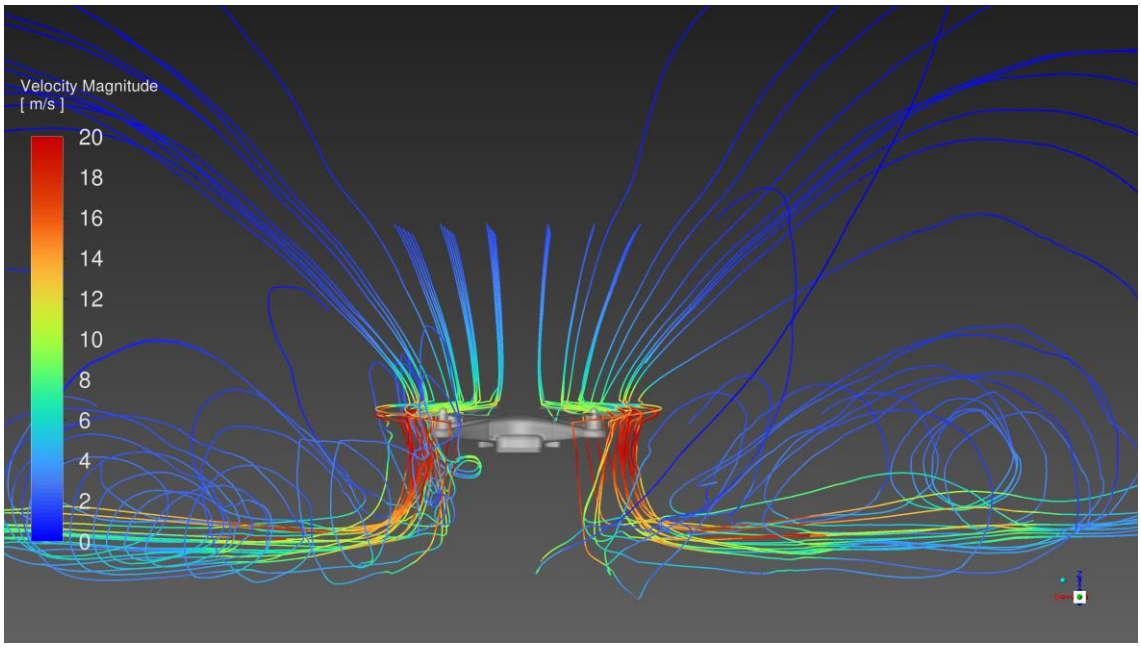

Figure 16. *Streamlines aft view for hovering.*

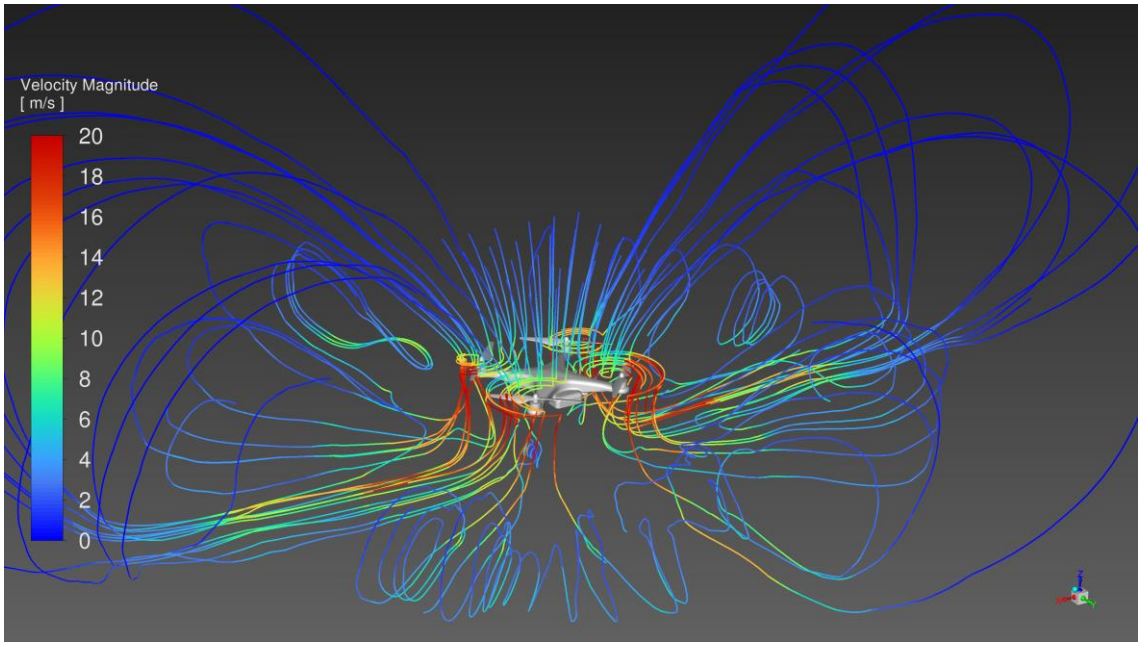

Figure 17. *Streamlines top view for hovering.*

Velocity magnitudes are shown in Figure 16 and Figure 17. In these figures, the maximum velocity value is limited to 20 m/s to better show the variation of velocity at different locations. Maximum velocity is over 150 m/s near the propeller tips. The maximum value is in a small area of the flow field, and no variation would be seen in the pathlines.

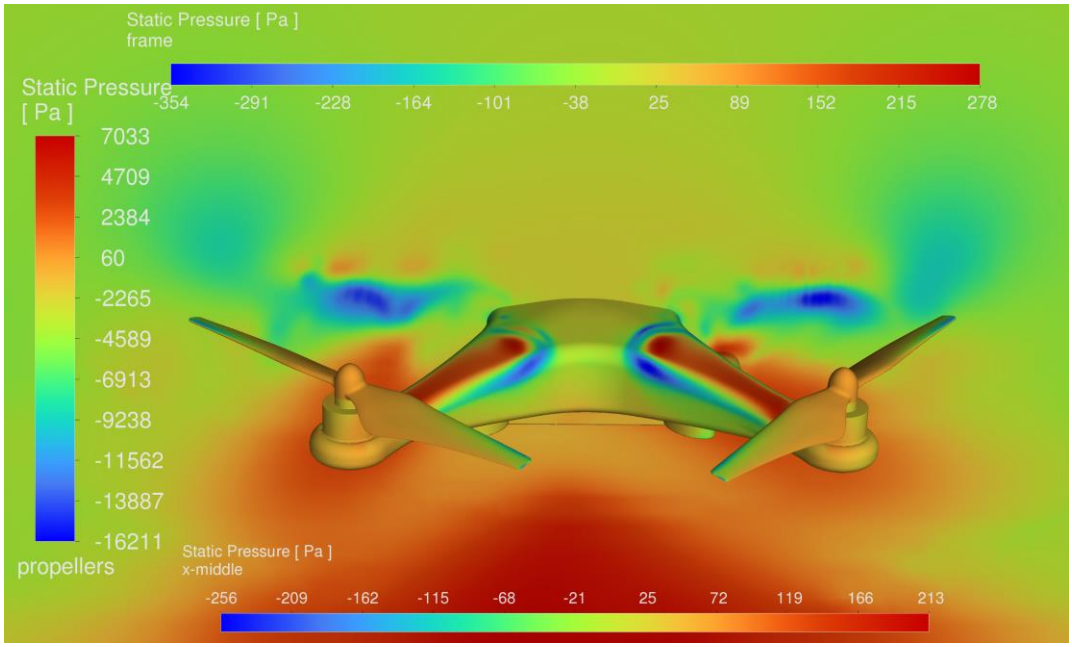

Figure 18. *Static pressure top view and cross section at middle for hovering.*

Figure 18 shows the top side view of drone, in this picture static pressure at middle is included. In Figure 19 orientation is obtained by tilting the picture of drone the way that bottom view can be seen. To this picture also cross-sectional static pressure at propellers

is included. With these two pictures both the top side and underside of drone can be seen.

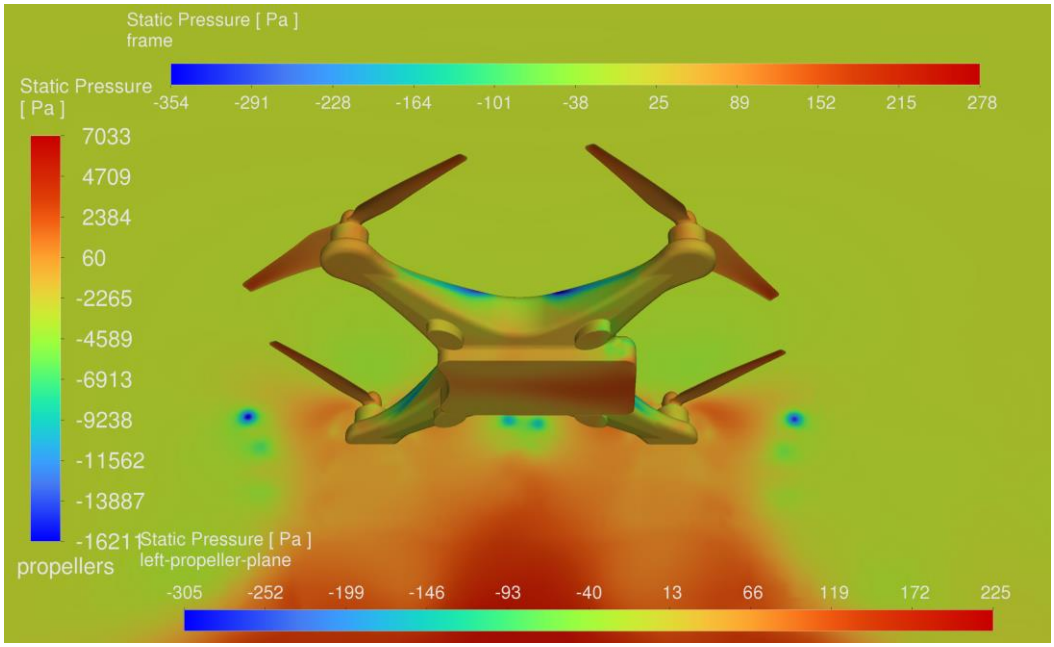

Figure 19. *Static pressure bottom view and cross section at propellers for hovering.*

Static pressure is the pressure relative to the reference pressure. Static pressure at the drone surface and the 2D *xz*-plane at the middle of the drone are shown in Figure 18, and the 2D *xz*-plane at the propeller center with the drone surface is shown in Figure 19. Propellers, frames, and planes have different scales and are named in the figures.

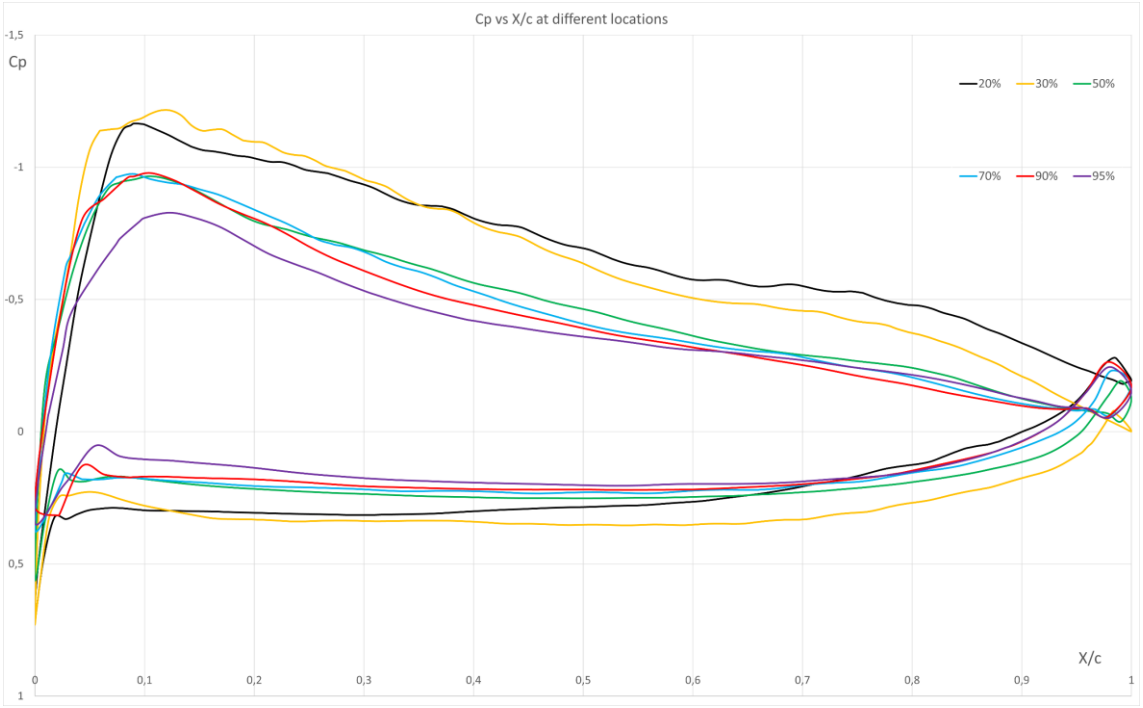

Figure 20. *Pressure coefficients chord wise of blade for hovering.*

From the 3D simulation, pressure coefficients are obtained at different airfoil locations: 20%, 30%, 50%, 70%, 90%, and 95%, and are shown in Figure 20. These pressures coefficients can be compared to the 2D simulations. In Ansys, reference velocity is needed to set separately for each pressure coefficient plot. The pressure coefficient is calculated with equation (26)

$$C_p = \frac{2(p - p_{ref})}{\rho_{ref} * v_{ref}^2}. \qquad 26$$

Where p the static pressure, p_{ref} is reference pressure, density is ρ_{ref}, and velocity is v_{ref}. Static pressure is obtained from simulation reference pressure is set to zero, density is the ISA condition density, and reference velocity is the velocity that the propeller is rotating at the point where it is examined. For example, at a 50% spanwise location, the distance from the center to that point is 60 mm. Therefore, the rotating velocity is 1,000 rad/s times 60 mm, which comes to 60 m/s. Pressure coefficients are calculated directly in Ansys and written into a text file. The data is then combined into one plot. In Figure 20, the upper side of the graph is the upper side of the airfoil, and the lower side of the graph is the lower side of the airfoil.

From the pressure coefficient plots, it is seen that at 50%, 70% and 90% plots are very close to each other, with larger differences in the trailing edge. Pressure coefficient for

95% has a small difference from the leading edge to the halfway point of the chord compared to the three plots. Larger differences can be seen at 20% and 30% locations, where the twist of the blade and also chord length are the highest seen in Figure 5d, from these plotted pressure contours. The differences in these plots are the result of different chord lengths, twists in the blade, shapes of the airfoils, and different velocities at each location.

Forward movement is similar to the stationary case. Only the forward velocity of 10 m/s is added to the case. This is performed by changing boundary conditions to inlets and setting velocity in the positive *y* direction that corresponds to drone forward movement. In the simulation drone is set to zero angle of attack forward movement.

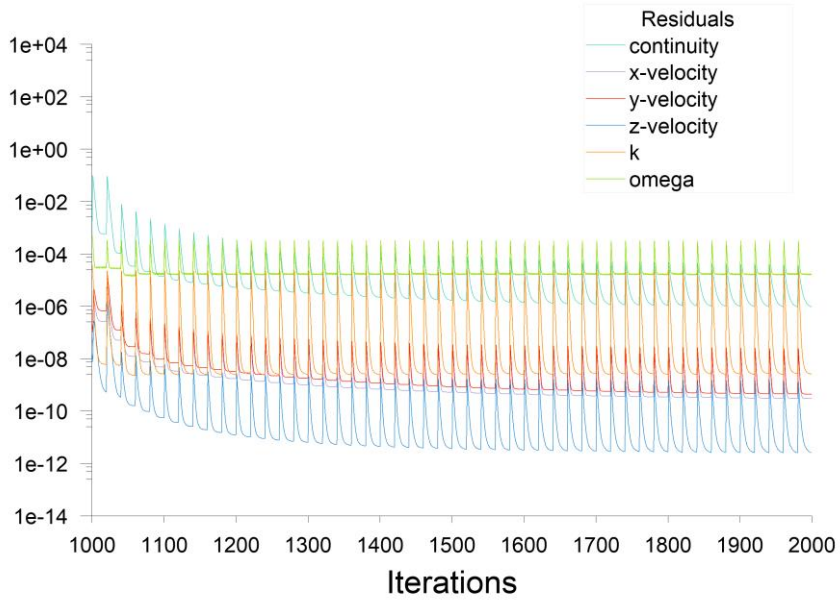

Figure 21. *Scaled residuals for forward.*

Figure 21 shows the first 1,000 iterations of the transient simulation. Iteration numbering starts from 1,000 because simulation is started from 1,000th steady state iteration. From this, it is seen that all residuals drop quickly below 1e-6 except omega, which drops to 1e-5. This graphic shows that the solution converges at the end of each timestep. For the whole simulation convergence, lift coefficient, moment coefficient, drag coefficient, and force at propellers are analyzed. These shows convergence after 7 propeller rotations that correspond to 1260 timesteps. With forward velocity, these report plots aren't as periodic as those for the stationary case, but still convergence can be seen. Net mass flow in and out of boundaries also shows some indication of convergence, it should be

relatively low. In this case, it is around 1e-10. This shows that airflow into and out of the domain is equal.

Iterations steady state	1,000
Simulation time steady state	2h40min
Converge iterations for steady	4,500
Converge time for steady	12h16min
Convergence transient timesteps	1,260
Converge transient time	31h9min
Timesteps transient no acoustic	1,200
Simulation time no acoustic	29h27min
Timesteps transient with acoustic	2,760
Simulation time with acoustic	72h35min
Overall simulation time	104h 42min
Maximum memory used	128.16 GB
Billing units consumed (estimate)	15,805

***Table 3**. Simulation statistics forward movement.*

Simulated iterations, timesteps, and the time required to achieve them are presented in Table 3. In the same table the billing units and maximum memory utilized during the simulations are included. Billing units used in each simulation can be used to calculate the cost of each simulation and is important if simulation is used for commercial application.

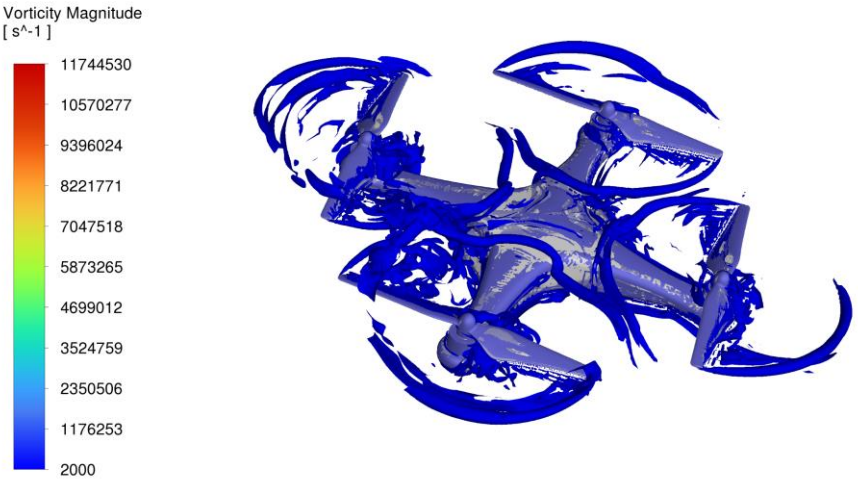

Figure 22. *Iso-surface of Q criterion colored by vorticity magnitude from 2000 to maximum for forward movement.*

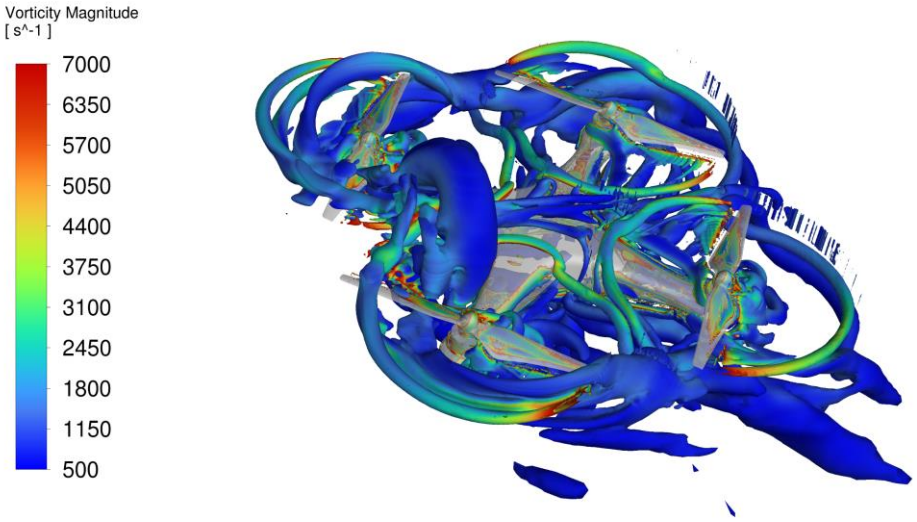

Figure 23. *Iso-surface of Q criterion colored by vorticity magnitude from 500 to 7000 for forward movement.*

With forward motion, Q criterion normalized iso-surfaces colored by vorticity magnitude can be seen in Figure 22 and Figure 23, where maximum and minimum values are limited and are seen on the right side of the figures. Maximum vorticity magnitudes are seen to form at the tip of propellers and at the trailing edge near the propeller center.

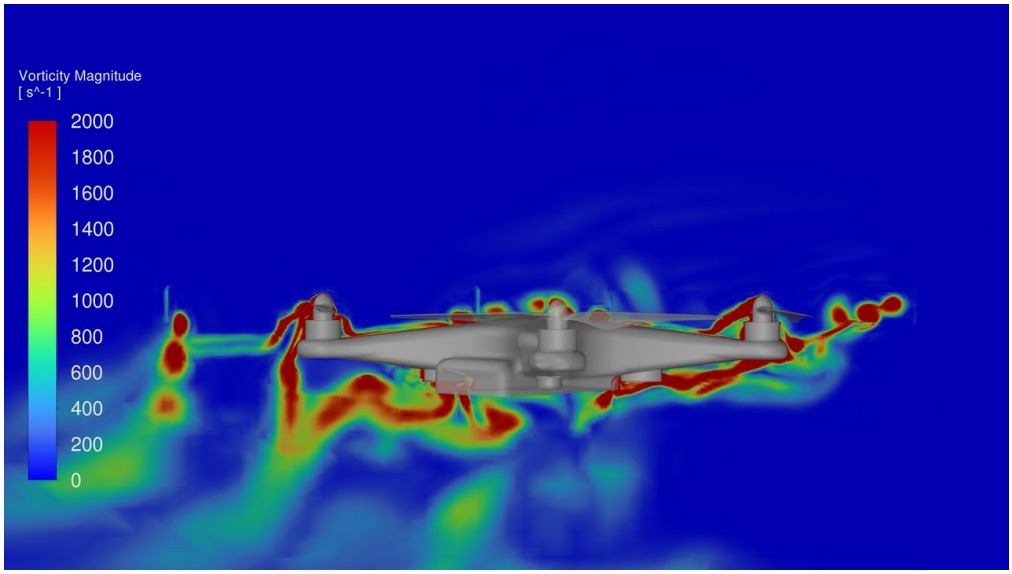

Figure 24. *2D visualization of vorticity magnitude from back left propeller to front right propeller for forward movement.*

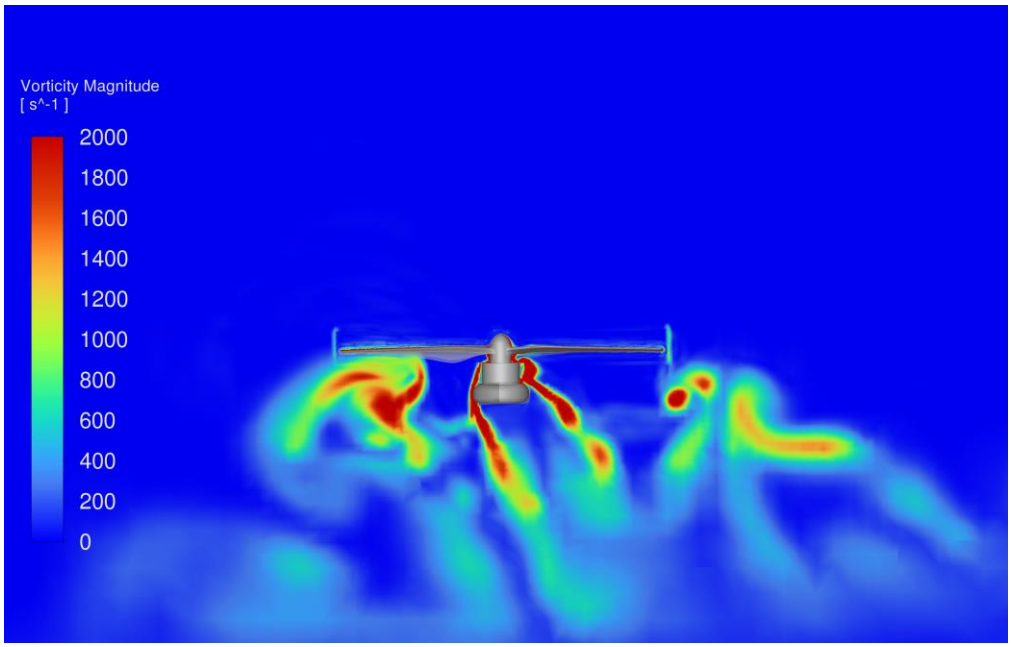

Figure 25. *2D visualization of vorticity magnitude for single propeller for forward movement.*

2D cross-sections of vorticity magnitudes are shown in Figure 24 and Figure 25. The first one's cross section is from propeller center to diagonal propeller center, and in the second figure, the propeller cross section is shown. The maximum value is limited to 2,000, and the values above are colored red, corresponding to the value 2,000 seen on the right side of the figure. Maximum vortices form from the tip of the propeller and from the center of the propeller.

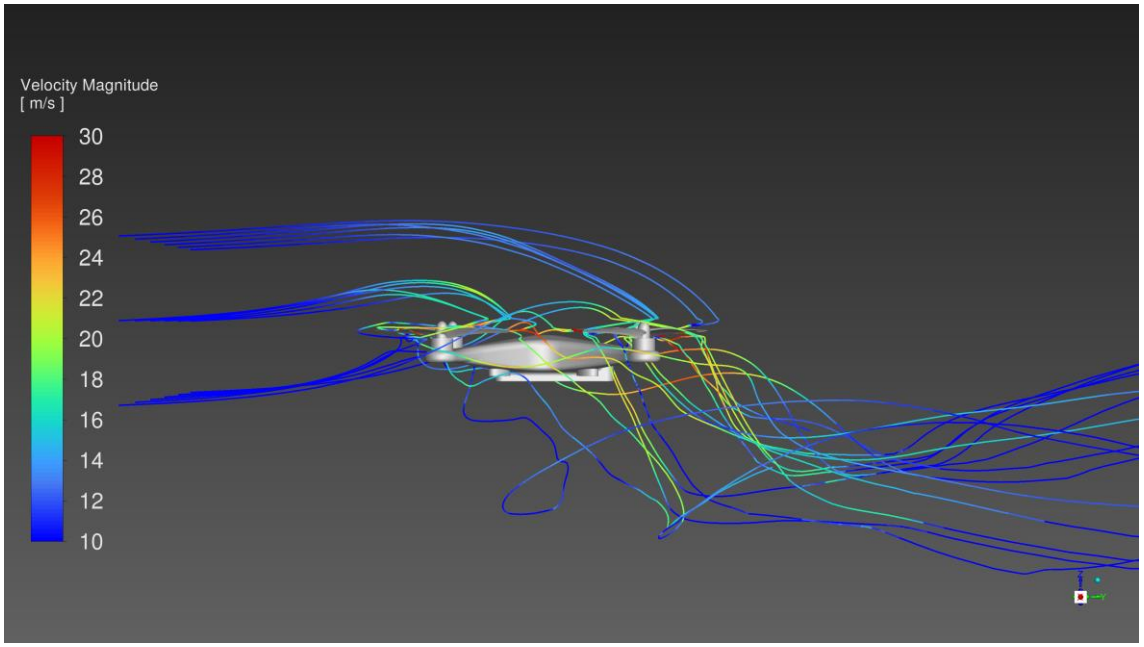

Figure 26. *Streamlines side view for forward movement.*

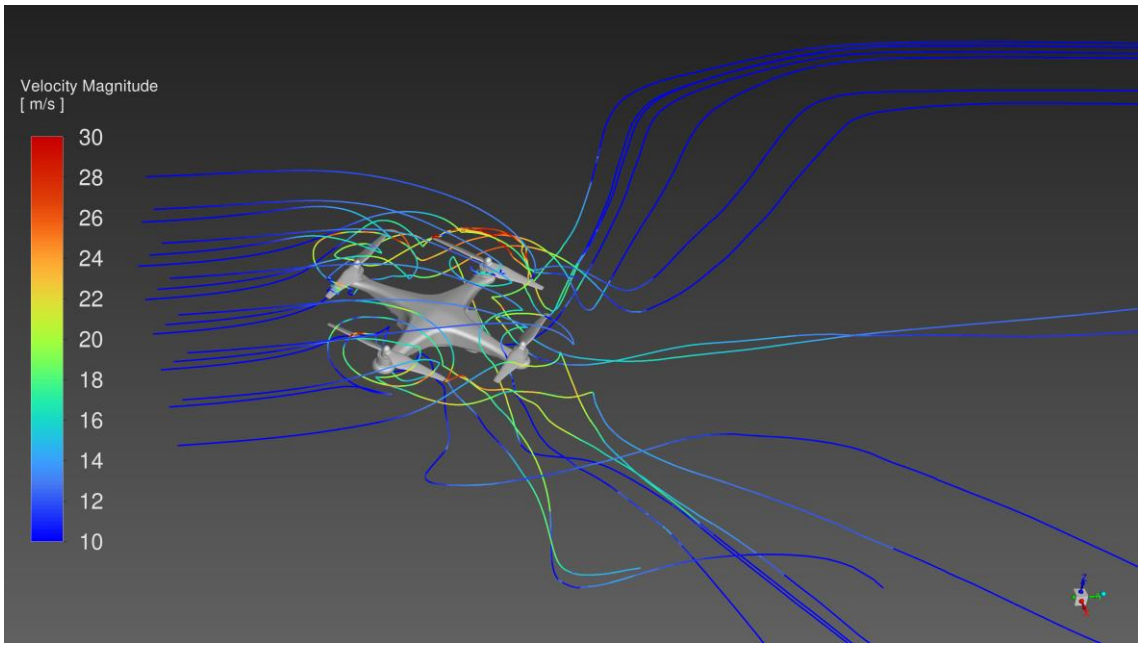

Figure 27. *Streamlines from top-side view for forward movement.*

Velocity magnitudes are shown in Figure 26 and Figure 27. In these figures, the maximum velocity value is limited from 10 m/s to 30 m/s to better show the variation of velocity at different locations. Maximum velocity is over 160 m/s around the propeller tips. The maximum value is only in a small area, as it was with the stationary case. From the pathlines, the direction of the flow can be seen to be from front to back, and at the back, the flow spreads in a larger area.

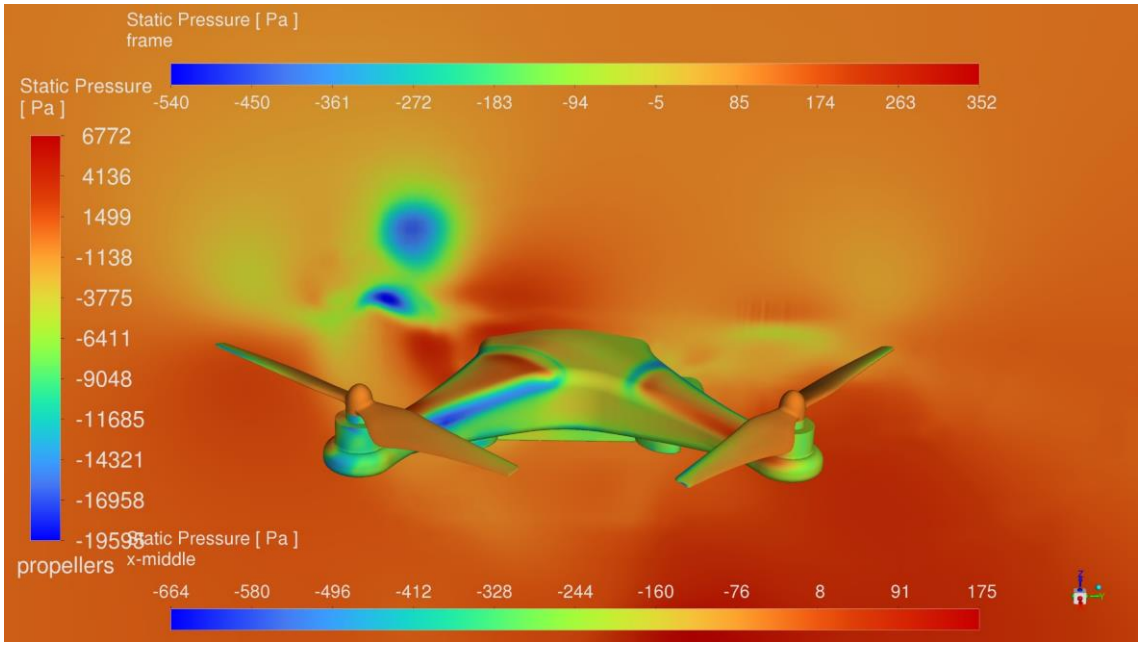

Figure 28. *Static pressure top view and cross section at middle for forward movement.*

Transition between views on Figure 28 and Figure 29 is made by tilting the drone from top view to bottom view. Also, location of cross-sectional view of pressure coefficient is changes between figures.

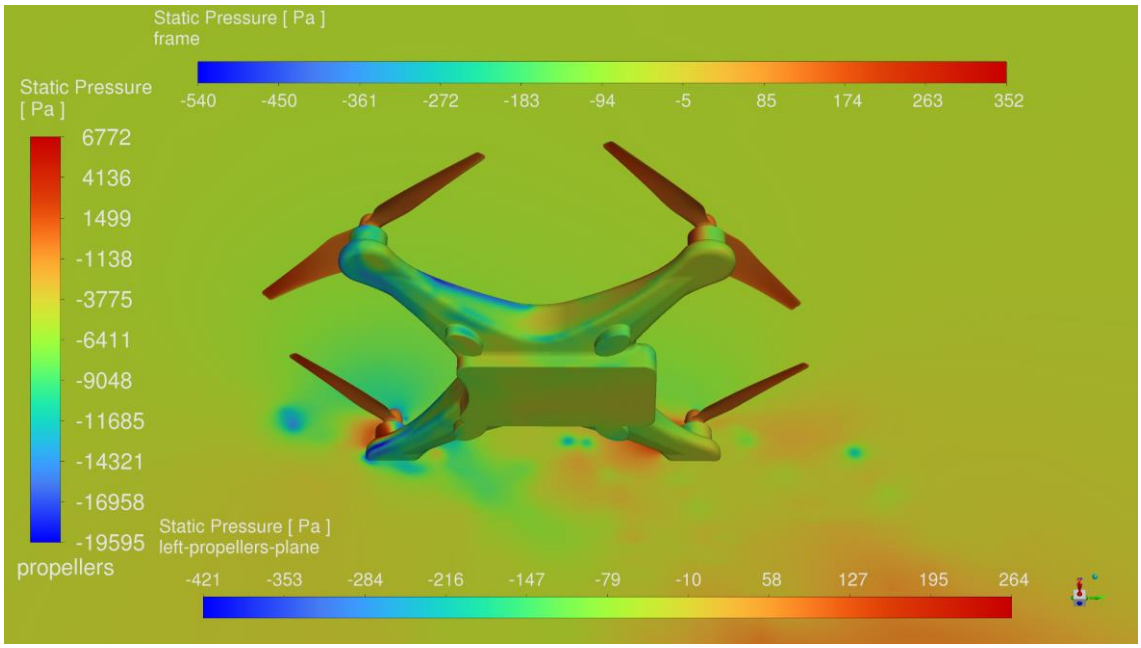

Figure 29. *Static pressure bottom view and cross section at propellers for forward movement.*

Static pressure is the pressure relative to the reference pressure. Static pressure at the drone surface and the 2D *xz*-plane at the middle of the drone are shown in Figure 28,

and the 2D *xz*-plane at the propeller center with the drone surface is shown in Figure 29. Propellers, frames, and planes have different value scales and are named in the figures.

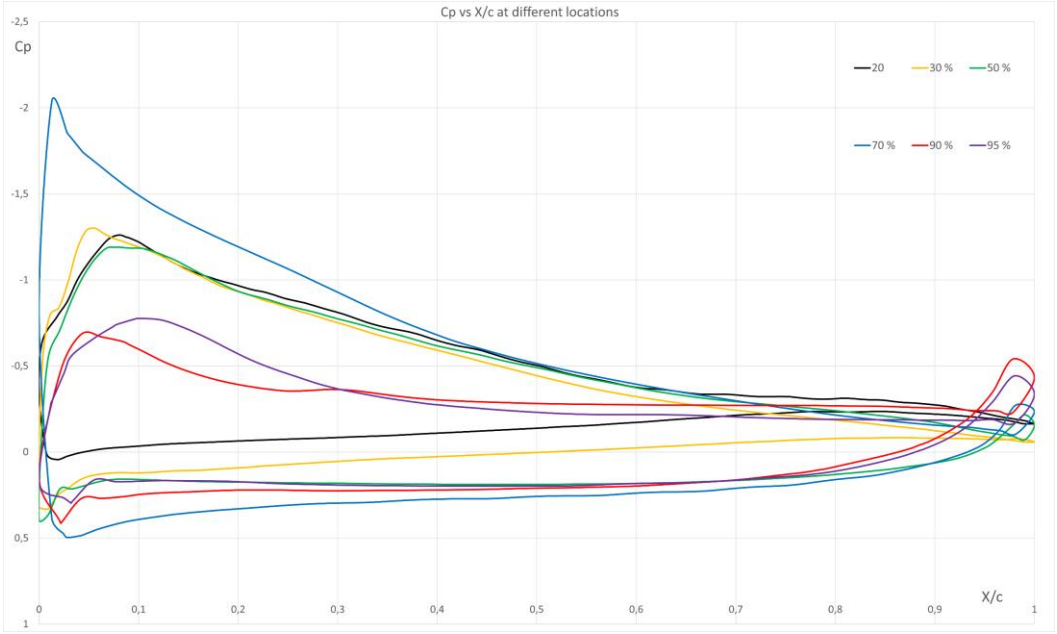

Figure 30. *Pressure coefficients chrod wise of blade for forward movement.*

Pressure coefficient at different airfoil locations is seen in Figure 30. The propeller that is examined here is the front left propeller and it rotates counterclockwise. There is some variation between propellers from what the pressure coefficient is examined and at what state of rotation it is examined. The main difference is whether the propeller is in front of the drone or in the back of the drone. Also, when a propeller blade is near the frame, it influences the aerodynamics of the blade.

From the pressure coefficient plots, it is seen that at 20%, 30% and 50% plots are very close to each other on the upper side of the airfoil, but some difference is seen on the underside of the airfoil. Pressure coefficients for the 95% and 90% locations are similar, with a slight difference from the leading edge to the 0.3 chordwise location on the upper side of the airfoil. The most significant difference is seen in the airfoil location at 70% which varies a lot from the stationary hovering case, where this location was similar to 50% and 90%. These differences might be explained with ongoing forward velocity that disturbs pressure on the airfoil surface. Pressure coefficients are plotted in one location of the rotation of the blade, which is parallel to the forward movement. And there might be quite a lot of differences in what location these are plotted. But it was chosen to only plot pressure coefficients for this one frame, as in the mesh it is not determined named locations on the airfoil surface. Without the named location, it is quite complicated to plot pressure coefficients for different rotation phases of the blade.

3.4 Aeroacoustics

To simulate aeroacoustics signals, first the microphones or receivers' locations must be determined, as well as the faces of the domain that are producing the sounds. Sound is produced mainly from the rotating propellers but also from other walls in the drone. All the walls from the drone are selected for sources. The Ffowcs Williams & Hawkings (FW&H) model is chosen, and acoustic signals are calculated simultaneously after every timestep. Density is set to 1.225 kg/m^3 and far-field sound velocity to 340 m/s.

In the standard (EN ISO 3744:2010, 2010), it is defined how sound pressure should be measured from different noise sources. In addition to that, EU regulation (EU2019/945, 2020) has some refined additions when measuring noise from drones. This regulation is meant for measuring sound pressure levels that drones can legally produce. In this work, this regulation is not directly applied because more interesting is the sound produced at higher rotational speed and with higher lift. When measuring the maximum sound produced by different drone types, it is said that the drone must be in a stationary hovering state with the MTOM, and the object should be placed 1.5 m from the ground. However, in this work, drone noise is examined with propellers rotating at 80% of the maximum rotational speed, and therefore the thrust generated is more than the weight of the drone. Also, the drone's location is set at 192 mm from the ground, measured from the center of the propellers.

Defining the receiver locations is based on (EN ISO 3744:2010, 2010) standard, where locations for 12 receivers are specified. The characteristic length of the drone is calculated by setting an imaginary box around the drone in a way that covers the whole drone and one side is on the ground. The drone's characteristic length is from the center of the face on the ground to the box's farthest corner. In this case, the characteristic length is approximately 0.4 m. Therefore, the radius is twice the characteristic length, rounded up to the next highest value in the series: 4 m, 6 m, 8 m, 10 m, 12 m, 14 m, and 16 m. Therefore, the measurement radius *r* is 4 m. Calculating receiver locations for receivers 1-8, the *z* coordinate is changed from the standard value of 1.5 m to 0.192 m because the drone's *z* location is different from the standard.

Receiver	x (mm)	y (mm)	z (mm)
1	3351	0	192
2	2369	2369	192
3	0	3351	192
4	-2369	2369	192
5	-3351	0	192
6	-2369	-2369	192
7	0	-3351	192
8	2369	-2369	192
9	2350	976	2840
10	-976	2350	2840
11	-2350	-976	2840
12	976	-2350	2840

Table 4. *Receiver locations coordinates in mm.*

In Table 4, coordinates are calculated from values and equations given in the (EN ISO 3744:2010, 2010) section F with an *r* value 4 m. Receiver locations around the drone are visualized in Figure 31.

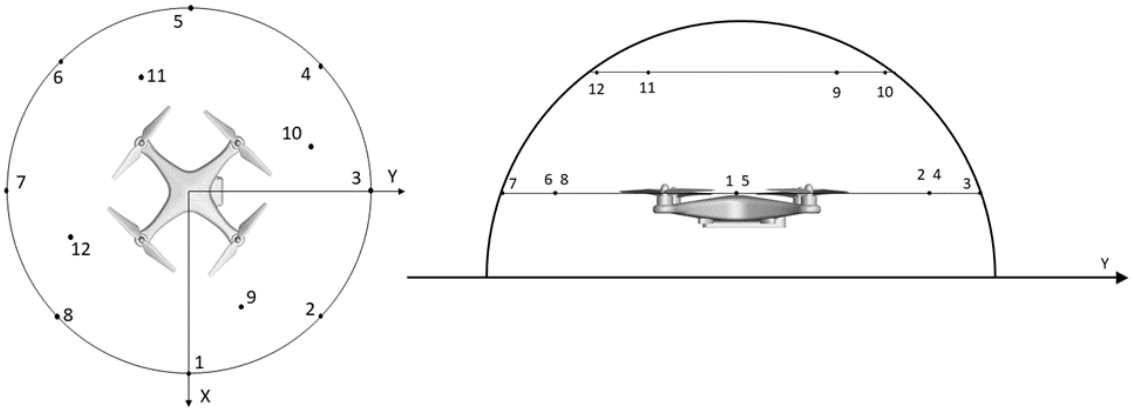

Figure 31. *Receiver locations around the drone.*

The drone, in a stationary state with its propellers spinning at 1,000 rad/s, was initially simulated for 1,200 timesteps to achieve a fully developed flow field. Aeroacoustics calculation was turned on, and simulations are continued in two patches where, first, 960 timesteps are simulated, and after that, 1,800 timesteps. So overall, 2,760 timesteps are simulated with acoustics turned on with a timestep size of 0.0000349 s. Simulation time with aeroacoustics turned on is therefore 0.096324 s, which corresponds to a calculated frequency resolution of 10.38 Hz. Simulating in two parts provides the chance to ensure aeroacoustics data is calculated correctly. When ensuring that simulated acoustic data

is correct and that all settings are set correctly, simulation time needs to be high enough to capture source data and that the plots are reasonable. With the first part, where 960 timesteps are simulated, the frequency resolution is around 30 Hz, and the data is already quite accurate.

Number	OSPL	Number	OSPL
Receiver 1	82.83 dB	Receiver 7	81.51 dB
Receiver 2	72.34 dB	Receiver 8	72.03 dB
Receiver 3	81.95 dB	Receiver 9	76.70 dB
Receiver 4	72.04 dB	Receiver 10	83.01 dB
Receiver 5	82.72 dB	Receiver 11	76.49 dB
Receiver 6	73.13 dB	Receiver 12	82.06 dB

Table 5. *Overall sound pressure levels at stationary drone.*

Ansys shows a frequency resolution of 10.62 Hz where the number of samples is 2,699. This value is a little bit lower than the simulated timesteps. Frequency resolution means at what rate sampling of acoustic signals is done. Table 5 shows overall sound pressure levels at each receiver. Comparing these results to the location of the receivers shown in Figure 31, it is seen that receivers 1, 3, 5, and 7 between propellers receive much higher sound pressure levels than receivers 2, 4, 6, and 8 that are in line with the x-shaped frame. Airflow from the propellers is directed more at these odd-numbered receivers. This flow direction can also be seen in Figure 17.

Sound pressure levels experienced by receivers that are plotted to figures are chosen based on the fact that there must be something interesting at that location. For receiver 3, airflow for the forward movement case is directed around that, and some differences can be seen between forward and stationary cases. For receiver 4, it is diagonally in line with the fuselage, and as it is seen from Table 5, it experiences much lower OSPLs than receiver 3, and some differences are also seen with the SPLs. Receiver at location 10 is chosen because it is above receiver 3 but also because it is above the drone and on the aft side. These three receiver locations are plotted for stationary and forward cases so these can be compared.

Figure 32 shows one of the first four odd-numbered receivers where most of the airflow is directed as explained before. Figure 33 shows, one of the first four even-numbered receivers. And in Figure 34 is seen one of the four receivers that are located above the drone. These three figures provide a sufficient understanding of the behavior of the SPL at various frequencies and receiver locations, making it unnecessary to display all of them.

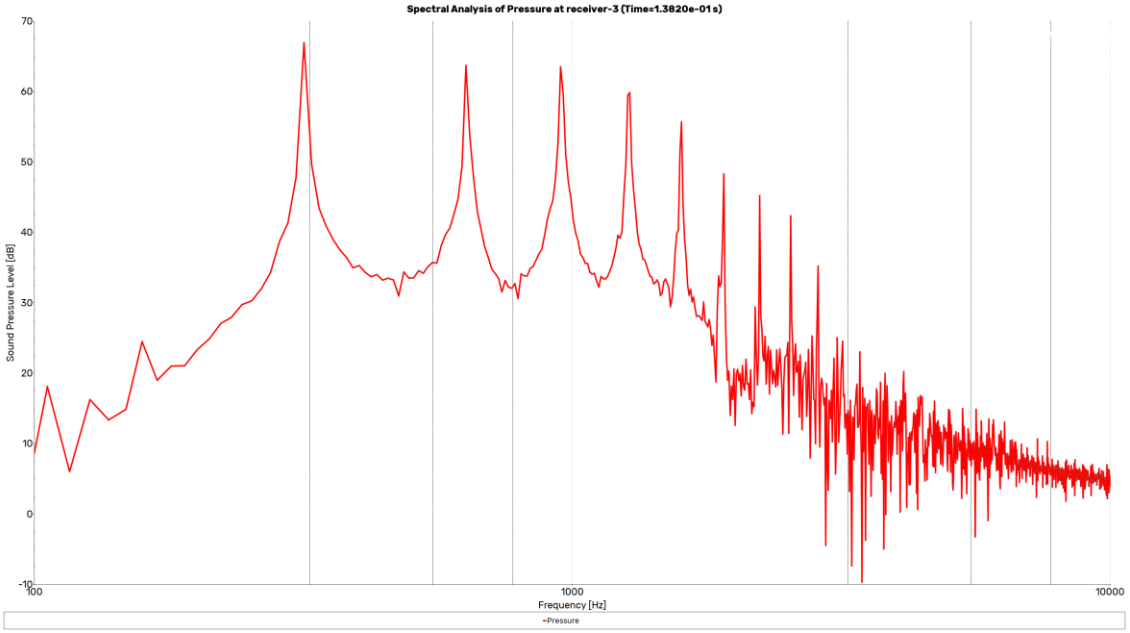

Figure 32. *Sound pressure levels experienced by receiver 3 in hover.*

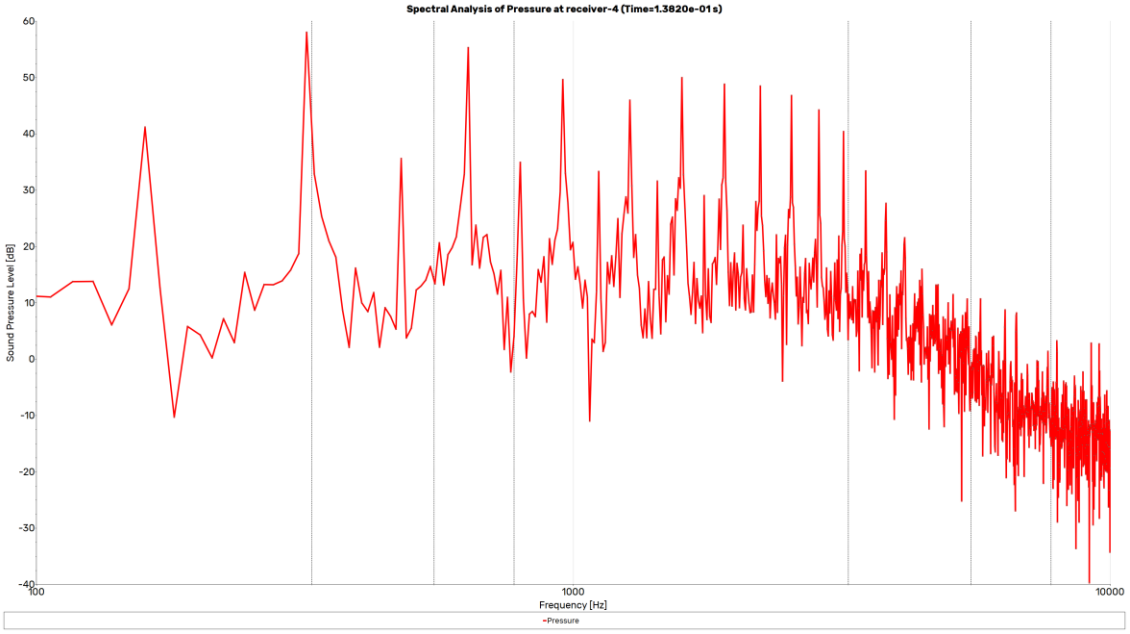

Figure 33. *Sound pressure levels experienced by receiver 4 in hover.*

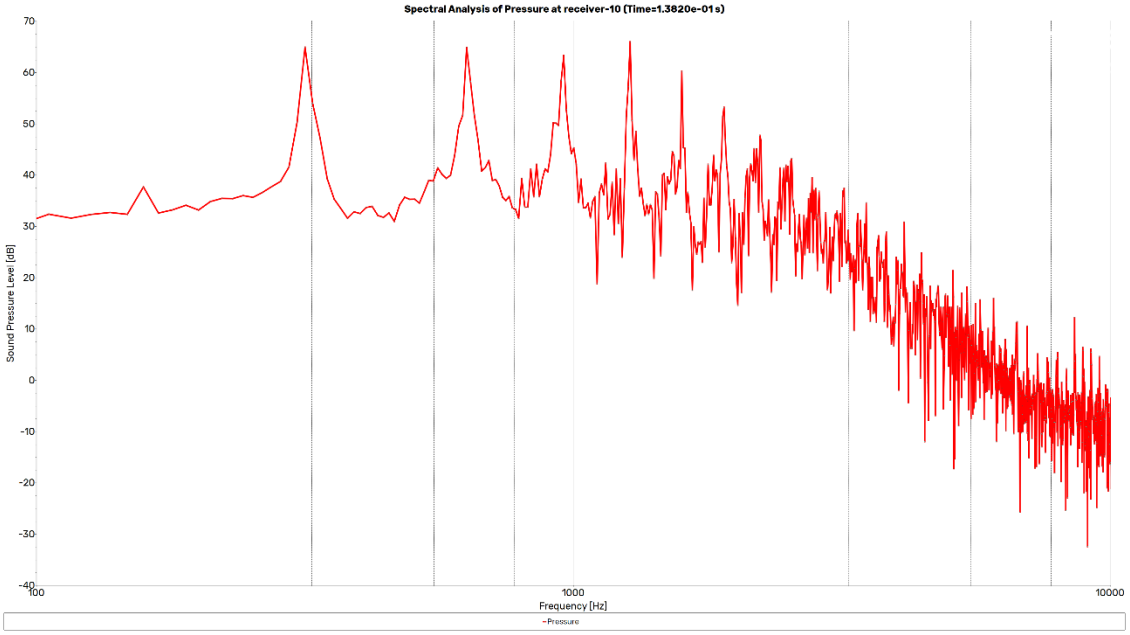

Figure 34. *Sound pressure levels experienced by receiver 10 in hover.*

Blade Passing Frequency (BPF) is calculated with equation (27).

$$BPF = \frac{RPM * n}{60} \tag{27}$$

The propeller rotational speed is represented by RPM, and the number of blades is represented by *n*. In this simulation, the calculated BPF with equation (27) is 318.31 Hz. From the sound pressure figures, it can be seen that the sound pressure spikes every 318 Hz, which is close to the calculated BPF. The first spike is the first BPF, the second is the second BPF, and so on.

In the scenario where forward motion is included to the simulation, the aeroacoustics simulation does not differ from the stationary case.

Number	OSPL	Number	OSPL
Receiver 1	83.08 dB	Receiver 7	84.56 dB
Receiver 2	78.38 dB	Receiver 8	77.42 dB
Receiver 3	80.64 dB	Receiver 9	84.49 dB
Receiver 4	76.46 dB	Receiver 10	87.16 dB
Receiver 5	83.37 dB	Receiver 11	87.04 dB
Receiver 6	76.36 dB	Receiver 12	88.60 dB

Table 6. *Overall sound pressure levels at forward moving drone.*

Overall sound pressure levels for forward motion are seen in Table 6. Based on these results, it can be said that overall pressure levels are higher with forward motion. The biggest difference is seen at the receivers located in the line of the X-shaped fuselage and the receivers located above the drone. The least difference is seen at the receivers located on the *x* and *y* axes, where receiver 3 has a lower sound pressure level in forward movement. Comparing this receiver's 3 lower sound levels to the flow pathlines shown in Figure 27, it is reasonable because airflow is directed by propellers around that receiver.

For forward movement, the same receivers that are plotted in the stationary case are chosen. This allows for a comparison of results between the two cases.

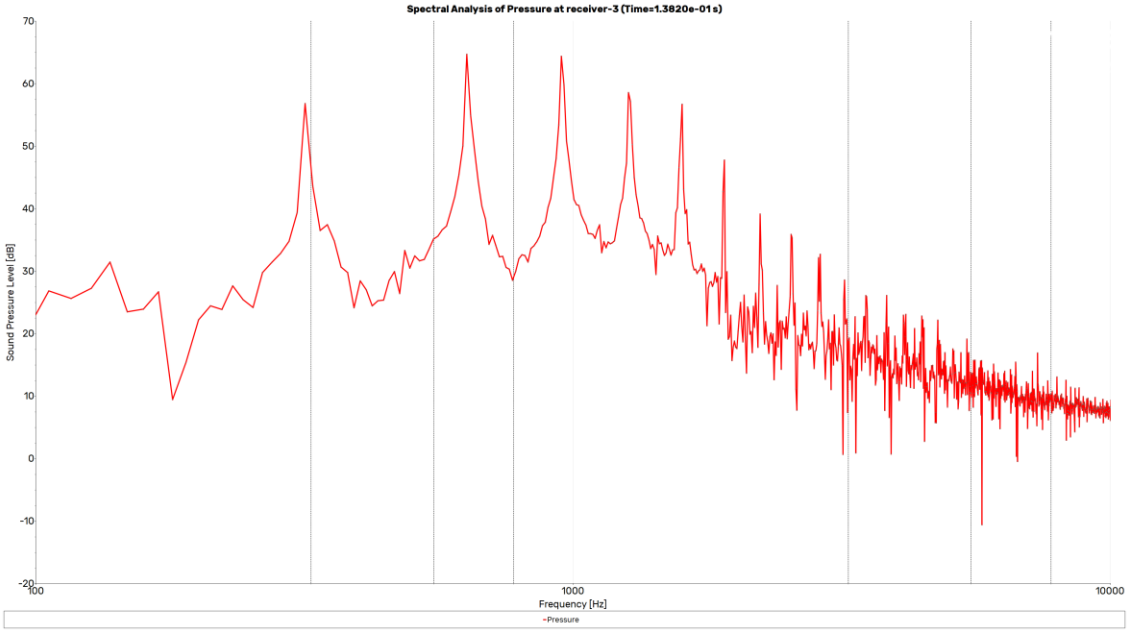

Figure 35. *Sound pressure levels experienced by receiver 3 in forward movement.*

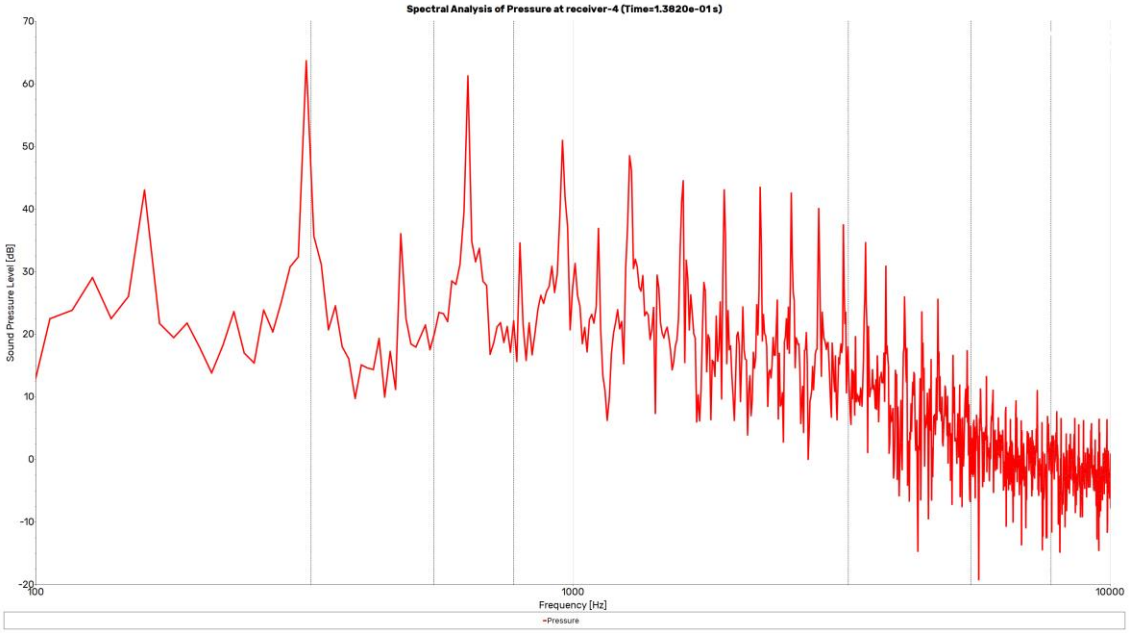

Figure 36. *Sound pressure levels experienced by receiver 4 in forward movement.*

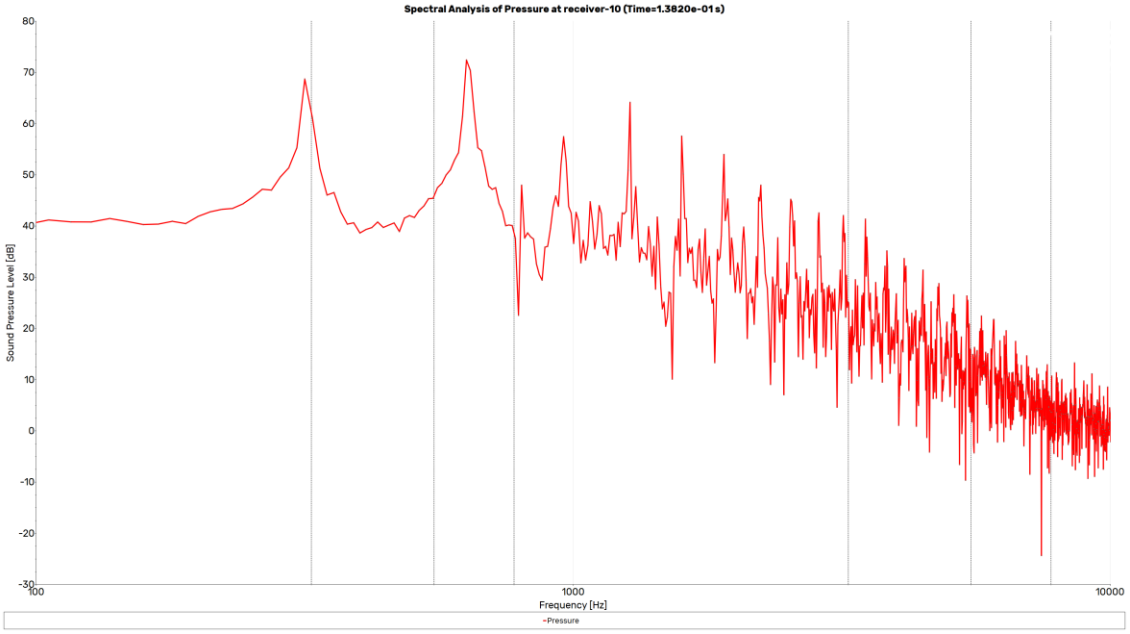

Figure 37. *Sound pressure levels experienced by receiver 10 in forward movement.*

The blade passing frequency for forward motion is the same 318.31 Hz as in the stationary case. Sound pressure level spikes are seen around every 318 Hz. Precise spike locations are hard to see from figures represented here but are confirmed in Ansys Fluent.

3.5 2D wing simulation

CFD simulation of drone airfoil is included in these CFD simulations. During QLBM simulations, it was realized that 3D simulation results and quantum simulation data cannot be compared to each other due to the limitations of the simulations at the moment.

The airfoil shape for CFD simulations is chosen to be meshed from the airfoil at 95% spanwise location, from center to tip. Flow domain is a semicircle in front and a square at the back of the airfoil. Flow domain size for the radius of the semicircle and length of the square is chosen to be 10C, where C is the characteristic length of airfoil. Mesh is constructed from 81,000 elements. Around the airfoil, sphere of influence is used to get the element size smaller than at outer region of flow domain. Inflation and edge sizing are also used to get the mesh size small enough near airfoil. Mesh size is not that crucial with low velocity, and courant number is way below one with this mesh.

Used parameters and settings for simulation are pressure-based solver and turbulence model k-omega SST. Inlet velocity is set to match the velocity used in the QLBM simulation with the highest value and is 0.0456 m/s, which corresponds to a Reynolds number of 35.6 and a density of 1.225 kg/m^3. Pressure-velocity coupling scheme is set to SIMPLEC. Spatial discretization is set to second order (upwind) and least square cell-based for gradient. URF values are left for Fluent initial values. A steady state is used at the beginning to get initial values for flow field. After initial values, transient time is turned on and 10,000 timesteps are simulated.

Timesteps used for simulation is 10,000 this matches the timesteps used in QLBM simulations also the timestep size is set to match quantum simulation time 3.62 s.

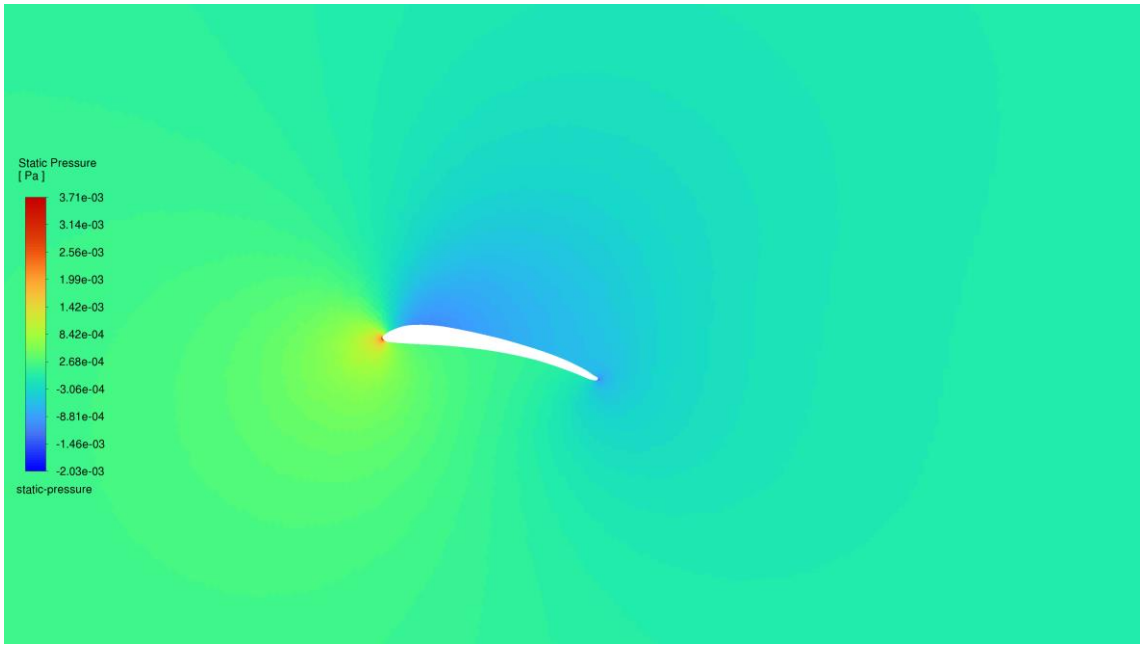

Figure 38. *Static pressure around airfoil for RANS.*

Static pressure variation around the airfoil is seen in Figure 38. From the Figure it is seen that variation is relatively low, and that is because such low velocity is simulated, and a larger pressure difference does not form. From the figure, a high-pressure point at the leading edge and a low-pressure point at the trailing edge are seen. Also, lower pressure at the upper side relative to the underside of the airfoil is seen. This pressure distribution is unusual for airfoils because such low Reynolds number flows are not simulated typically. With higher velocity high pressure point that is seen only at the trailing edge would be seen farther away at the underside of airfoil.

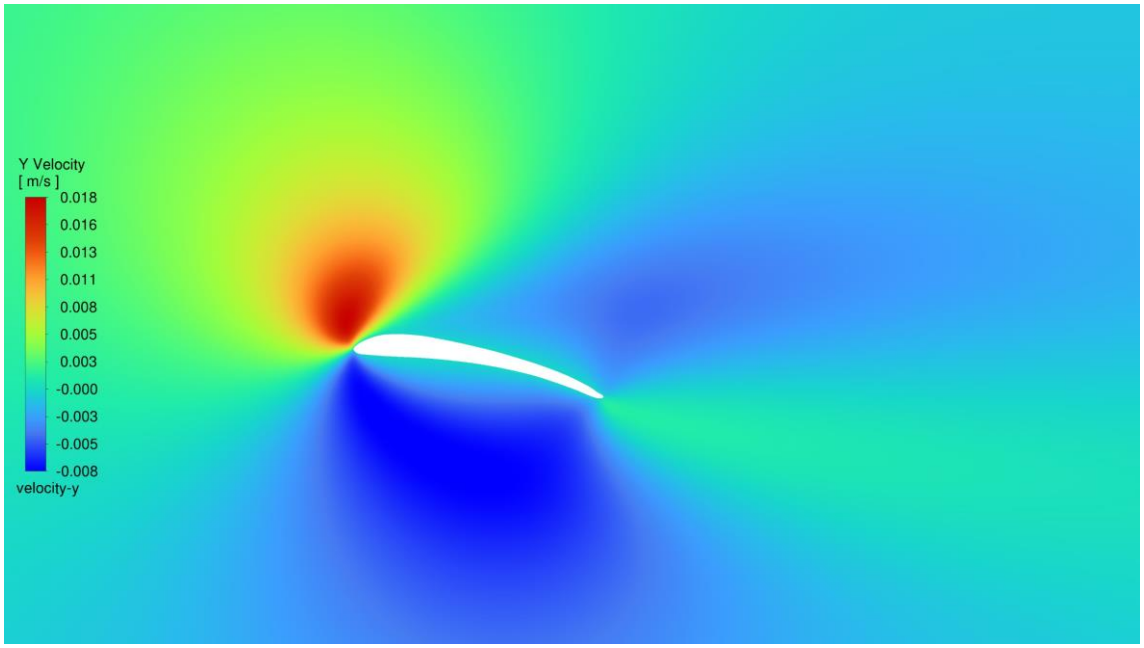

Figure 39. *Velocity to y-direction for RANS.*

Velocity in the *y*-direction is seen in Figure 39. Higher velocity is located at the leading edge, slightly above the airfoil.

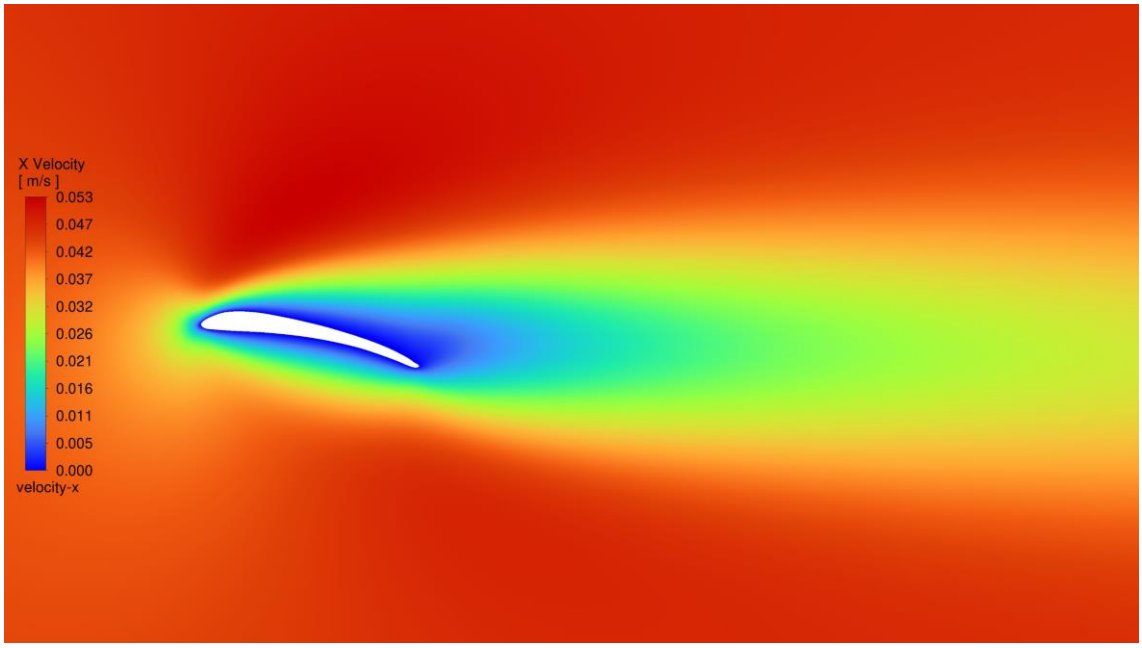

Figure 40. *Velocity to x-direction for RANS.*

Figure 40 illustrates the x-direction velocity. Majority of the velocity magnitude is the result of x-direction velocity, and not a lot of difference is seen between velocity in the x-direction and with velocity magnitudes.

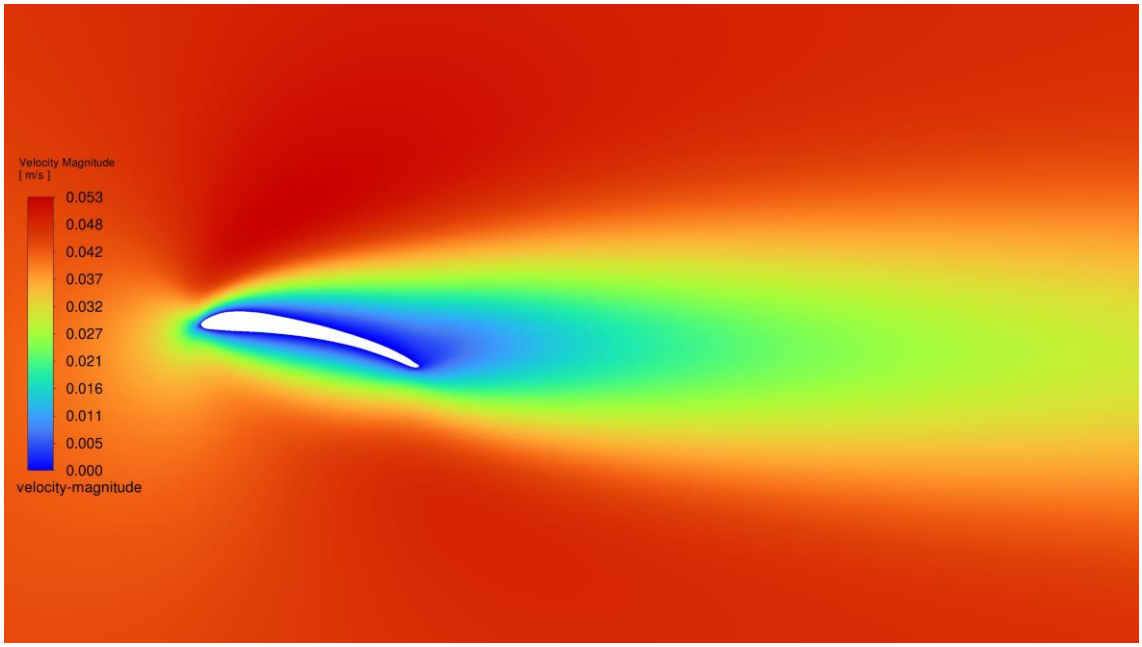

Figure 41. *Velocity magnitude around airfoil for RANS.*

Velocity magnitude is seen in Figure 41. It is seen that airflow separates from the airfoil immediately, which is expected with very low Reynolds number airflow. Moreover, the flow's trail is very wide. With flow that would produce some relevant lift from propeller on drone flow separation should be seen only near the trailing edge and mostly on upper side of airfoil. For such a flow the trail of the flow should be much narrower.

With steady state, convergence is achieved with 1,974 iterations with residual values set to 1E-6. Simulation time for this with 128 cores is 1 min 18 sec. Transient time is turned on, and simulation achieves convergence instantly. This is because there is no rotation or any other changes in mesh. Also, velocity is so low that no turbulence forms the flow is considered as laminar flow. The simulation time for 10,000 timestep is 16 min 57 sec. Residuals are very low from first iteration, velocities in the scale of 1E-11 and continuity 1E-7.

It is noted that these CFD simulation results of the airfoil do not satisfy the needs of the drone to achieve flight condition. Although these simulation results are needed to compare traditional CFD and QLBM at the moment.

4 QLBM SIMULATION

In this section of the thesis, quantum simulations are conducted. Simulations are done with the Quancients QLBM simulation software. Quancient is developing software that uses QLBM to simulate CFD problems. These simulations utilize an early version of the software, which is currently under development. Due to the early state of development, it is limited in some sections. Simulations are performed by the company due to the state of the software. Results from the simulations are plotted with simple code using Matlab.

Using QLBM to simulate CFD problems is a new method. In these simulations, a similar 3D case that was simulated with traditional CFD is not possible. At the moment, the state of development allows 2D simulations. And currently, only low Reynolds number flows can be simulated. It was estimated that simulating airflow over an airfoil at a velocity of 114 m/s, would use around 26 qubits. To simulate the case using 26 qubits is possible with HPC computers but is computationally demanding. Additionally, there are other problems involving higher velocity simulations, such as boundary condition problems that could not be fixed yet.

To simulate 2D case, an airfoil from the DJI Phantom 3 propeller is chosen for the simulation. The airfoil chosen for the simulation is located near the tip. It is the location where vortices form in 3D simulations and is therefore intruding to simulate. More precise location is 95% from center to tip.

Generating a lattice structure for the problem starts by generating an airfoil shape. At the moment the lattice structure is generated from a picture of an airfoil. The picture of the airfoil is obtained from a 3D CAD model of the quadcopter.

QLBM simulation data is compared to LBM simulation data with the same parameters. This ensures that the results are similar and that there are no errors in QLBM simulations. Although there are some differences in these simulations, lattice size is different for these two simulations. The need to use a low Reynolds number in simulations comes from the lattice dimension's length. With higher velocities, problems occur at the boundaries.

Simulation output data is density and velocity in the *x* and *y* directions. To plot these variables, the simulation is also set to output the location of the airfoil in the domain, physical simulation time, timesteps, and lattice coordinates in the *x* and *y* directions.

For both LBM and QLMB pressure, velocity magnitude and velocity in the y- and x-directions are plotted. Pressure is calculated from density by the equation $P = C_s^2 * \rho$, where ρ is density from simulations relative to reference density and $C_s^2 = 1/3$ is the lattice

speed (Krüger et al., 2016, pp. 112 & 272). Different lattice speeds are shown in Table 1. Velocity magnitude is calculated from *y* and *x* components.

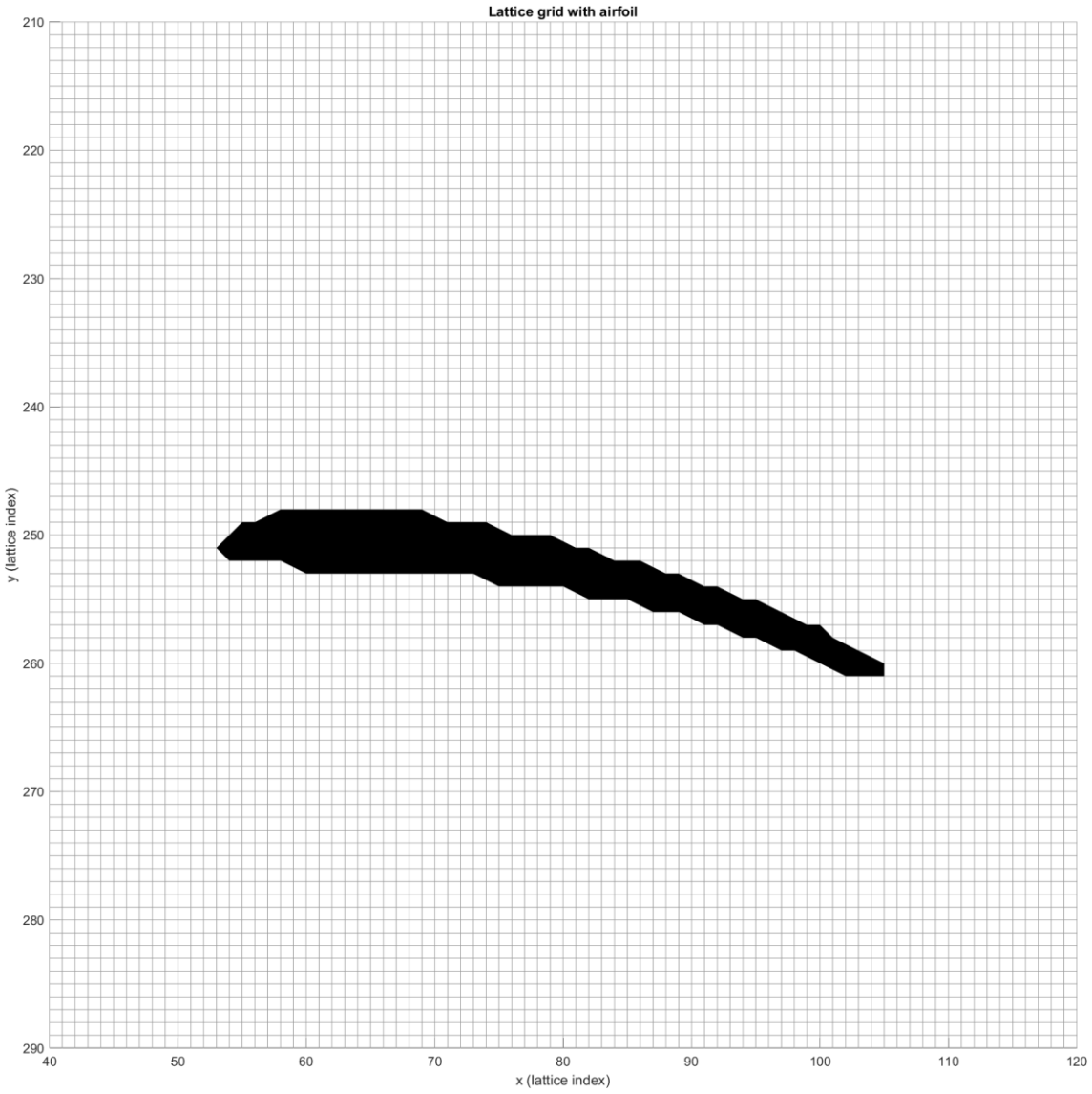

Figure 42. *Lattice structure near airfoil.*

Flow domain is generated with different nodes as it was shown in Figure 3. Lattice structure generated around the airfoil is seen in Figure 42, where the airfoil shape is filled. In this flow domain only solid nodes (airfoil) and fluid nodes are presented. The figure shows the resolution of the airfoil used in the simulation, and it can be seen that the airfoil and lattice structure are very coarse.

For LBM simulations the number of lattice nodes is set to 512 to *x*-direction and to 1,024 to *y*-direction. This results in a total of 524,288 nodes, of which 229 are solid airfoil nodes. Reynolds number is 35.6 with velocity 0.0456 m/s and timesteps simulated is 10,000 with overall simulated physical time 3.62 s. Data is saved every 1,000 timestep.

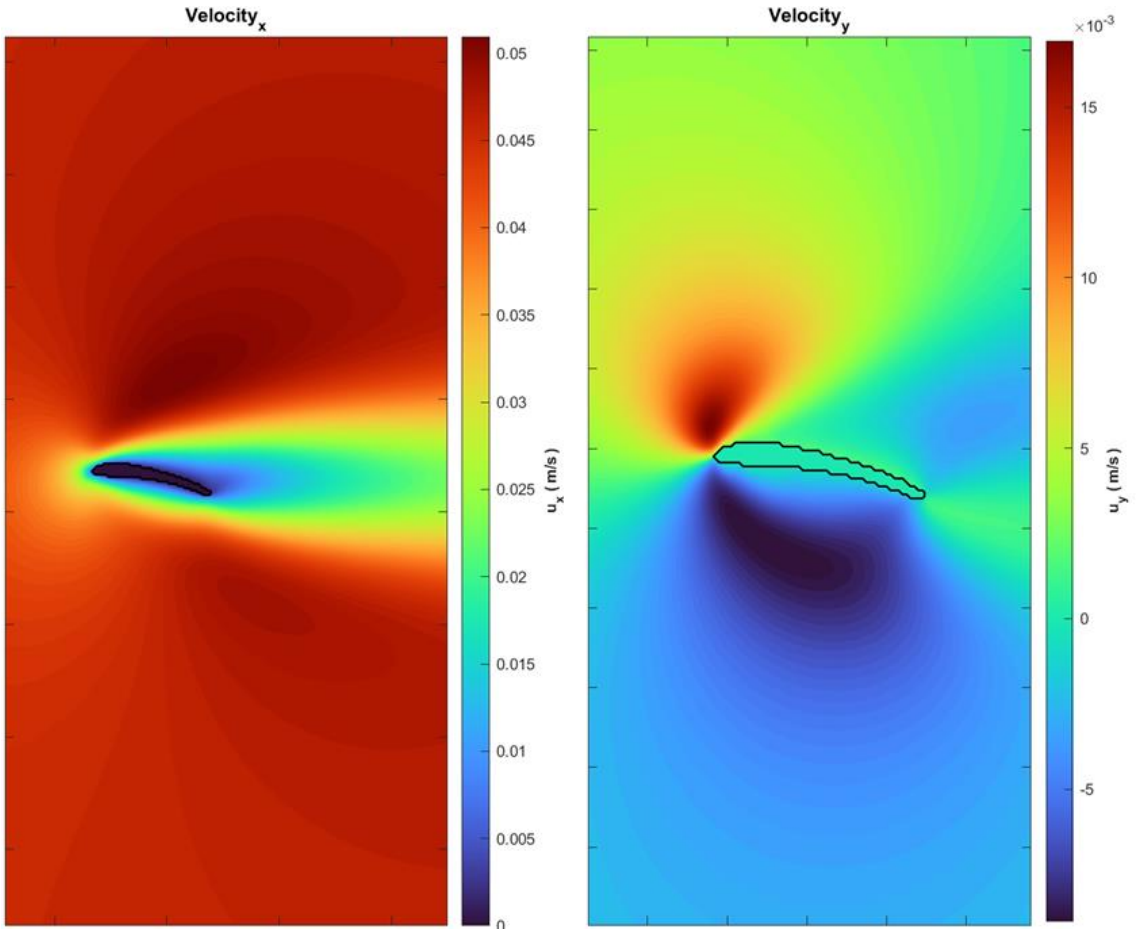

Figure 43. *Velocity to x- and y-direction for LBM.*

From Figure 43, velocity components in both directions are seen. On the right-side velocity values in m/s are shown that corresponds to color scale on figures. The contours indicate that most of the flow occurs in the *x* direction, while significantly less flow occurs in the *y* direction. Airfoil shape is also shown to better visualize flow.

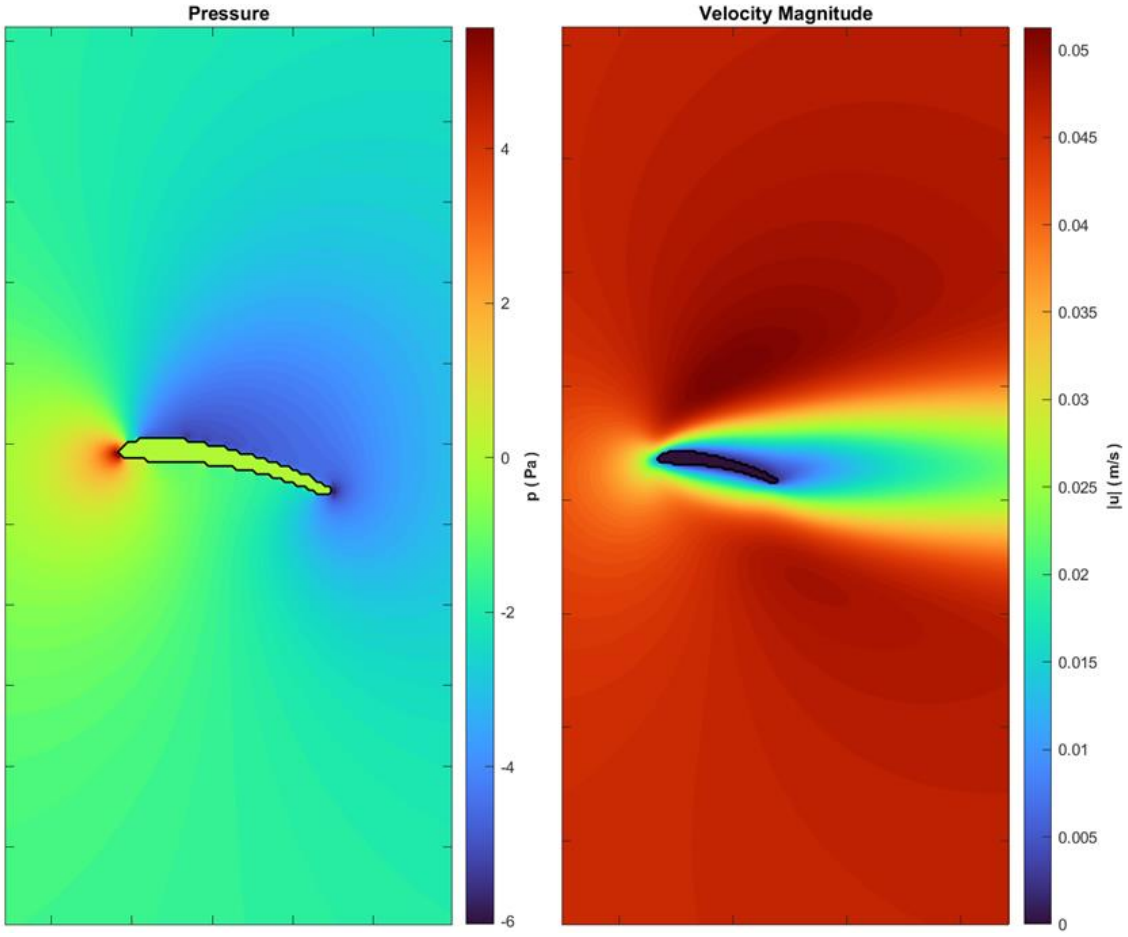

Figure 44. *Pressure and velocity magnitude from LBM.*

Figure 44 displays the pressure and velocity magnitudes surrounding the airfoil. Pressure in Pa is seen on the left side figure and velocity on right side figure in m/s.

Data is saved every 1,000 timesteps. In total, data is saved 10 times. Checking when the solution is fully developed is not very accurate, but some estimates can be made. Comparing values of every timestep to previous timestep values and analyzing maximum values of those presents at what timestep the solution is fully developed. Also, analyzing animated flow and pressure fields supports this. After 5,000 timesteps, the solution is fully developed.

Simulation time for LBM simulation is approximately 4 min with 8 core computer. The simulation is run on Python, where JAX 0.7.0 library is used to accelerate the calculations.

QLBM Reynolds' number is also 35.6 with a velocity of 0.0456 m/s. Simulation consists of 10,000 timesteps with an overall physical simulation time of 3.62 s. For QLBM simulations, the number of lattice nodes is set to 256 in the *x*-direction and to 512 in the *y*-

direction. This results in a total of 131,072 nodes. Out of the total nodes, 229 are designated as solid airfoil nodes. Domain size is smaller than it was for LBM, but airfoil size is the same in terms of nodes. Data is saved every 100 timesteps. This gives a better possibility to make an animation of the developing flow than with the LBM. Also, checking when the flow field is fully developed is more precise. Results for QLBM are combined with the RANS simulation figures to better visualize the differences and similarities.

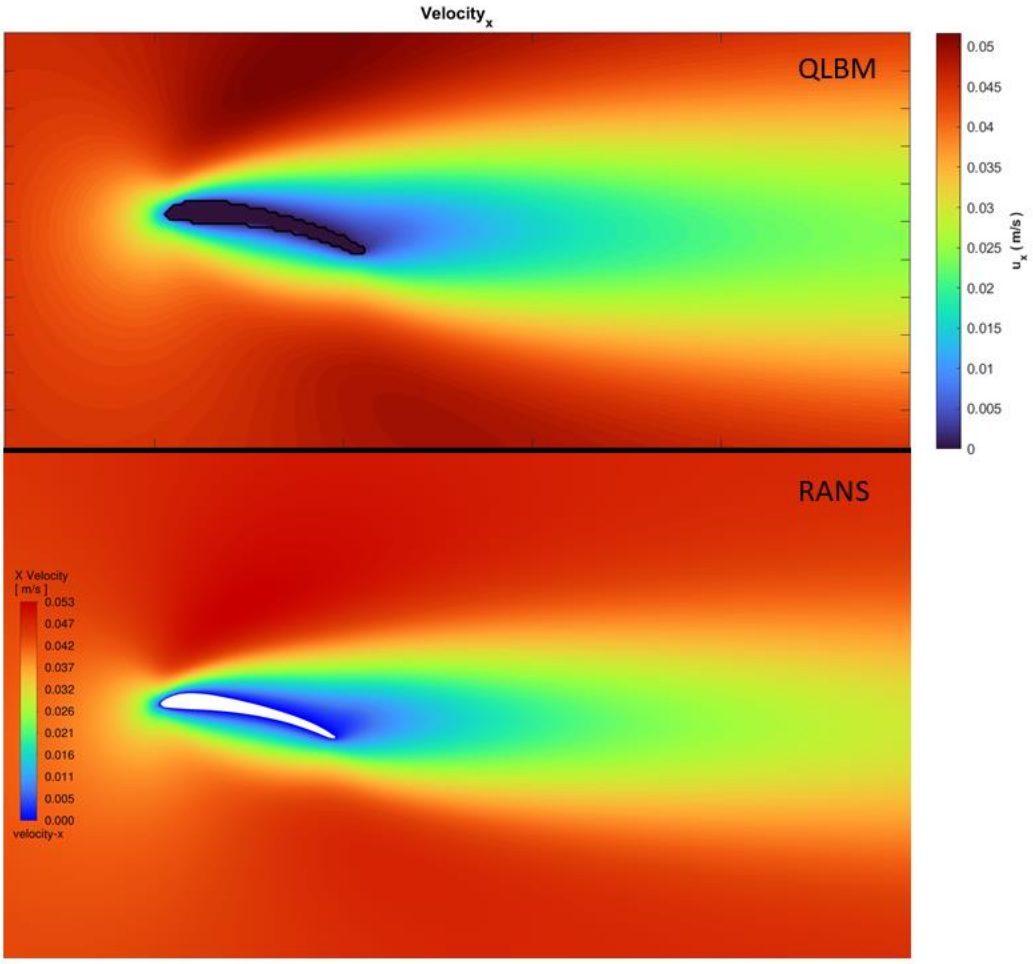

Figure 45. *Velocity to x-direction for QLBM and RANS.*

From the Figure 45 ,the velocity components to *x*-direction are seen. For both figures velocity is shown in m/s. Color scale for QLBM is slightly darker for maximum value shown in red and for minimum shown in blue. From these figures it can be seen that velocity contours are pretty similar and not a lot differences can be seen.

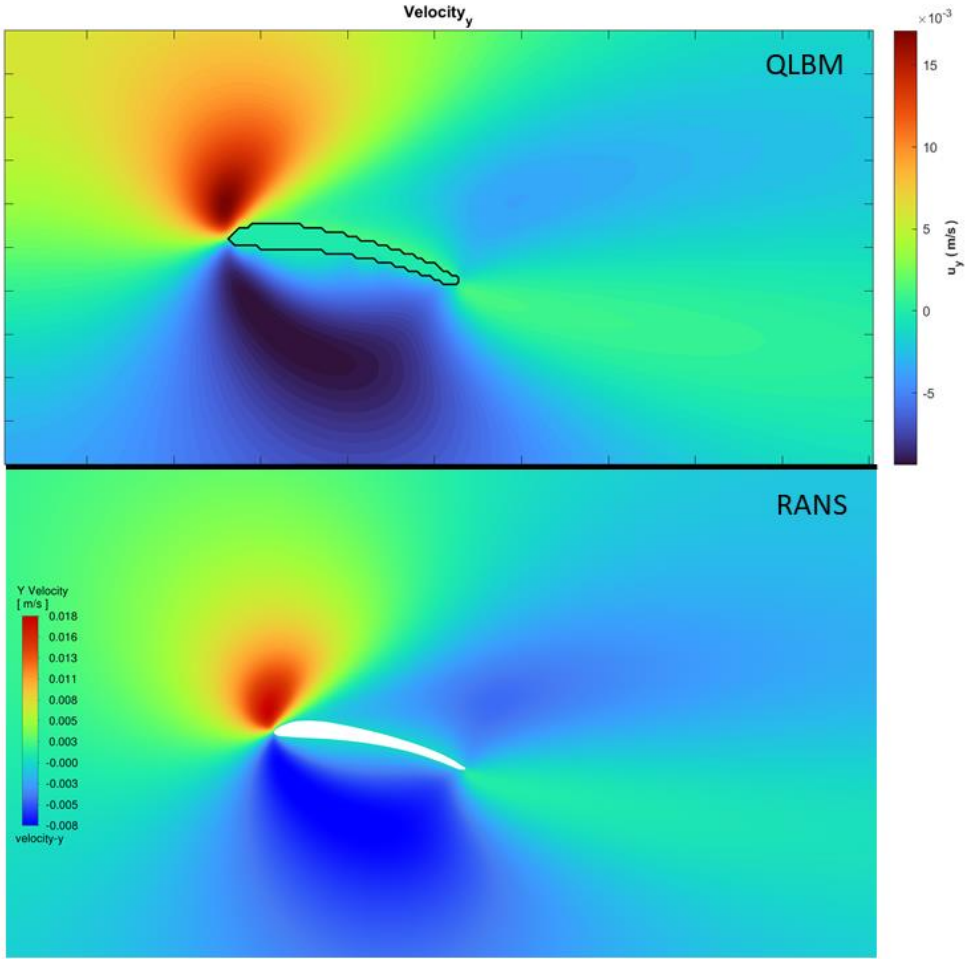

Figure 46. *Velocity to y-direction for QLBM and RANS.*

Figure 46 shows the velocity to *y*-direction. Maximum and minimum value colors are slightly different. Velocity scale is seen for QLBM on the right side, where values are displayed as exponent. Scale for RANS is seen on the left side and numbers represented as float numbers for both velocity is in m/s. Velocities does not show significant difference.

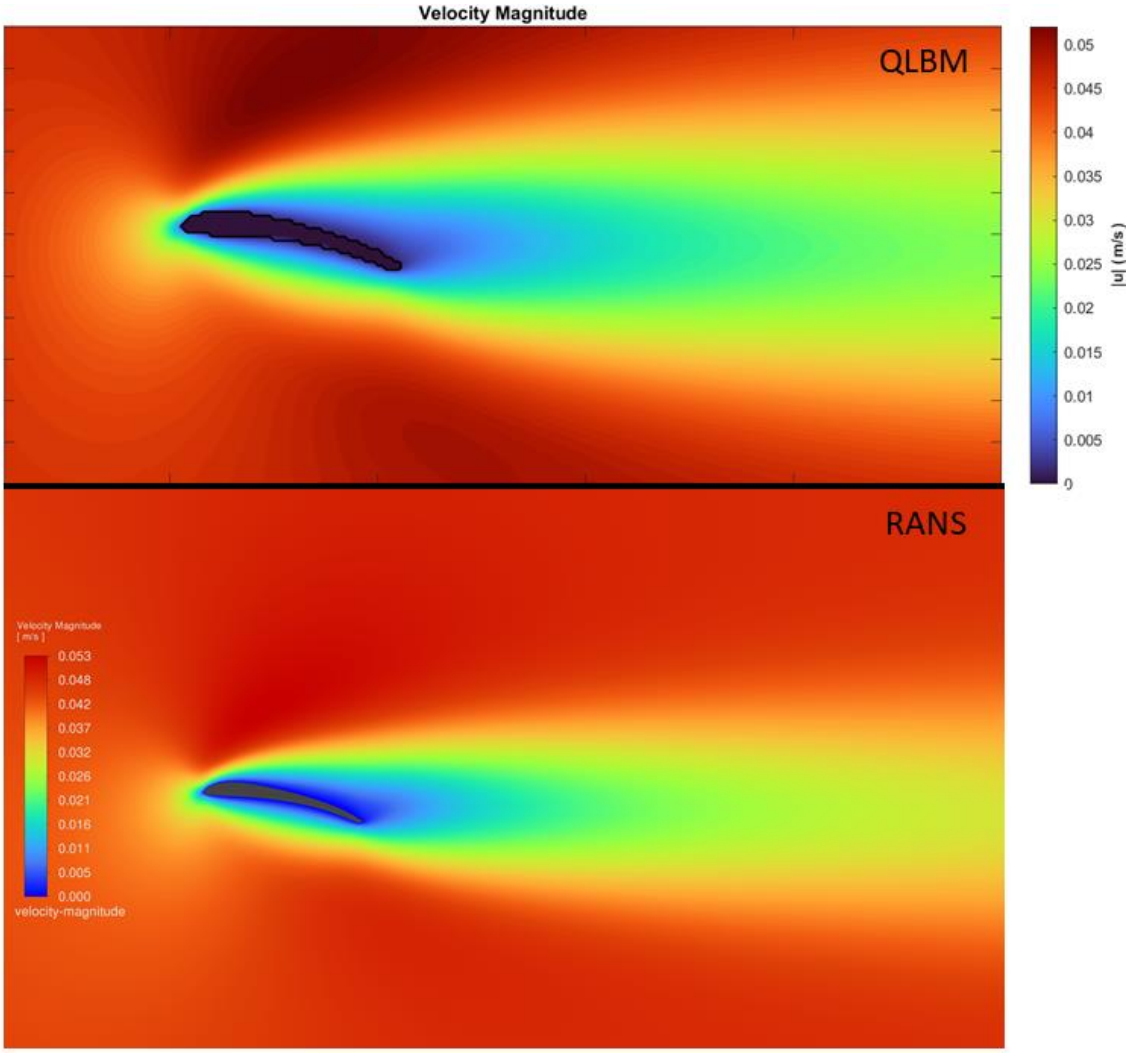

Figure 47. *Velocity magnitude for QLBM and RANS.*

Velocity magnitude for QLBM is seen on the top of Figure 47 and on the bottom for RANS. Where velocity values are presented in m/s. As it was for velocity components, that not a significant difference in contours was seen. Similar excepted results is seen in the velocity magnitudes.

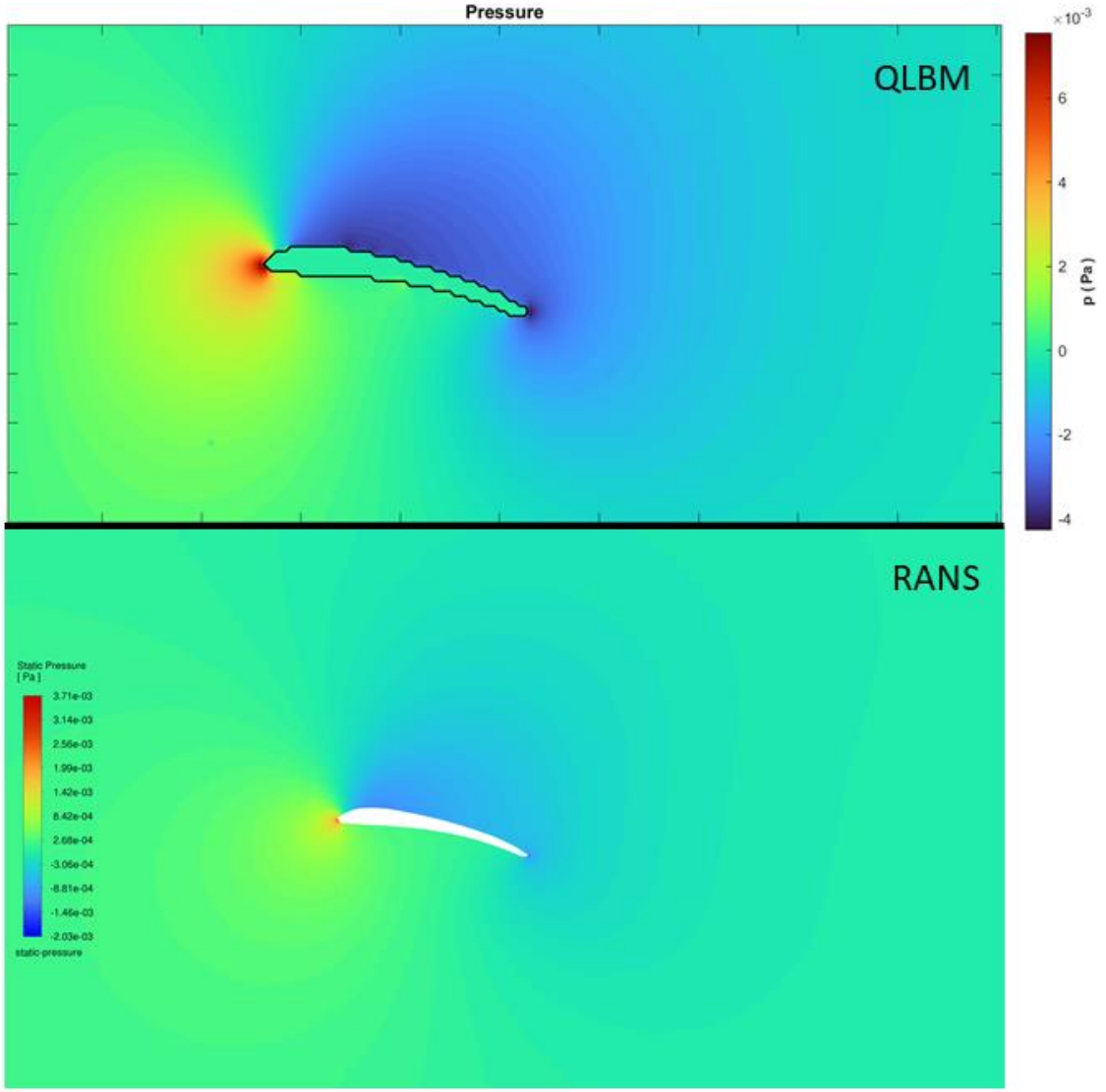

Figure 48. *Pressure for QLBM and RANS.*

Pressure contours are shown in Figure 48 where static pressures are shown in Pa. There is a slight difference in the pressure scale between these figures. Results are not expected to be the same, as RANS is an average method and RANS is a more precise method. The range of results are in the same scale. Another difference that can be seen is the high- and low-pressure points that are seen in greater area. Contour for QLBM is plotted with Matlab where with RANS it is plotted in Ansys. This might add some difference to the variation that is seen near leading and trailing edge.

With QLBM, at what timestep the flow field is fully developed was analyzed similarly to the LBM, with the difference being that data is saved more often. Comparing values of two consecutive timesteps, it can be seen that after 4,500 timesteps, changes in velocity and pressure are small enough to consider the flow field to be fully developed. This conclusion can also be confirmed from animated pressure and velocity contours that are

plotted. After 4,000 timesteps, no significant change is visible. Although this is an inaccurate method to say exactly when the flow field is fully developed, it gives a rough estimate and supports the previously calculated values.

Simulation time for QLBM simulation was 28 hours with 16 core. Calculation is run through Qiskit emulator. From the simulation time of QLBM and LBM it can be said that running simulation in quantum emulator significantly increases the simulation time. This is excepted result as the code is developed to run on quantum computer. Quantum emulator is a tool that allows these kind of quantum simulations to be run without the need to use real quantum computer.

In these simulations some errors and mistakes occurred. Some major errors occurred with the boundary conditions, which caused the flow to behave unexpectedly. These errors were fixed after first simulation. One of the minor errors was that the orientation of the flow field was flipped 90 degrees and was fixed during simulations. An error that was noted after simulations was that the y velocity field showed high velocity at locations where there should be low velocity and vice versa. Also, the absolute values were flipped. This noted to be sign error in formulas and was fixed by multiplying y velocity values by negative one. These errors did not affect the results shown in this work.

5 RESULTS

In this part the results are evaluated. Different cases and simulation methods are compared.

5.1 CFD Aerodynamics

CFD simulations were performed in the Mahti environment with 128 cores enabled. CPUs used in Mahti are AMD Rome 7H12 containing 64 cores. Each node has two of these CPUs.

With transient stationary case simulation convergence was obtained with 1,080 timesteps and 26 h 42 min and with the forward case, 1,260 timesteps and 31 h 9 min. CPU hours are calculated by multiplying the number of CPU cores by the simulation time. For stationary, this CPU hour is 3,420, and with forward, 4,000 rounded to tens of hours. Overall simulated CPU hours for each case are 12,890 and 13,400, rounded to tens. At first, the convergence solution must be simulated before acoustic sampling can begin, about 70% of this simulation time is used for aeroacoustics.

Validating aerodynamic CFD results is done by comparing CFD results to other studies but also analyzing the flow field and its characteristics. Also, the stationary case is compared to the forward case.

Stationary simulation is close to real-life case. The difference from reality is that when propellers are rotating at 80% of maximum rotational speed, the drone would start moving upwards. In these simulations, the drone is at a fixed point. Forward motion simulation is similar to stationary simulation in that the drone would start moving upwards with the set rotational velocity. With forward velocity, the ground would be fixed and the drone would be moving relative to it. In these simulations, the ground and the drone do not have relative movement. Also, to achieve forward velocity, the drone should be tilted forward as was said in the theory part about the drone movement. The drone is stationary relative to the ground, and therefore the forward simulation is closer to wind tunnel experimental than it is to movement on the ground.

Comparing the forward case from Figure 22 and stationary case Figure 12 Q criterion iso-surfaces, it is seen that the tip vortex in the stationary case are more uniform than

those in forward motion. The tip vortices are spread out more in the forward case when examining those with same minimum and maximum values. The tip vortices are moved by the oncoming flow similar effect is seen near the fuselage. To better visualize how the forward motion affects the vortices, the 2D views of the vortices are compared. In Figure 14, Figure 15, Figure 24 and Figure 25 that presents the cross-sections it is seen that in the stationary case, vortices are moving down. With forward motion, vortices at front propellers seem to stay at propeller level and interact more with the rotating propeller, and in the back, vortices are pushed down almost like in the stationary case.

Velocity contour lines in both cases are kept as minimal as possible, but the way it still shows well the flow direction before and after the propellers. To generate these types of velocity pathlines, the rake tool is used, where the number and locations of starting points of velocity pathlines are specified. Figure 16 and Figure 17 show pathlines in a stationary solution where lines are released from the upside of the drone. This visualization gives an overview of how the flow field behaves near propellers but also what the effect of the ground is to the flow. Figure 26 and Figure 27 show velocity pathlines in forward motion. Releasing velocity lines from the same location as in the stationary case would not show behavior near or under propellers, so pathlines are released from the front of the drone. This represents better the flow field around the drone. Similar pathlines are shown in the study by (Dbouk and Drikakis, 2022) , where the forward and stationary cases are also simulated, and velocity pathlines show similar flow phenomena. However, in that study, ground effect is not simulated.

Static pressure figures show pressure variation in different areas of the drone. 2D cross section is added in the same figures to better visualize pressure differences. From Figure 18 and Figure 19 , pressure distribution in the stationary case shows how higher-pressure forms under the drone. Tip vortices are also seen in the same figures. Those are considered low-pressure dots around the higher-pressure area. With the forward case, higher pressure is seen in Figure 29 under the drone, and although it is not as clear as in the stationary case, tip vortices are also seen as lower-pressure dots. Figure 28 shows a lower pressure area between propellers.

As shown before, pressure coefficients are calculated by equation (26), which contains the reference velocity that is changed depending on the location where Cp is calculated. For both cases, the value that is used as a reference velocity is the propeller velocity, not the flow velocity near the propeller. Pressure coefficients in Figure 20 show big variation at the leading edge, and at the bottom side of the blade a bigger spike is seen. Near the trailing edge, values change from positive to negative and vice versa around 0.95 X/c. With forward motion in Figure 30, some difference is seen compared to the

stationary case. The pressure coefficient at the underside of the leading edge does not have the same type of spike as the stationary case, and at the 70% spanwise location, a spike is seen on the upper side of the blade.

A quadcopter at stationary hovering with a propeller rotational speed of 1,000 rad/s produces an overall thrust of 42 N. This thrust value is the average value of the thrust produced with one rotation. The fluctuating range of the thrust with one revolution is around 38 N to 47 N. In forward motion, the average thrust value is 44 N and the fluctuating range is 38 N to 54 N. The increase in produced average thrust is 4.8%. Similar results are seen in the study by (Paz et al., 2020), where different forward velocities are studied when moving over an obstacle. Increase in thrust is seen with higher forward velocities. In that study, the stationary case is not studied, but with forward velocities of 1 m/s and 10 m/s, a 25% thrust increase is seen when moving close to ground. Even though these two studies are slightly different, it can be said that a 4.8% increase is much lower than 25%. The difference might be explained by the fact that in this study forward movement is simulated with only incoming airflow and the ground is stationary relative to the drone, whereas in their study the drone is moving compared to the ground. This was one of the compromises or mistakes that were made during simulations. During the simulation it was noted that there are other methods to simulate forward motion, one of these methods is to generate second mesh. Second mesh would have contained the ground and these two meshes would have relative movement. But it was noticed that generating this type of mesh and get it to work with the original mesh was too complicated.

In CFD simulations, multiple methods exist to determine the size of the timestep, as previously mentioned. In this work, time-step size was chosen by testing different values, and during simulations, residuals and aerodynamic characters were observed. Also, after simulation, the Courant number was observed at different locations. The mistake that was made with the time-step size is that it was chosen too early when not enough research was done about it and was not changed to be smaller afterwards. There are some studies where rotating propellers are studied and where timestep size corresponds to revolutions of 1 deg (Kelecy, 2021) and 2.2 deg (Chen et al., 2023). These values cannot be directly compared to each other or to this study because there is no mention of cell sizes. It seems that in multiple studies, around one degree of rotation per timestep is used. Courant number in these simulations is over one at some locations of the mesh which indicates of too large timestep.

5.2 CFD Aeroacoustics

Aeroacoustics from CFD simulations is analyzed and validated by comparing the results to similar studies. Also, a comparison between the stationary and forward cases is made.

Peaks at BPF are recognized as periodic noise introduced in chapter 2.2.1. In the simulations, all propellers rotate at the same speed. This generates one periodic noise peak per BPF. In reality, propellers rotate at slightly different speeds, and therefore, in theory, there might be up to four peaks per BPF, depending on the rotating speed differences. Although even in reality rotational speed differences between propellers in stationary hovering cases are quite small. A larger difference in real-world cases would be seen with a forward-moving drone where the front propellers are rotating at a lower speed than the propellers in the back. In these simulations, propeller rotational speed difference is not considered, and there might be some inaccuracy in the results of sound pressure levels.

The broadband part is introduced in chapter 2.2.2 and is found between the periodic peaks. In the simulation there is no background noise or noise from motors. In the simulations, the accuracy of frequency resolution affects the broad band noise. The difference in frequency between two BPFs is 320 Hz, and with a frequency resolution of 10 Hz, there are around 30 points where SPL to frequency is measured. By simulating more timesteps, resolution would get smaller and frequency to SPL more accurate, but the accuracy is already at a good level. Other studies support the use of this accuracy where the same frequency resolution is used (Kelecy, 2021). Although there are studies where lower frequency resolution is used, it depends on the purpose of the study.

Background noise and motor noise effects compared to numerical simulation are studied by (Vieira et al., 2015), where a single propeller is used. This study gives good indication what role motors and other background noise have compared to simulated results. Experimental measurement is also done in their study comparing motor noise to motor and propeller noise, and it is noted in their study that the motor has great influence at higher frequencies to SPL.

Comparing forward flight and stationary SPL figures, it is seen that with forward flight, slightly higher SPLs are reached overall at all frequencies. The most significant difference can be seen at lower frequencies. As an exception, the receiver 3 location where SPL are lower. The same phenomenon is seen in OSPL and can be explained by the direction of the flow field. Similar results are seen in the study by (Dbouk and Drikakis, 2022), where forward movement is also shown to produce higher SPL, especially at lower frequencies.

5.3 Classical CFD and QLBM comparison

Classical CFD 3D simulation cannot be directly compared to QLBM 2D simulation because simulating the whole drone ground effect plays a major role. Also, in those simulations much, higher velocity is simulated at the desired airfoil location. One of suitable method to compare 3D simulation and 2D simulation would be comparing the pressure coefficient around the airfoil, which was shown in drone flow simulations. However, there are multiple differences in these two simulations, and therefore a comparison would be inaccurate. Additionally, at low velocities, the pressure difference on the airfoil surface is relatively small compared to the surrounding pressure, which may also lead to some inaccuracies.

With the LBM and QLBM simulations, it is shown that quantum simulations provide very similar results compared to the classical LBM. Simulating with the LBM is crucial to get understanding that using the quantum states models the streaming and collision properly. Comparing simulation output data that is shown in Figure 43, Figure 44 for LBM and for QLBM Figure 45, Figure 46, Figure 47 and Figure 48 it is seen that the results are very similar, and the only difference that can be seen is with pressure, where absolute values of min and max are higher with QLBM. This difference might be the result of another difference in these simulations, which is the flow domain size, which is double in classical LBM, as it was shown before.

The airfoil used for both traditional CFD and QLBM is the same, but after mesh generation there are some differences, where with quantum simulation the airfoil shape is much coarser. Flow domains also have some differences, where with traditional CFD, a commonly used flow domain shape is used. With QLBM, a rectangle-shaped flow domain is used, and where it is vertically bigger than in horizontal direction. The main reason the flow domain was chosen to be oriented this way is that boundary errors occurred when it was the other way around.

When comparing velocities from traditional CFD simulation to the QLBM results in Figure 45, Figure 46 and Figure 47, it can be seen that there are not many differences when comparing flow fields overall. Although some differences can be seen near the airfoil, and this comes mainly from the mesh resolution. High-velocity and low-velocity areas forms in the same locations for both simulations, and absolute values for minimum and maximum velocities are very close.

With pressure in Figure 48 both contours look very similar. With both simulation methods higher pressure occurs at the trailing edge and low pressure at the leading edge. Also, from both figures, lower pressure on the upper side of the airfoil compared to the lower

side occurs similarly. For both contours the result scale is in the same range. With QLBM slightly higher absolute values are seen. With QLBM pressure is not calculated directly. It is converted from density to pressure. This should not add inaccuracy to results, but there is a possibility that wrong lattice speed is used. Differences can be explained with the way values are calculated in RANS and QLBM. RANS is an average model, whereas QLBM is more precise.

The output data from the QLBM simulation is limited, and there is no direct method to check the convergence of the solution. Velocity and pressure values were compared between two timesteps. With this method the time when the flow field is fully developed can be found. Also, animation of the flow field was analyzed, and from those same results were seen. This method works well for flows where turbulence and other fluctuations in the flow field do not appear. At higher velocities, there is a significant amount of fluctuation in the values throughout the entire simulation. With traditional CFD, convergence is checked differently because it can be checked during simulations and with different variables, similarly to the simulations made for the whole drone.

In traditional CFD simulation is started with steady state simulation to get initial values, and simulation is stopped at convergence. After steady simulation, transient time simulation is done to obtain the same simulation time as it was in QLBM. This step was found to be unnecessary because there is not any fluctuation or turbulence in the flow field that could be captured after the flow field is fully developed. The purpose of conducting transient time simulations was to achieve results that are more comparable to those produced by QLBM. The simulation is similar to transient time simulation from beginning to end. With traditional CFD simulation could also be done only with transient time, but it is computationally more demanding and therefore not reasonable.

When comparing the simulation times of RANS (18 min) to LBM (4 min), LBM is faster. Also, the use of 128 cores in RANS simulation was unnecessarily high and did not significantly decrease the simulation time. But simulating 10,000 timesteps with LBM is faster. Simulation time from QLBM (28 hour) is significantly higher than it is for RANS or LBM. But it is not fair comparison as the QLBM is designed to run on quantum computer. Emulator is method to prove that it can be run on quantum computer in future, but also gives the ability to improve the code more efficiently. Running this simulation on quantum computer would give better estimate of the QLBM case real simulation time.

As can be observed from the figures, compared to QLBM, RANS models struggle in accurately capturing the flow features. Although these are not seen very clearly in these

low velocity simulations. With higher velocities it is expected that the difference in accuracy is much clearer with QLBM compared to RANS.

From these simulations it seems that QLBM simulations gives as good results as traditional CFD with low velocities. More simulations with higher velocities should be made that it could be said certainly that results are as good.

6 CONCLUSION

In this chapter, conclusion of the work is made. Improvements for future works are introduced.

Classical aerodynamic simulations gave good results that were verified with different methods. One of those was comparing the two simulations to each other. This does not give a very accurate result about the reliability of each simulation. Verifying each simulation separately is more accurate and was done by checking the convergence. This cannot be directly seen from residuals. Aerodynamic properties through the simulation are checked and from those analyze at what state the simulation is converged. Also, analyzing the flow field properties at different locations with pathlines, vectors, and contours was used. The final verifying method was to compare results to other studies. From these it can be said that results are quite accurate. Although forward movement simulation is not as accurate as stationary if it is compared to a quadcopter moving forward in real life. Some of the mistakes that were done in this part of the simulations were that too large timestep size was used, although those solutions seemed to be reasonable. The indication of too large timestep was the courant number at some location of the flow field. Courant number should be observed better through the simulation and changes made based on that.

Aerodynamic results from drone CFD simulations were recognized to be too complex to be compared with QLBM. Therefore, after the quantum simulations were conducted traditional CFD simulations needed to be done for airfoil. This ensured that the results would be comparable. This was not the initial plan but worked very well considering how late the QLBM results were obtained.

From aeroacoustics it can be said that CFD simulation provides good results of noise produced, but it cannot be directly compared to real-life drone noise. With these simulations, only noise produced by the propeller is included. To get a better understanding of how significant the effects of background noise and motor noise are to the total noise produced, some further studies should be done. One of these would be to compare simulation results to experimental results. From other studies it was found that motor and background have a significant effect on higher BPFs. Simulations done in this study are still very useful and can be used, for example, for comparing different propeller types.

Aeroacoustics simulations could not be done with the QLBM. These kinds of simulations might be possible to do in future works, as the plan for the model is that it could also simulate acoustics.

The present work lies predominantly on a try-out of a QLBM proof of concept, to demonstrate its capabilities to accurately model aerodynamic flows for basic 2D configurations. Thus, the test case flow characteristics speed and Reynolds number presented are of a significant reduced scale when compared to common aerodynamic flows encountered in reality. Test case shows that density, pressure and velocity can be modeled accurately but is not yet capable for modeling other characteristics.

As QLBM is developed constantly and it was made to work for this specific case, some errors were seen that were fixed during the simulations. These small and relatively simple simulations, compared to traditional CFD, show that a lot of development still needs to be done to accurately simulate similar cases that are possible with traditional CFD. In further studies, errors noted in this work need to be fixed, and the model needs to be scaled to higher velocities so that it could be used in simulations that are closer to the rotational speed of the propeller used in 3D simulations. QLBM simulations showed promising results. This work shows that it can be used to simulate airflows and is not just a theory.

Overall, this thesis was successful. It gave a solid understanding of the QLBM simulations. Future research should include better and more extensive simulations with the QLBM so it could be compared to 3D CFD simulations. Classical drone simulations were excellent practice for getting familiar with Ansys Fluent and with CFD simulations. It gave good knowledge of how Fluent is used and what needs to be taken into consideration when CFD simulations are done. Also, the use of HPC in simulation was good practice, because it is used in simulations more often. This thesis lays the foundations for future research and work done in the field.

REFERENCES

AgEagle Aerial Systems Inc., 2025. eBee X mapping drone - Drones. AgEagle Aer. Syst. Inc. URL https://ageagle.com/drones/ebee-x/ (accessed 7.24.25).

Anderson, J., 2016. Fundamentals of Aerodynamics. McGraw-Hill Higher Education, New York, UNITED STATES.

Ansys, inc, 2025a. Ansys Fluent Theory Guide. URL https://ansyshelp.ansys.com/public///Views/Secured/corp/v252/en/pdf/Ansys_Fluent_Theory_Guide.pdf (accessed 5.4.25).

Ansys, inc, 2025b. Ansys fluent text command list. URL https://ansyshelp.ansys.com/public//Views/Secured/corp/v252/en/pdf/Ansys_Fluent_Text_Command_List.pdf (accessed 5.7.25).

Blazek, J., 2015. Computational Fluid Dynamics: Principles and Applications. Elsevier Science & Technology, Chantilly, UNITED KINGDOM.

Budinski, L., 2022. Quantum algorithm for the Navier Stokes equations by using the streamfunction vorticity formulation and the lattice Boltzmann method. Int. J. Quantum Inf. 20, 2150039. https://doi.org/10.1142/S0219749921500398

Budinski, L., 2021. Quantum algorithm for the advection–diffusion equation simulated with the lattice Boltzmann method. Quantum Inf. Process. 20, 57. https://doi.org/10.1007/s11128-021-02996-3

Chander, S., 2024. The Current Landscape of Quantum Hardware Development - An Overview. Intersect Stanf. J. Sci. Technol. Soc. 17.

Chen, Q., Zhang, J., Zhang, C., Zhou, H., Jiang, X., Yang, F., Wang, Y., 2023. CFD analysis and RBFNN-based optimization of spraying system for a six-rotor unmanned aerial vehicle (UAV) sprayer. Crop Prot. 174, 106433. https://doi.org/10.1016/j.cropro.2023.106433

CSC, 2024. Performance checklist - Docs CSC. URL https://docs.csc.fi/computing/running/performance-checklist/ (accessed 5.4.25).

Dbouk, T., Drikakis, D., 2022. Computational aeroacoustics of quadcopter drones. Appl. Acoust. 192, 108738. https://doi.org/10.1016/j.apacoust.2022.108738

DJI Official, 2025a. Buy DJI Mavic 4 Pro - Triple-Lens Flagship Camera Drone - DJI Store. URL https://store.dji.com/fi/product/dji-mavic-4-pro (accessed 7.24.25).

DJI Official, 2025b. Support for Phantom 3 Standard. DJI. URL https://www.dji.com/fi/support/product/photo (accessed 6.6.25).

DJI Official, 2015. DJI Unveils the Phantom 3, the Next Evolution of the World's Most Popular Consumer Drone. DJI Off. URL https://www.dji.com/fi/newsroom/news/dji-unveils-the-phantom-3-the-next-evolution-of-the-world-s-most-popular-consumer-drone (accessed 5.4.25).

EN ISO 3744:2010, 2010. Acoustics. Determination of sound power levels and sound energy levels of noise sources using sound pressure. Engineering methods for an essentially free field over a reflecting plane (ISO 3744:2010).

EU2019/945, 2020. Komission delegoitu asetus (EU) 2019/945, annettu 12 päivänä maaliskuuta 2019, miehittämättömistä ilma-alusjärjestelmistä ja kolmansien maiden miehittämättömien ilma-alusjärjestelmien käyttäjistä.

GAOTek, 2025. GAOTek Survey and Inspection VTOL Drone - Auto Return - GAO Tek. URL https://gaotek.com/product/gaotek-survey-and-inspection-vtol-drone-auto-return/ (accessed 7.24.25).

Georgescu, C.A., Schalkers, M.A., Möller, M., 2025. Fully Quantum Lattice Gas Automata Building Blocks for Computational Basis State Encodings. https://doi.org/10.48550/arXiv.2506.12662

Kelecy, F.J., 2021. An Examination of Quadcopter Drone Noise Using Computational Aeroacoustics.

Kovalev, I.V., Voroshilova, A.A., Karaseva, M.V., 2019. Analysis of the current situation and development trend of the international cargo UAVs market. J. Phys. Conf. Ser. 1399, 55095-. https://doi.org/10.1088/1742-6596/1399/5/055095

Krüger, T., Kusumaatmaja, H., Kuzmin, A., Shardt, O., Silva, G., Viggen, E.M., 2016. The Lattice Boltzmann Method: Principles and Practice, 1st ed, Graduate Texts in Physics. Springer Nature, Cham. https://doi.org/10.1007/978-3-319-44649-3

Leonteva, A., Masella, G., Outteryck, M., Orioli, A.P., Whitlock, S., 2025. Comparative Benchmarking of Utility-Scale Quantum Emulators. https://doi.org/10.48550/arXiv.2504.14027

Mak, J.H., Salim, W.S.-I.W., 2022. Analysis of Propeller Blade Through CFD Modelling. J. Des. Sustain. Environ. 4, 25–31.

Marte, J.E., Kurtz, D.W., 1970. A Review of Aerodynamic Noise From Propellers, Rofors, and Liff Fans.

Nielsen, M.A., Chunag, I.L., 2010. Quantum computation and quantum information, 10th anniversary ed. ed. University Press, Cambridge.

Nonami, K., 2025. Overview of the Global Drone Industry, in: 2025 IEEE International Conference on Mechatronics (ICM). Presented at the 2025 IEEE International Conference on Mechatronics (ICM), pp. 1–6. https://doi.org/10.1109/ICM62621.2025.10934845

Paz, C., Suárez, E., Gil, C., Baker, C., 2020. CFD analysis of the aerodynamic effects on the stability of the flight of a quadcopter UAV in the proximity of walls and ground. J. Wind Eng. Ind. Aerodyn. 206, 104378. https://doi.org/10.1016/j.jweia.2020.104378

Paz, C., Suárez, E., Gil, C., Vence, J., 2021. Assessment of the methodology for the CFD simulation of the flight of a quadcopter UAV. J. Wind Eng. Ind. Aerodyn. 218, 104776. https://doi.org/10.1016/j.jweia.2021.104776

Schalkers, M.A., Möller, M., 2024. Efficient and fail-safe quantum algorithm for the transport equation. J. Comput. Phys. 502, 112816. https://doi.org/10.1016/j.jcp.2024.112816

Shinde, T., Budinski, L., Niemimäki, O., Lahtinen, V., Liebelt, H., Li, R., 2025. Utilizing classical programming principles in the Intel Quantum SDK: implementation of quantum lattice Boltzmann method. ACM Trans. Quantum Comput. 6, 9:1-9:18. https://doi.org/10.1145/3678185

Skyline UAV, 2025. Vapor 55 UAV Helicopter | Pulse Vapor 55. Skyline UAV. URL https://www.skylineuav.com.au/fleet/vapor-55-uav-helicopter/ (accessed 7.24.25).

Stephan, J., 2025. Multicopter Flight Control: Fundamentals of Flight Mechanics, Control Allocation, and Control Design, 1st ed, Advances in Industrial Control. Springer, Cham. https://doi.org/10.1007/978-3-031-78487-3

Succi, S., 2018. The Lattice Boltzmann Equation: For Complex States of Flowing Matter, 1st ed, Numerical mathematics and scientific computation. University Press, Oxford. https://doi.org/10.1093/oso/9780199592357.001.0001

Versteeg, H.K., Malalasekera, W., 2007. An introduction to computational fluid dynamics: the finite volume method, 2nd ed. ed. Pearson/Prentice Hall, Harlow.

Vieira, A., Cruz, L., Lau, F., Mortagua, J., Santos, R., 2015. A new computational framework for UAV quadrotor noise prediction, in: Proceedings of the 5th CEAS Air & Space Conference, Delft, The Netherlands. pp. 7–11.

VTT, 2025. VTT and IQM launch first 50-qubit | VTT News. URL https://www.vttresearch.com/en/news-and-ideas/vtt-and-iqm-launch-first-50-qubit-quantum-computer-developed-europe (accessed 7.22.25).

VTT, 2023. Finland launches a 20-qubit quantum computer – | VTT News. URL https://www.vttresearch.com/en/news-and-ideas/finland-launches-20-qubit-quantum-computer-development-towards-more-powerful-quantum (accessed 5.8.25).

Wawrzyniak, D., Winter, J., Schmidt, S., Indiniger, T., Janßen, C.F., Schramm, U., Adams, N.A., 2025. Dynamic Circuits for the Quantum Lattice-Boltzmann Method. https://doi.org/10.48550/arXiv.2502.02131

Xu, L., Li, M., Zhang, L., Sun, H., Yao, J., 2025. Improved quantum lattice Boltzmann method for advection-diffusion equations with a linear collision model. Phys. Rev. E 111, 045305. https://doi.org/10.1103/PhysRevE.111.045305

Zhaoli, G., Chang, S., 2013. Lattice Boltzmann Method and Its Applications in Engineering | Advances in Computational Fluid Dynamics. World Scientific Publishing Co. Pte. Ltd.