

Simulation of rotating elements of a quadcopter using OpenFOAM

- Examination of cost and accuracy of using
reference frames and dynamic meshes

Shravan Vijayaraghavan (shrvi971)

Simulation of rotating elements of a quadcopter using OpenFOAM

Examination of cost and accuracy of using reference
frames and dynamic meshes

Shravan Vijayaraghavan (shrvi971)

Academic supervisor: Marcus Jansson
Industrial supervisor: João Miranda
Examiner: Roland Gårdhagen

Abstract

The use of Computational Fluid Dynamics(CFD) is wide-spread as applications range from bio-medical engineering to the aerospace and defense industries. It was initially used mainly for the aerospace industry but finds its uses today in many other fields. Commercial softwares that model fluid flows often are very expensive to purchase licenses for and they require professionals with adequate knowledge of fluid dynamics to operate and use. To eliminate the large costs associated with licenses, OpenFOAM was established as an open-source software that could be used to model fluid flows. However, the drawback associated with OpenFOAM was its lack of a resource of information on the workings of the software and dependence on a community to solve problems. To get rid of this problem, AirShaper was established with the sole intention of providing CFD simulations to clients with rudimentary understanding of CFD at low costs thereby, removing the need for hiring experts.

At present, AirShaper has limitations on implementing stationary models on their platform. In order to expand client base, there was a need for research on the use of rotation models. As the company implements OpenFOAM's code and solvers onto its platform, three different rotation methodologies available in OpenFOAM were investigated for computational cost and accuracy. The methods investigated were; Single Reference Frame(SRF), Multiple Reference Frame(MRF) and Arbitrary Mesh Interface(AMI) methods.

Using a free-standing propeller and propellers of a quadcopter, the capabilities of the methods were investigated. Experimental results and numerical comparisons available from NASA's research on small quadcopters were compared with the results obtained from this thesis. The MRF method was judged the best method in terms of computational cost to implement for the accuracy provided. The AMI method proved to have the highest accuracy albeit it took almost twice the computational time as the MRF method. Recommendations to include the MRF method as the default rotation method on AirShaper's platform was provided and to provide the AMI method to clients seeking higher accuracy.

Acknowledgements

I'd like to take this opportunity to thank and acknowledge all the people involved in bringing this thesis to completion. I would like to start off by expressing my sincerest gratitude to the founder of AirShaper, Wouter Remmerie for giving me the opportunity to conduct my thesis at AirShaper. I would also like to express my sincerest gratitude to my industry supervisor at AirShaper, João Miranda, who always took the time out to help me with any problems I encountered and guided me in the right direction.

Furthermore, I would also like to thank my academic supervisor and examiner, Marcus Jansson and Roland Gårdhagen, for their support and guidance not only during the thesis, but also throughout my two years at university. Finally I would like to thank my family and friends for their encouragement and support during my thesis.

Nomenclature

Abbreviations and Acronyms

Abbreviation	Meaning
LiU	Linköping University
CFD	Computational fluid dynamics
UAV	Unmanned Aerial Vehicle
CAD	Computer aided design
CPU	Central processing unit
FOAM	Field Operation and Manipulation
SRF	Single reference frame
MRF	Multiple reference frame
AMI	Arbitrary Mesh Interface
SST	Shear Stress Transport
SIMPLE	Semi-Implicit Method for Pressure Linked Equations
PISO	Pressure Implicit with Splitting of Operator
RANS	Reynolds Averaged Navier Stokes Equation
LES	Large Eddy Simulations
DES	Detached Eddy Simulations
RPM	Revolutions per minute
PCG	Preconditioned conjugate gradient
DIC	Diagonal incomplete-Cholesky
GAMG	Geometric agglomeration algebraic multigrid
VTK	Visualization ToolKit

Latin Symbols

Symbol	Description	Units
A	area	$[m^2]$
u	Velocity	$[ms^{-1}]$
p	Pressure	$[Pa]$
t	Time	$[s]$
ρ	Density	$[kgm^{-3}]$
τ	Stress tensor	
T	Temperature	$[K]$

Greek Symbols

Symbol	Description	Units
ω	Omega	$[rads^{-1}]$
ϵ	Epsilon	$[Jkg^{-1}s^{-1}]$

Contents

1	Introduction	1
1.1	Objectives	2
1.2	Limitations	2
1.3	Outline	3
2	Theory	5
2.1	Governing Equations	5
2.1.1	Incompressible Navier-Stokes Equations	5
2.2	Turbulence Modelling	5
2.2.1	k- ω SST model	6
2.2.2	k- ϵ model	6
2.3	OpenFOAM Solvers	7
2.3.1	simpleFoam	7
2.3.2	pimpleFoam	7
2.4	Rotating Motions	7
2.4.1	Single Reference Frame (SRF)	8
2.4.2	Multiple Reference Frame (MRF)	8
2.4.3	Arbitrary Mesh Interface (AMI)	8
3	OpenFOAM	11
3.1	blockMesh	12
3.2	snappyHexMesh	13
3.3	polyMesh	14
3.4	topoSet	14
3.5	fvSchemes	15
3.6	fvSolution	16
3.7	controlDict	16
3.8	ParaView	16
4	Method	18
4.1	Free-standing Propeller	18
4.1.1	Geometry	18
4.1.2	Domain	18
4.1.3	Mesh	19
4.1.4	Convergence	22
4.1.5	Solver Setup	23
4.1.6	Single Reference Frame(SRF)-FSP	24
4.1.7	Multiple Reference Frame(MRF)-FSP	24
4.1.8	Arbitrary Mesh Interface(AMI)-FSP	26
4.2	Dji Phantom 3 Quadcopter	28
4.2.1	Geometry	28
4.2.2	Domain	28
4.2.3	Mesh	29
4.2.4	Solver Setup	30

4.3	MRF Dictionary- Quadcopter	31
4.4	AMI Dictionary- Quadcopter	33
5	Results	35
5.1	Free-standing Propeller	35
5.1.1	Velocity Profiles	35
5.1.2	Experimental Comparison	38
5.1.3	Computational Time	39
5.2	DJI Phantom 3	40
5.2.1	Single Rotor	40
5.2.2	Simplified Dji Phantom 3	41
5.2.3	Comparison of different configurations	42
5.2.4	Effects of variation in MRF Cylinder radius	44
5.2.5	Effects of weather	46
5.2.6	Effects of pitch and weather on the Simplified Dji Phantom 3	49
5.2.7	Overmount and Undermount configuration comparisons	50
6	Discussion	53
6.1	FSP	53
6.1.1	Velocity magnitudes	53
6.1.2	Experimental Comparisons	53
6.1.3	Computational Time	53
6.1.4	Outtakes	54
6.2	Dji Phantom 3	54
6.2.1	Single rotor	54
6.2.2	Simplified Dji Phantom 3	54
6.2.3	Comparison of different configurations	55
6.2.4	Effects of variation in MRF Cylinder radius	55
6.2.5	Effects of weather	56
6.2.6	Effect of weather and pitch angle	56
6.2.7	Overmount and undermount comparisons	56
6.2.8	Outtakes	57
7	Future Work	59
8	Conclusions	61
	Appendices	65
A	OpenFOAM Solvers and Script	65
A.1	OpenFOAM ./Allrun script	65
A.1.1	fvSchemes Dictionary	65
A.1.2	fvSolution Dictionary	66
B	Meshing in OpenFOAM	69
B.1	blockMesh	69
B.1.1	snappyHexMesh	70

1 Introduction

Unmanned Aerial Vehicles (UAVs) are known for having a wide variety of applications ranging from landscape surveying, industrial inspection and monitoring, to precision agriculture and aerial imaging. The extensive use of UAVs can be linked to the cost-effective solutions provided by them. Important advantages of multi-rotored vehicles compared to single rotor vehicles are the increased lifting capacity, smoother vertical flight and safety redundancy. Moreover, lower manufacturing and production values combined with the lack of a human pilot decrease acquisition and operating costs significantly. Thus, there is a large shift across the aviation industry to use UAVs instead of conventional manned aircrafts.

On a general basis, UAVs are usually dimensioned and optimized for particular missions. This is where CFD can come in handy as it can predict the flow characteristics and help to further optimize the UAV for the mission it is designed for. Prediction of flow behaviour around a UAV is extremely challenging as the components are very tightly packed and there is little room for adjustments. Also, given the close proximity the tip vortex and wake from one propeller can mix up with the tip vortices and wakes of the other propellers and the modeling of such flows becomes very complicated. CFD studies can be categorized into two areas; low and high fidelity methods. The high fidelity methods which can use unsteady flow models can help capture and predict these complicated flow behaviours whereas the low fidelity methods are not suitable for accurately predicting the unsteadiness of the flow. However, the low fidelity models are computationally cheap and offer a good co-relation which is why they are often used to perform iterative studies on UAVs.

AirShaper is an online platform that provides aerodynamic solutions for designers, engineers & architects where models can be uploaded and analyzed. The platform works by running a CFD simulation on an uploaded 3D model through their cloud based virtual wind tunnel. Founded in 2017, the company works with clients to improve the aerodynamic efficiency of their products.

The online platform's current functionality is limited to static geometries that are either on or above the ground which can be either static or moving. A lot of the external aerodynamic studies involve implementing rotating elements to get accurate results, which could be the case for drones, wind turbines and even vehicles. The implementation of rotating elements is a drawback at the moment for the platform as it works only with static models.

To expand the platform's reach into new markets and also get accurate results from running aerodynamic tests involving moving geometries, the company is investigating the use of rotating elements through this thesis. In CFD, there are different methods and approaches available to model rotating elements like the multiple reference frame (MRF) method, arbitrary mesh interface (AMI) method among others. The use and implementation of rotating elements can get quite complicated and

computationally expensive which is why prior research has to be conducted on the feasible methods that can be undertaken and also a comparison of the methods to investigate the superiority of one over the other.

The scope of this thesis was therefore, to examine three methods, namely; the single reference frame (SRF) method, the multiple reference frame (MRF) method and the arbitrary mesh interface (AMI) method, regarding their impact on accuracy, computational cost and feasibility towards commercial applications.

The respective methodologies are first established for a free-standing propeller and are tested on the propellers of the Dji Phantom 3, a common quadcopter used for commercial applications. Different tests are performed to understand the limitations and accuracy of each method and are presented in subsection 1.1.

1.1 Objectives

OpenFOAM, an open-source CFD tool was used for conducting the study as the online wind-tunnel platform works on code developed from OpenFOAM's solvers. The primary objectives of the following study are summarized below:

- Set up a reliable and robust methodology to perform CFD analysis on a free standing propeller using above mentioned methods.
- With the 3 methods established, implement the said methodologies for the propellers of the Dji Phantom 3
- Compare and contrast different methodologies: single reference frame, multiple reference frame and moving mesh methodologies in terms of accuracy and computational cost
- Rank the best performing method based on comparisons of numerical and experimental data
- Use the best performing method to check for capability by conducting tests such as effects of weather, and effects of pitch angle on the Dji Phantom 3 model

1.2 Limitations

The limitations of the thesis were:

- The CAD models of the free-standing propeller and the quad-copter were not made available by the company that manufactured them and hence, the CAD models may have slight modifications/alterations from the actual manufactured model.
- Limited numerical data conducted for the same methodologies as used in this study was found
- Computational resources in terms of CPU power were limited to 32 cores with approximately 200GB of RAM

1.3 Outline

The thesis was carried out to compare different methodologies to implement rotational components onto AirShaper's online wind-tunnel platform. The following section, section 2 describes the theory of the different methodologies used as well as the solvers used for the study. Following the theory, section 3 deals with the software used and gives a brief insight into how the software implements a CFD code. The methodology followed for the thesis is shown in section 4 which lists out the domain, mesh and solver settings used for the study. Further, results from the different methods are shown in section 5 and are subsequently discussed in section 6. Finally, conclusions, recommendations and future works are discussed in section 8.

2 Theory

2.1 Governing Equations

For a fluid, there are three equations that govern the properties of the fluid, the continuity, momentum and energy equation. These equations are called the governing equations for fluids and when the fluid is assumed to be Newtonian, they are written as:

Continuity equation:

$$\frac{\partial p}{\partial t} + \rho \frac{\partial u_k}{\partial x_k} = 0 \quad (1)$$

Momentum equation:

$$\rho \frac{\partial u_i}{\partial t} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho F_i \quad (2)$$

Energy equation:

$$\rho \frac{\partial e}{\partial t} = -\rho \frac{\partial u_i}{\partial x_i} + \Phi + \frac{\partial}{\partial x_i} \left(k \frac{\partial T}{\partial x_i} \right) \quad (3)$$

The above are the equations describing respectively the conservation of mass, the conservation of momentum and the conservation of energy[1].

2.1.1 Incompressible Navier-Stokes Equations

For an incompressible flow, the compressibility effects are negligible and $\rho=\text{constant}$, which means that the conservation of mass equation becomes,

$$\frac{\partial u_i}{\partial x_i} = 0 \quad (4)$$

The fluid here experiences no change in volume and the conservation of momentum reduces to the form:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho} \frac{\partial p}{\partial x_i} + \nu \nabla^2 u_i + F_i \quad (5)$$

The energy equation is not included as there is no coupling between the momentum and conservation of mass equations because of which the pressure field is not unique[1].

2.2 Turbulence Modelling

The partial differential equations described above govern the behaviour of fluids, however, there is usually no known closed analytic solution for them, except in rare very simplified problems. Furthermore, the direct numerical simulation (DNS) of the equations is computationally very expensive, therefore these have to be simplified. Possible simplifications vary from case to case but the most common involve

the modeling or partial modeling of the turbulent characteristics of the flow. Turbulence models can be divided into two major groups Large Eddy Simulations (LES) where only the smaller scales are modeled and Reynolds Averaged Navier Stokes Equation (RANS). The later uses the Reynold's decomposition to separate the flow variables into a fluctuating and a mean component. A property of the composition makes the fluctuating component zero, therefore, time averaging the components provides the flow properties. The need for closure of the RANS equations led to the creation of many turbulence models with the two-equation family of models being the most popular. They all rely on Boussinesq's approximation where the stress tensor is modeled in the viscous term of Navier-Stokes equation. The two equation models used are the k- ω SST model by Menter and the k- ϵ model.

2.2.1 k- ω SST model

The k- ω SST model was developed by Menter[2] in 1993 and it is a two-equation eddy-viscosity model with a shear-stress transport formulation built into it. The wide usage of the k- ω SST model is due to the fact that it inherits the best properties of both the models, the k- ω and the k- ϵ formulations, and switches between them when required. The k- ω formulation is very good for low reynolds models as it can be used in the viscous sub-layers of the boundary layers near the wall while the k- ϵ model is good for capturing the free-stream behaviour. As a RANS model, it is known to predict the adverse pressure gradient region and separation region quite effectively[2][3]. The governing equations of the model can be summarized as shown below:

Turbulence Kinetic Energy:

$$\frac{\partial k}{\partial t} + u_j \frac{\partial k}{\partial x_j} = P_k - \beta * k\omega + \frac{\partial}{\partial x_j} \left[(\nu + \sigma_k \nu_T) \frac{\partial k}{\partial x_j} \right] \quad (6)$$

Specific Dissipation Rate:

$$\frac{\partial \omega}{\partial t} + u_j \frac{\partial \omega}{\partial x_j} = \alpha S^2 - \beta \omega^2 + \frac{\partial}{\partial x_j} \left[(\nu + \sigma_\omega \nu_T) \frac{\partial \omega}{\partial x_j} \right] + 2(1 - F1)\sigma_\omega^2 \frac{1}{\omega} \frac{\partial k}{\partial x_i} \frac{\partial \omega}{\partial x_i} \quad (7)$$

Additional information about the model can be obtained from [2] [3],[4] and [5].

2.2.2 k- ϵ model

The standard k- ϵ turbulence model uses two partial differential equations. It solves the kinematic turbulent viscosity by solving transient equations for the turbulent kinetic energy (k) and its dissipation (ϵ), hence the name. The model in itself is semi-empirical since there are model coefficients that are assumed, but there are also closure relations. The equations are given in the form:

Turbulence Kinetic Energy:

$$u_i \frac{\partial k}{\partial x_i} = \frac{\partial}{\partial x_j} \left(\left(\mu + \frac{\mu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right) + 2\mu_t E_{ij} E_{ij} - \rho \epsilon \quad (8)$$

Specific Dissipation Rate:

$$u_i \frac{p\epsilon}{x_i} = \frac{1}{x_j} \left(\left(\mu + \frac{\mu_t}{\sigma\epsilon} \right) \frac{\epsilon}{x_j} + C_{\epsilon 1} \frac{\epsilon}{k} 2\mu_t E_{ij} E_{ij} - C_{\epsilon 2} \rho \frac{\epsilon^2}{k} \right) \quad (9)$$

where U_i is the velocity component for the relevant direction and E_{ij} is the component of rate of deformation.

The k- ϵ turbulence model is known for its easy implementation and stability. However, they are known to show stability issues when it comes to rotating flows[6].

2.3 OpenFOAM Solvers

OpenFOAM solvers implement and solve the continuity, momentum and energy equations separately. The solvers used for the study are discussed below:

2.3.1 simpleFoam

simpleFoam is a steady-state incompressible, turbulent flow solver which uses the SIMPLE algorithm. An algorithm expressing the implementation of the SIMPLE method is shown[7]:

1. Setup boundary conditions and compute gradients of pressure and velocity
2. Solve the momentum equation to compute intermediate velocity field
3. Solve pressure correction equation
4. Update pressure field and boundary conditions
5. Repeat until convergence is obtained

The time steps used for *simpleFoam* are integrated time steps and not actual time steps since it is a steady-state solver. This can be seen in the *controlDict* file, where *deltaT* is assigned as 1.

2.3.2 pimpleFoam

pimpleFoam is a transient solver for incompressible flow with a large time step implementing the combination of the PISO-SIMPLE algorithm. A large time step is possible for the solver as the PIMPLE algorithm uses larger courant numbers. The SIMPLE algorithm as discussed in the *simpleFoam* section uses a pressure correction term instead of a velocity correction term as they are unknown. The PISO algorithm also neglects the velocity correction in the first step but uses it at a later stage which leads to a faster rate of convergence[8].

2.4 Rotating Motions

Three approaches are chosen to model rotating geometries, the SRF (Single Reference Frame), the MRF (Multiple Reference Frames) and the AMI (Arbitrary Mesh

Interface). The SRF method is considered the simplest as a velocity is prescribed at the inlet to induce a rotating motion. The MRF method can have two or more reference frames and it works by inducing a rotating motion to a smaller region inside the domain where the geometry is placed. The AMI method, as the name suggests, moves the mesh around a prescribed region to induce the rotating motion. The theory for the three methods is described in the following sub-sections:

2.4.1 Single Reference Frame (SRF)

The SRF model computes fluid flow in a rotating frame of motion. The model is compatible with both steady and transient solutions but the study aims to discuss and compare only the steady-state model. The governing equation for the SRF simpleFoam model is given by:

$$\nabla \cdot (u_R u_R) + 2\Omega \times (\Omega \times r) = -\nabla p + \nabla \cdot (v_{eff}(\nabla u_R + (\nabla u_R)^T)) \quad (10)$$

Equation 10 includes the Coriolis force and the centrifugal force. In the above equation, u_R is the velocity when viewed from the rotating frame[9]. The velocity viewed from the stationary frame is given by:

$$u_L = u_R + \Omega \times r \quad (11)$$

2.4.2 Multiple Reference Frame (MRF)

The MRF model computes fluid flow using the stationary and the rotating reference frames. The MRF model in OpenFOAM is only compatible with steady state solvers. The governing equations for the MRF model can be split into a rotating and a stationary part:

Stationary part:

$$\begin{aligned} \nabla \cdot (u_1 u_1) &= -\nabla p + \nabla \cdot (v_{eff}(\nabla u_1 + (\nabla u_1)^T)) \\ \nabla \cdot u_1 &= 0 \end{aligned} \quad (12)$$

Rotating part:

$$\begin{aligned} \nabla \cdot (u_R u_1) + \Omega \times u_1 &= -\nabla p + \nabla \cdot (v_{eff}(\nabla u_1 + (\nabla u_1)^T)) \\ \nabla \cdot u_R &= 0 \end{aligned} \quad (13)$$

The rotating part has the Coriolis force added to the equation. The required steps/procedures to obtain these equations from the governing equations can be found through [9].

2.4.3 Arbitrary Mesh Interface (AMI)

The AMI mesh introduced to OpenFOAM works by displacing the mesh points without altering the topological information of the mesh where the topology of the mesh defines how the points, edges, faces and cells are built. The mesh motion type used

in this thesis is the Solid Body Motion. Such a mesh motion occurs when no relative displacement between two points takes place [10]. The *solidBodyMotionFunction* of OpenFOAM is defined by a variety of classes and the motion is a combination of a vector for translation and one for rotation.

3 OpenFOAM

OpenFOAM is an C++ based open-source platform mostly knowb for its CFD capabilities. OpenFOAM stands for Open Source Field Operation And Manipulation which allows for development of customized numerical solvers apart from pre and post processing of results from these solvers. One can set up the solvers in a number of different ways but there are three basic folder directories that need to be set-up before running a case. A flowchart representing the directories and the sub-files is shown in Figure 1.

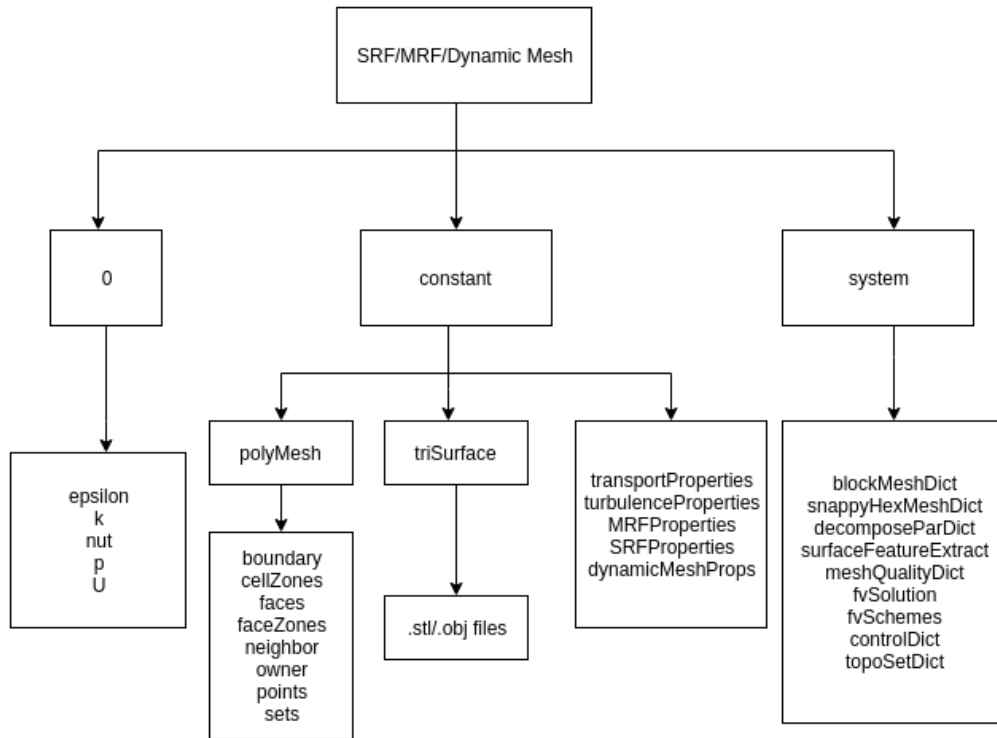


Figure 1: Flowchart indicating the common directories found under the cases.

A generalized description of the files under each directory is enumerated below:

- **0** - This is the initialization time directory which contains the initial and boundary conditions for the flow variables. Since, the study deals with turbulence models, Figure 1 shows the different variable files that need to be present. Further, each variable dictionary file must contain three entries, the *dimensions* entry which specifies the dimensions of the variable, the *internalField* entry which specifies the internal field and finally the *boundaryField* which specifies the boundary conditions for the different patches created with the *blockMesh* file.
- **constant** - The constant directory contains the turbulence properties and

transport properties amongst other files. The *turbulenceProperties* file contains the type of turbulence model used such as RANS, LES, DES or laminar while the *transportProperties* file lists out whether the fluid used is Newtonian or non-Newtonian and also the description of the kinematic viscosity. When the user wishes to use their own CAD geometry, then they must place the .stl or .obj file in the constant directory under a sub-directory called *triSurface*. The constant folder directory will also contain the *polyMesh* sub-folder which as shown in Figure 1 contains the mesh information.

- **system** - The system folder contains the files that control the running of the solver and can contain post processing functions such as forces calculation and field averaging. The main files that can be found in the system directory are the *fvSchemes*, *fvSolution* and the *controlDict* files. Apart from these three files, the *blockMesh* and the *snappyHexMesh* dictionary files are also found in the system directory. These files would be considered the primary files while there are secondary files, for example, the *decomposePar* dictionary file that is used when solving a case in parallel which can also be found in the system directory.

A detailed description of how to create a mesh is shown in the following sections which begins with creating a background mesh using *blockMeshDict*, and refining the mesh using *snappyHexMeshDict*. Further, the way to setup schemes and solver settings is listed out.

3.1 blockMesh

blockMesh is OpenFOAM's mesh generation utility which comes with features to generate a parametric mesh that can be graded. The mesh is generated by a dictionary file called *blockMeshDict* where the background mesh is specified. A standard *blockMeshDict* file has the following features which can be manipulated for the required mesh:

- *convertToMeters* - Scaling factor for the co-ordinates.
- *vertices* - Vertex co-ordinates for the background mesh are to be listed here in the (x y z) co-ordinate system.
- *edges* - If the mesh requires an arc or a spline then the edges features should be used.
- *blocks* - The list of vertex labels and the mesh size is provided in the blocks feature.
- *patches* - The list of patches are mentioned in this section

An example illustrates the working of the *blockMesh* feature:

```
convertToMeters 1;
vertices
( 0 0 0 )
```

```

( 0.5 0 0 )
( 0.5 0 0.5 )
( 0 0 0.5 )
hex (0 1 2 3) simpleGrading (1 1 1)
boundary
    inlet
    {
        type patch
        faces      (0 1 2 3)
    }

```

The above example creates a block with the vertices as specified by the co-ordinates. The size of the mesh is specified by the *simpleGrading* feature which corresponds to the size in the x,y and z directions. Additionally, the block is assigned as the inlet with the help of patches. Using the *blockMesh* feature, the domain of the model used in this study was generated and is shown in subsection 4.1.2.

3.2 snappyHexMesh

snappyHexMesh is a tool provided by OpenFOAM for generating hexahedral elements from triangulated surface geometries. This requires a pre-existing/ background mesh generated by the *blockMesh*. File extensions provided by *.stl* and *.obj* are compatible with *snappyHexMesh*. This feature works by cutting the background mesh with the geometry provided and then refining the mesh around the geometry. *snappyHexMesh* comes with three main functions, the *autoRefineDriver* which is responsible for refining and cutting the mesh, the *autoSnapDriver* which is responsible for snapping the mesh to the surface and finally, the *autoLayerDriver* which adds inflation/boundary layers to the mesh [11]. The above three functions can be activated individually or together:

```

castellatedMesh true;
snap true;
addLayers true;

```

After this stage, one must define the geometries that need to be cut and snapped by *snappyHexMesh* in the following way:

```

geometry
{
    Propeller
    {
        type triSurfaceMesh;
        file "Propeller.stl";
    }
    refinementBox
    {
        type searchableBox;
        min (-0.3 -0.075 -0.075);
        max (-0.2 0.075 0.075);
    }
}

```

```
    }  
};
```

The geometry used here is that of a Propeller with the file extension *.stl*. The propeller is just an example and the user can replace it with any file needed. The *refinementBox* is an area where the mesh can be refined around the geometry to capture the wake and physics of the flow. For the study of the free-standing propeller as will be discussed in the subsubsection 4.1.2, two refinement boxes will be used.

Each of the three functions have additional parameters that can be modified and adjusted to produce a better mesh. To get a better understanding of how each parameter works, the user must read the user guide of OpenFOAM[12]. However, the main parameters that one can adjust to produce a good mesh are shown below:

- Explicit feature edge refinement - Specify a level for the cells intersected by the edges
- Surface based refinement - Here, every surface defined in the geometry section is specified by two levels, a minimum and maximum level with which the surface gets refined
- Region wise refinement - Specify refinement levels for the refinement boxes created

An important criteria while adjusting parameters in the *snappyHexMeshDict* file is the *locationInMesh* command. The points described in this command are the points in the mesh that should be kept. For the case of the free-standing propeller, the mesh inside the domain should stay while the propeller should be removed. Thus, the *locationInMesh* should contain the points that are located inside the domain and away from the propeller region.

A point to be noted while working with the *snappyHexMeshDict* file is that if the user is not absolutely sure about the quality of the mesh in the beginning, then they must ensure that the *addLayersControl* function is turned off (indicated by false in the dictionary file) in the beginning as it can lead to additional problems.

3.3 polyMesh

The *polyMesh* folder contains the files that describe the mesh. The files that are created in the *polyMesh* folder are *points*, the file that describes the points of the mesh, *faces*, the file that describes the faces of the cells, and *owner* and *neighbour* files that describe which faces belong to a cell and information about the connectivity of the cells. A boundary file is also created that lists out the boundaries of the mesh and the associated type they belong to.

3.4 topoSet

The *topoSet* command helps create *cellSets*/*faceSets* through a dictionary called the *topoSetDict*. The *topoSet* results are stored in the *polyMesh* folder and the dictionary

file will normally contain the following commands:

```
actions
{
    name selectedCells;
    type cellSet;
    source boxToCell;
    sourceInfo
    {
        box (x0 y0 z0) (x1 y1 z1);
    }
}
```

Running the command *topoSet* will create a box of the dimensions mentioned by the co-ordinates(x0 to z1) and it will assign it to the type *cellSet* which is saved in the *polyMesh* folder. In subsection 4.1, a detailed description of the *topoSet* dictionary file is explored.

3.5 fvSchemes

The *fvSchemes* lists the discretization schemes for the components in the equations. The schemes that need to be chosen are the time discretization scheme, gradient schemes, divergence discretization scheme, laplacian schemes and interpolation schemes. The available time discretization schemes are summarized in Table 4.

Table 4: Time discretization schemes available from OpenFOAM.

Euler	First order, bounded, implicit
backward	Second order, implicit
steadyState	No time derivatives are solved
localEuler	Local-time step, first order, bounded, implicit
CrankNicholson	Second order, bounded, implicit

The gradient schemes available, through *gradSchemes*, is summarized in Table 5.

Table 5: Gradient schemes available from OpenFOAM.

Gauss linear	Second order, Gaussian integration
cellLimited Gauss linear	Limits gradient to extrapolate to faces
leastSquares	Second order, least squares
fourth	Fourth order, least squares

The default interpolation scheme is the linear interpolation scheme. The other available interpolation schemes are the *cubic* and *midPoint* interpolation schemes. The discretization of divergence schemes is presented in Table 6

Table 6: Divergence schemes available from OpenFOAM.

upwind	First order bounded
linearUpwind	First/Second order, bounded
QUICK	First/Second order, bounded
TVD	First/Second order, bounded

Additional information about the schemes and their use is found in the OpenFOAM User Guide [12].

3.6 fvSolution

In OpenFOAM the available linear solvers are preconditioned conjugate or bi-conjugate gradient solvers, smooth solvers, generalized geometric-algebraic multi-grid (GAMG) solvers and also diagonal solvers (for explicit systems). The *fvSolution* file allows for under-relaxation parameters to be set and also the algorithms (SIMPLE or PISO) that need to be used for the solvers are set here.

3.7 controlDict

The *controlDict* file controls the settings in terms of number of iterations that the solver should run for and also the precision of the solver. The file begins with listing out the application (simpleFoam, pimpleFoam) that the *controlDict* file will be used for and then moves onto listing the start and end times and also the time interval for which data should be stored.

3.8 ParaView

ParaView is an open-source application used for data analysis and visualization that can be done in 3D or via programming. ParaView uses Visualization ToolKit (VTK) as the rendering and data processing engine [13]. ParaView was used as the data visualization and extraction tool throughout the thesis.

4 Method

4.1 Free-standing Propeller

4.1.1 Geometry

The propeller used for the study involving the free-standing propeller was an APC manufactured propeller. APC Propellers are used widely in the model aviation sector as they are made from composite materials and are known to have high performance curves. The company also manufactures over 500 types of propellers ranging from small-quadcopter blades to blades with a 16 inch (40.64 cm) diameter. The manufacturer lists out technical information for each propeller sold in terms of power, torque, thrust and pressure-coefficient values for different speeds and propeller RPMs[14]. This enables an easy comparison of experimental and numerical results. Based on the availability of proper CAD geometries, an APC 15×4 propeller was used for the study and the propeller can be seen from Figure 2.

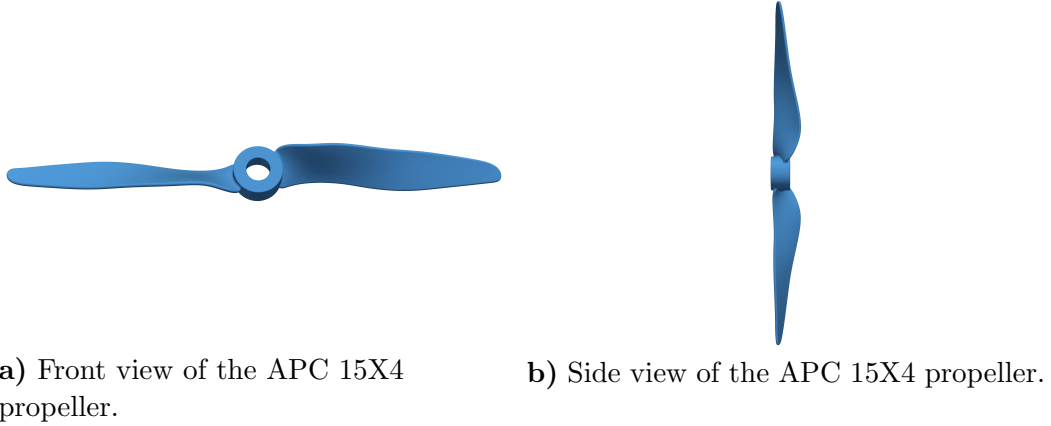


Figure 2: CAD model of the APC 15X4 propeller.

4.1.2 Domain

A cylindrical domain was modeled with the *blockMeshDict* file which consisted of two refinement cylinders around the propeller. The purpose of two refinement cylinders was to make sure that the cell size would decrease gradually around the propeller and also, ensured that the wake generated by the propeller would be adequately captured. The domain did not have to be too wide as the boundary conditions would mimic the case of a propeller exposed to the outside environment.

The domain used for the study is shown in Figure 3. A quantitative description of the domain is shown in Table 7 which indicates the size of the global domain.

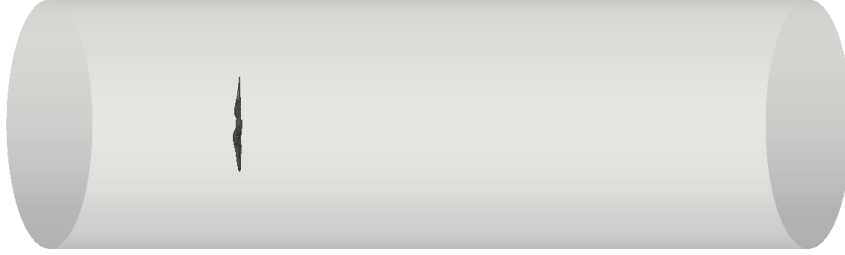


Figure 3: A visual representation of the side-view of the domain with the propeller(shaded in black).

Table 7: Global domain size for the propeller

Respective Lengths	Units [mm]
Length of domain (L)	1000
Upstream Length (L_f)	150
Downstream Length (L_b)	750
Cylinder diameter (L_d)	10

4.1.3 Mesh

Studies involving external aerodynamics require a refined mesh around the object of study as this is crucial to understand the flow physics especially in the wake region. OpenFOAM's mesh utilities allow for a simple mesh with hexahedral elements without compromising the quality of the mesh. The utilities, *blockMesh* and *snappyHexMesh* were used to generate a hexahedral mesh with 2 refinement boxes around the propeller region.

The side view of the domain with the structured mesh is shown in Figure 4. An O-grid mesh was implemented that allowed for finer cells around the propeller section and a close-up of the O-grid mesh can be viewed from Figure 5.

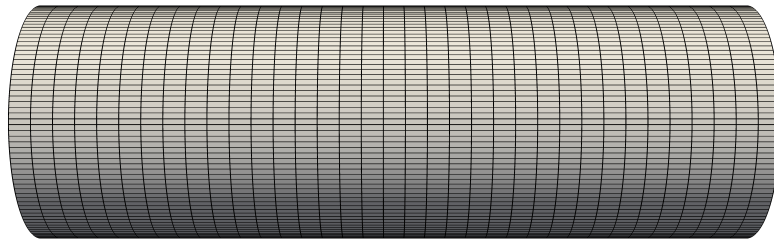


Figure 4: Side view of the domain(FSP) showing the structured cell layout.

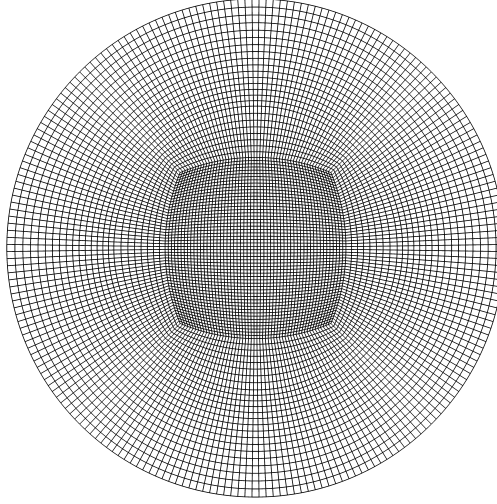


Figure 5: The O-grid mesh approach, allowing for more cells to be packed around the propeller area.

A mesh independence study was carried out for the SRF and MRF cases with meshes of different sizes. The propeller was rotated at an RPM of 1000 with an inlet velocity of $10[m/s]$ for both the cases. Parameters such as lift, drag and velocity magnitudes along a line were compared for the cases and are presented subsequently.

SRF-Mesh Independence

The independence study for the SRF method involved using the $k-\omega$ SST turbulence model for meshes of 3 different sizes as seen from Table 8.

Table 8: SRF method mesh independence - $k-\omega$ SST model.

Mesh Size	Lift [N]	% Difference	Drag [N]	% Difference
300,000	0.000138	-	0.0963	-
600,000	0.000219	37%	0.0958	0.51%
1,290,000	0.000222	1.35%	0.0927	3.23%

From Table 8, it is seen that the lift and drag vary less than 5% for mesh with a size of 1.29 million elements. Further, the velocity magnitude along the center line is shown in Figure 6, where the flow downstream of the propeller is seen not to settle for the mesh with 300k elements. Thus, the mesh with 1.29 million elements was used as the SRF method's mesh for comparison against the other methods.

MRF- Mesh Independence

The mesh independence study for the MRF method was carried out for two different turbulence models, the $k-\epsilon$ and $k-\omega$ SST. The $k-\epsilon$ model is shown followed by the $k-\omega$ SST method.

Table 9: MRF method mesh independence - k- ϵ model.

Mesh Size	Lift [N]	% Diff	Drag [N]	% Diff.
300,000	-6.917e-5	-	0.1763	-
600,000	-7.105e-4	9.8%	0.1891	6.7%
1,290,000	0.001834	38.9%	0.4629	59%
4,000,000	0.00171	6.7%	0.4432	4.2%

The lift and drag for the meshes of the higher order were seen to deviate with a maximum of 6.7% for the lift and 4.2% for the drag. Furthermore, the velocity magnitude, as seen from Figure 7, along the center line was plotted and the meshes with the size of 1.3 million and 4 million can be seen nearly overlapping each other. Both of these results together allow us to conclude that mesh independence is reached for a mesh of 1.29 million elements.

The same procedures was followed for the k- ω SST model and the results are presented below:

Table 10: MRF method mesh independence - k- ω SST model.

Mesh Size	Lift [N]	% Difference	Drag [N]	% Difference
300,000	1.787e-5	-	0.09898	-
600,000	7.22e-5	75.2%	0.1098	9.8%
1,290,000	0.00117	93.2%	0.0984	11.5%
4,000,000	0.00132	10.3%	0.0962	2.2%

From Table 10, it is evident that there is a significant difference between the lift for the 600k and the 1.29 million mesh count. This difference becomes small when compared to the 4 million mesh count which is also evident from Figure 8.

The mesh independence study shows that mesh with 1.29 million elements is sufficient for both the k- ω SST and the k- ϵ model, thus it is used henceforth, as the mesh for comparison against the other two rotation methods.

AMI - Mesh Independence

The independence study for the AMI method involved using the k- ω SST turbulence model for meshes of 3 different sizes as seen from Table 11.

Table 11: AMI method mesh independence - k- ω SST model.

Mesh Size	Lift [N]	% Difference	Drag [N]	% Difference
100,000	0.00177	-	0.0849	-
400,000	0.00385	48.9%	0.0912	6.9%
1,760,000	0.00427	9.83%	0.0977	6.6%

Table 11 shows a maximum deviation of 9.8% and 6.6% for the lift and drag for the mesh count of 1.7 million. Further, Figure 9 shows a good agreement in the velocity profiles obtained with meshes with 400k and 1.7 million cells, with a maximum

deviation of 3%. Thus, the mesh with 1.7 million cells was used as the reference for the AMI.

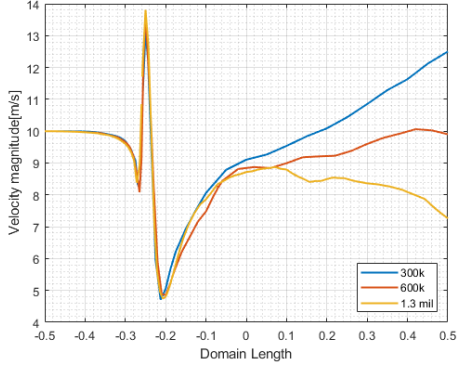


Figure 6: Velocity magnitude comparison for the SRF method.

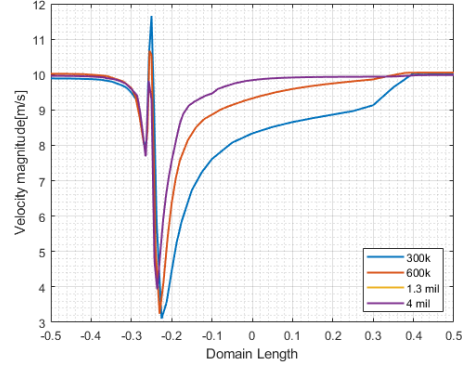


Figure 7: Velocity magnitude comparison for the MRF method with $k-\epsilon$ model.

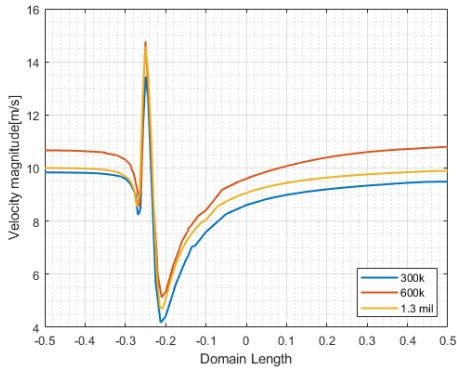


Figure 8: Velocity magnitude comparison for $k-\omega$ SST model for MRF method.

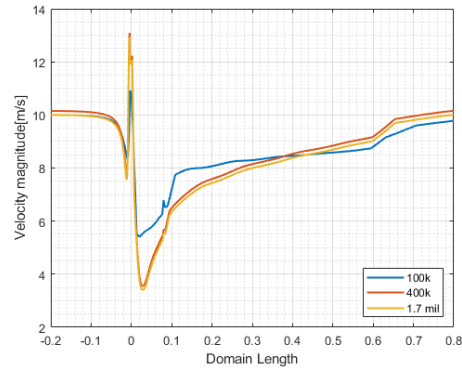


Figure 9: Velocity magnitude comparison for the AMI method.

4.1.4 Convergence

Convergence for the study was monitored using two approaches. Firstly, the Residuals were monitored for the solver and the solution was run until the values dropped to sufficiently low levels (Numerical convergence). Secondly, the lift and drag forces were monitored to ensure that there were no significant oscillations in the measured quantities (Physics convergence).

The residuals are the numerical error associated with the imbalance in the right and left hand sides of the discretized equations at a particular iteration. The equation residuals were observed to drop to the order of $1E-7$ after around 600 iterations and leveling out after that. This residual value was considered adequate for this study.

The residuals can be seen from Figure 10 and the residuals of the forces can be seen from Figure 11 and Figure 12.

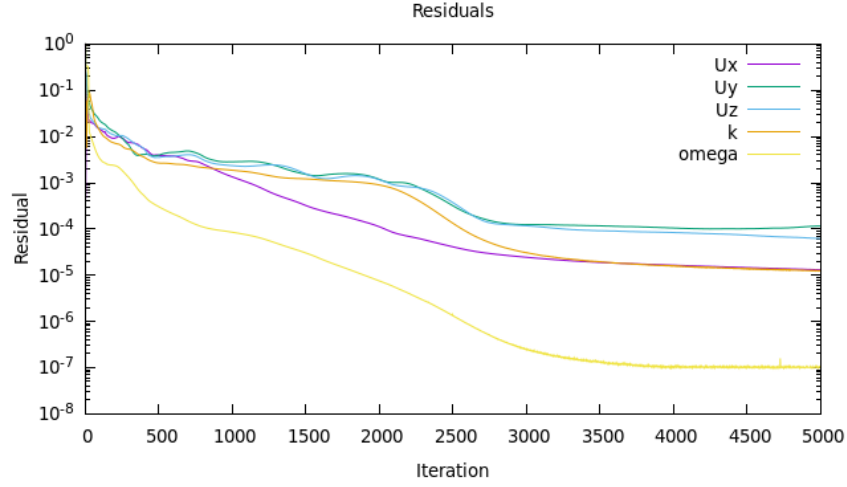


Figure 10: Residual monitoring for the $k-\omega$ SST model with the 1.3 million mesh size.

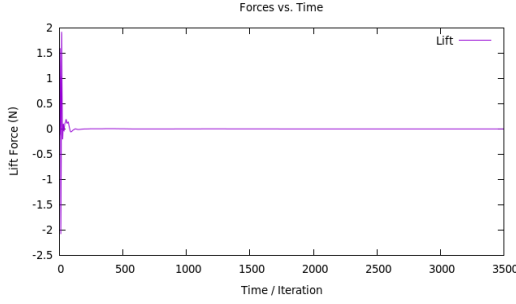


Figure 11: Lift force monitoring for the $k-\omega$ SST model with the 1.3 million mesh size.

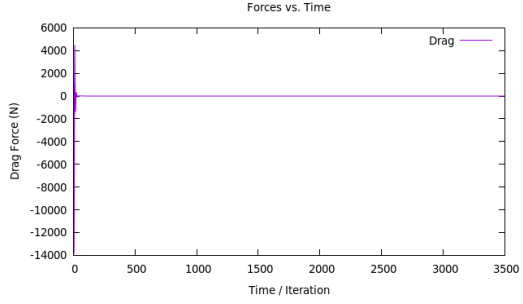


Figure 12: Drag force monitoring for the $k-\omega$ SST model with the 1.3 million mesh size.

4.1.5 Solver Setup

Working with a turbulence model in OpenFOAM requires assigning additional initial and boundary conditions for the turbulence equations, epsilon, k and ω . These are added to the initial conditions for p and U as a sufficient condition to close the equations system. As the focus for the first part of the study was to understand and set-up the simulations for a free-standing propeller exposed to air, the boundary conditions used are summarized in the following tables:

Table 12: Boundary conditions for the FSP case.

Boundary	Type
Inlet	pressureInletOutletVelocity-10 m/s
Outlet	zeroGradient
Cylinder wall	zeroGradient
Propeller	noSlip

The boundary conditions for all the three methods namely the SRF, MRF and the

AMI were maintained the same. The finite volume solver schemes and solutions used for the study are summarized in the Table 13 & Table 14.

Table 13: Finite volume schemes used for the propeller using the $k-\omega$ SST model.

Schemes	Type
ddtScheme	Steady state
gradScheme	Gauss linear
divScheme	bounded Gauss upwind
laplacianScheme	Gauss linear corrected
wallDist	meshWave

Table 14: Finite volume solutions used for the propeller using the $k-\omega$ SST model.

Type	Solver	Smoother
p	PCG	DIC
U	smoothSolver	symGauss Siedel
epsilon	smoothSolver	symGauss Siedel
k	smoothSolver	symGauss Siedel
omega	smoothSolver	sumGauss Siedel

4.1.6 Single Reference Frame(SRF)-FSP

The SRF capability was setup using the *SRFProperties* file for the domain and mesh shown in the previous sections. The *SRFProperties* file had the following operations:

```
SRFModel rpm;

origin (0 0 0);
axis (1 0 0);

rpmCoeffs
{
    rpm 1000;
}
```

The SRF model worked by assigning the angular velocity in terms of a global axis and revolutions-per-minute(RPM). As seen from the file, the RPM specified is 1000 for an axis revolving around the x-axis.

4.1.7 Multiple Reference Frame(MRF)-FSP

The MRF capability of OpenFOAM was setup using the *topoSet* and *MRFProperties* files for the domain and mesh shown in the previous sections. The *topoSet* dictionary file helps create *cellZones* which can be used to distinguish the multiple

frames and it was done the following way:

```
actions
{
  name Propeller-set;
  type cellSet;
  source cylinderToCell;
  sourceInfo
  {
    p1 (-0.3 0 0);
    p2 (-0.2 0 0);
    radius 0.125;
  }
}

{
  name Propeller;
  type cellZoneSet;
  source setToCellZone;
  sourceInfo
  {
    set Propeller-set;
  }
}
```

The first task while creating a *topoSet* file is to name a *cellSet* with any name and assign a source. The source can range from assigning a bounding box to a cylinder or also assign all the cells in the *cellSet*. In the case for the propeller, the source is a *cylinderToCell* source. Since, the source is a cylinder, a radius for the *cellSet* is defined with a start and end point. This concludes the first step after which the *cellZone* is assigned to the propeller by using the *cellZoneSet* command to the source area set before.

With the help of the *topoSet* file, a *cellZone* named *Propeller* is created where the fluid is meant to rotate and the *MRFProperties* for the *cellZone* are applied in the following way:

```
MRF1
{
  cellZone Propeller;
  active yes;
  nonRotatingPatches
  (
    inlet
    outlet
    cylinderwall
  );
  origin (0 0 0);
```

```

    axis (1 0 0);
    omega 104.72;
}

```

The MRF model works by first assigning the *cellZone* which is required to rotate; the Propeller cellZone in our case. Further, patches that do not rotate are assigned in the *nonRotatingPatches* section which are the inlet, outlet and cylinder walls in our case. Finally, like the SRF model, a global axis and the angular velocity, ω is specified. An angular velocity of 104.72 roughly translates to 1000 revolutions-per-minute.



Figure 13: The cellZone cylinders for the MRF and AMI cases. The cellZone for the MRF case was named Propeller same as before whereas for the AMI case was named innerCylinder same as for the propeller.

Figure 13 shows a visual representation of the cylinder zones created for the MRF and AMI cases. It is to be noted that the cylinders for both the cases are of the same diameter.

4.1.8 Arbitrary Mesh Interface(AMI)-FSP

The mesh interface feature was assigned using three features; the first feature involved assigning *cellZones* and *faceZones* which was done using the *snappyHexMesh* feature and is shown below:

```

actions
{
    name innerCylinder;
    faceType boundary;
    cellZone innerCylinder;
    faceZone innerCylinder;
    cellZoneInside inside;
}

```

The second step involved replacing the *cellZones* patches named innerCylinder into AMI patches as the AMI feature of OpenFOAM only works with cyclicAMI patches

for dynamic meshes and was done using the createPatchDict.

```
patches
(
    {
        name AMI1;
        patchInfo
        {
            type cyclicAMI;
            matchTolerance 0.0001;
            neighbourPatch AMI2;
            transform noOrdering;
        }
        constructFrom patches;
        patches (innerCylinder);
    }

    {
        name AMI2;
        patchInfo
        {
            type cyclicAMI;
            matchTolerance 0.0001;
            neighbourPatch AMI1;
            transform noOrdering;
        }
        constructFrom patches;
        patches (innerCylinder-slave);
    }
)
```

With the AMI patches constructed, the final step was to modify the *dynamicMeshDict* to accommodate and rotate the AMI patches.

```
dynamicFvMesh dynamicMotionSolverFvMesh;
motionSolverLibs ("libfvMotionSolvers.so");
motionSolver solidBody;
cellZone innerCylinder;
solidBodyMotionFunction rotatingMotion;
origin (0 0 0);
axis (0 1 0);
omega 104.72;
```

The motion solver used is the *solidBody* motion solver with a *rotatingMotion* function.

4.2 Dji Phantom 3 Quadcopter

4.2.1 Geometry

The Phantom series from Dji was used as the reference model to study the effects of rotation on a quadcopter. The quadcopter is essentially constructed in an x-shaped frame with 4 motors and propellers mounted to the end of each arm. Two fixed stands on each side of the quadcopter are provided to help land the copter with ease. A camera is mounted towards the front section of the quadcopter with a separate stand at a small distance from the bottom of the x-frame.

Three different cases of the Phantom 3 were studied; the complete Phantom 3 with the camera and the landing side stands as seen from Figure 14, a simplified Phantom 3 with the x-frame and motor and propellers as seen from Figure 15 and finally, a single propeller(Phantom CF propeller), Figure 16, used by the Phantom 3 model.



Figure 14: Top iso-view of the entire Dji Phantom 3.



Figure 15: Simplified model of the Dji Phantom 3.

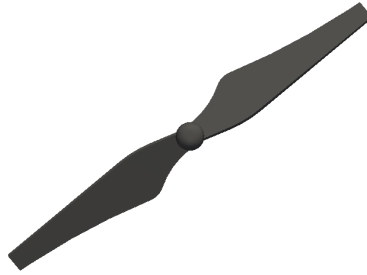


Figure 16: Single propeller of the Dji Phantom 3.

4.2.2 Domain

The domain used to study the quadcopter is shown in Figure 17, where the quadcopter is shaded in black. A rectangular domain was used with its specifications presented in Table 15. The domain height was adjusted after it was seen to interfere with the wake generated and additionally, it is to be noted that the quadcopter was placed $0.2[m]$ above the point of origin.

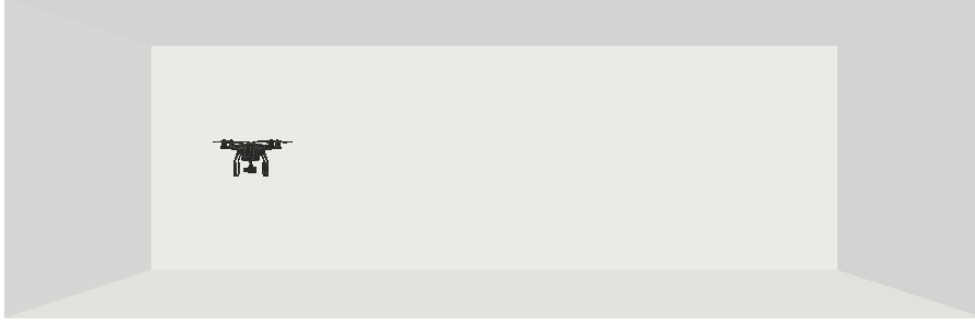


Figure 17: Side view of the domain used for the Dji Phantom 3 with the quadcopter(shaded black).

Table 15: Global domain size for the quadcopter.

Respective Lengths	Units [m]
Length of Domain (L)	4.3
Upstream Length (L_f)	0.865
Width (L_b)	3
Height (L_h)	2.18

4.2.3 Mesh

A mesh independence study was performed for the quadcopter with a series of meshes of different sizes. A rotational speed of the rotors of 3600 RPM was chosen with the quadcopter at an ideal hover condition. The thrust, drag and torque produced by the quadcopter for each mesh were used to evaluate the mesh independence. The parameters mentioned above are shown in Table 16 for the different mesh sizes.

Table 16: Mesh independence study for the quadcopter.

Mesh Size	Thrust [N]	% Diff.	Drag [N]	% Diff.	Torque [Nm]	% Diff.
376,000	0.159	-	1.003	-	0.217	-
1,020,000	0.178	10.6%	0.903	9.97%	0.356	39%
3,200,000	0.521	65.8%	0.602	5%	0.527	32.4%
6,500,000	0.825	36.8%	0.475	26.7%	0.677	22.1%
8,700,000	0.793	3.87%	0.454	4.62%	0.708	4.37%

As seen from Table 16, the variation in the difference between the three parameters after the mesh of size 6.5 million is lesser than 5%. This difference was deemed satisfactory and the mesh with a global size of 6.5 million was concluded as the mesh independent case.

A front and top view of the grid(6.5 million mesh count) on the quadcopter can

be viewed in Figure 18 while the important mesh characteristics are presented in Table 17.

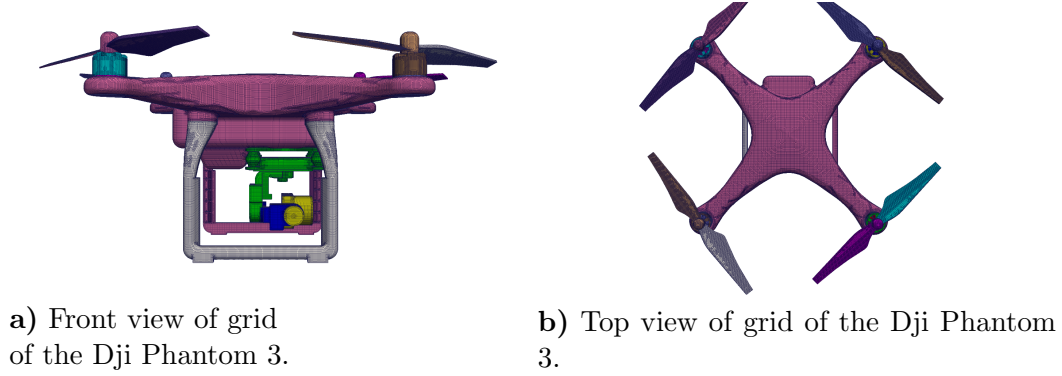


Figure 18: Overset mesh grid on the Dji Phantom 3 model.

Table 17: Mesh properties for the 6 million mesh count case.

Properties	Values
Size	6.5 million
Max. aspect ratio	7.421
Avg. orthogonality	6.862
Max. skewness	5.027 (5 skewed faces)

4.2.4 Solver Setup

An incompressible, steady-state solver was used for the study of the quadcopter with the $k-\omega$ SST turbulence model[2]. The boundary conditions used for the quadcopter are summarized in Table 18.

Table 18: Boundary conditions for the quadcopter using the MRF and AMI methods.

Boundary	Type
Inlet	fixedValue-0/5/10 m/s
Outlet	inletOutlet
Left/Right	slip
Top/Bottom	slip
Propeller blades	movingWallVelocity-0 m/s
Remaining quad	noSlip
AMI1&2	cyclicAMI (only for AMI)

The finite volume schemes and solvers used for the study are further summarized in Table 19 & Table 20.

Table 19: Finite Volume schemes for the quadcopter.

Schemes	Type
ddtScheme	Steady state
gradScheme	Guass linear
divScheme	Bounded gauss upwind
laplacianScheme	Gauss linear corrected
wallDist	meshWave

Table 20: Finite volume solvers for the quadcopter.

Type	Solver	Smoother
p	GAMG	Guass Siedel
U	GAMG	Guass Siedel
yPsi	smoothSolver	Guass Siedel
k	smoothSolver	Guass Siedel
omega	smoothSolver	Guass Siedel

nonOrthogonalCorrectors of the order 10 and relaxation factors for equations of U,k and omega were also assigned with the help of the *fvSolution* dictionary file.

4.3 MRF Dictionary- Quadcopter

Assigning the rotating and stationary part for the study of the quadcopter was done in 4 distinct parts. As the quadcopter could be divided into 4 equal parts , when the camera mount and the side stands are not present the quadcopter has double symmetry. The features used are shown for $1/4^{th}$ of the drone with the arm. The camera mount and side stands were included as parts that did not require any rotation. Figure 19, shows the arm along with the propeller attached to the motor and only the propeller required rotation in this case.

**Figure 19:** $1/4^{th}$ section of the quadcopter, used to assign patches for the MRF properties file .

The first step involved assigning an arbitrary rotating cylinder around the propeller with the help of the *topoSetDict* file.

```
actions
{
    name Propeller1-set-quad;
    type cellSet;
    source cylinderToCell;
    sourceInfo
    {
        p1 (-0.13 -0.121 0.078);
        p2 (-0.13 -0.121 0.102);
        radius 0.121;
    }
}

{
    name Propeller1;
    type cellZoneSet;
    source setToCellZone;
    sourceInfo
    {
        set Propeller1-set-quad;
    }
}
```

With the help of the *topoSetDict*, a cylinder of radius 0.121 [m] was constructed around Propeller 1 and assigned a *cellZone* to be used for the *MRFProperties* file. The *MRFProperties* file for 1/4th of the quadcopter is shown below:

```
MRF1
{
    cellZone Propeller1-set-quad;
    active yes;
    nonRotatingPatches
    (
        quad-arm
        quad-motor
    );
    origin (-0.195 -0.1815 0.141);
    axis (0 0 0.0239);
    omega -376.99;
}
```

It can be seen from the *MRFProperties* file that, only the propeller was assigned as the rotating region while the *nonRotatingPatches* included the quad arm and motor region. The creation of *cellZones* using the *topoSetDict* was carried out for

the entire quadcopter and the same applied for assigning *nonRotatingPatches*. The camera mount and side stands were also assigned as *nonRotatingPatches*.

4.4 AMI Dictionary- Quadcopter

A simplified $1/4^{th}$ model, similar to Figure 19 was only used for the AMI case as the source file of OpenFOAM would have to be modified to enable multiple rotating elements to take place and this was beyond the scope of this thesis.

The AMI setup for the quadcopter was similar to the setup used for the free-standing propeller. The *snappyHexMeshDict* file was use to assign *cellZones* and *faceZones*. The *cellZones* were then replaced with AMI patch names using the *createPatchDict*. Finally, the *dynamicMeshDict* was used to assign the *cellZone* as a rotating motion with a specified RPM. The difference in setup for the quadcopter was the value of angular velocity used and the boundary conditions.

5 Results

5.1 Free-standing Propeller

5.1.1 Velocity Profiles

The u-velocity was plotted for the three methods and is shown in Figure 20. The wake generated for the SRF and the MRF ($k-\omega$ SST) methods are more profound and propagate for a certain distance before settling down in the domain. Flow separation just behind the propeller was noticed again for these methods evidently. The difference between the SRF and the MRF ($k-\omega$ SST) was that the SRF's wake was seen narrowing down behind the propeller while the MRF ($k-\omega$ SST)'s wake could be seen growing. The MRF ($k-\epsilon$) method had flow separation of a much smaller magnitude behind the propeller and the wake generated was rather small.

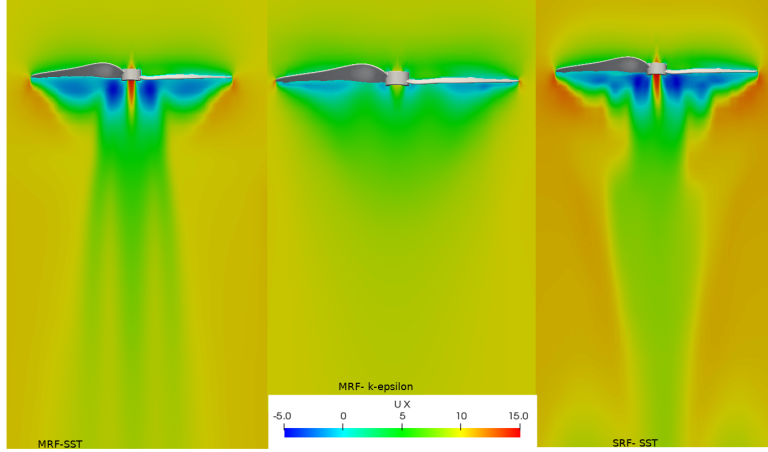


Figure 20: u-velocity comparison for the SRF and MRF methods. Left, a) MRF- SST method; Center, b) MRF- k-epsilon method; Right, c) SRF- SST method

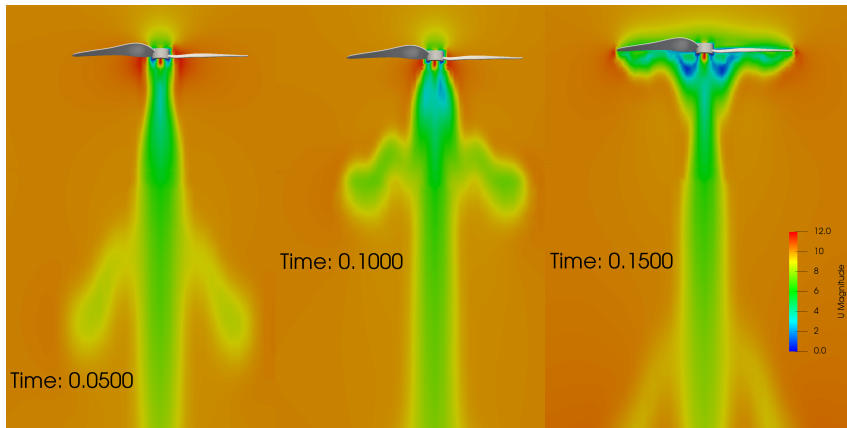


Figure 21: u-velocity time profile for the AMI method.

The time profiles for the AMI method are shown in Figure 21. At time, $t = 0.05[s]$, the wake from the propeller had two "extensions" (either side of propeller) that

were seen after the rotating region. These "extensions" over time, $t = 0.1[s]$ came closer to the rotating region. When the flow was developed at time, $t = 0.15[s]$, the "extensions" settled down close to the propeller region. The wake generated by the AMI method was found to be narrower when compared to the other two methods, suggesting a higher momentum generation.

The mesh were sliced at different locations along the length of the domain, shown in Figure 22, and the x-velocity was plotted in the resulting planes. The plots help analyze the propagation of the wake along the domain. It was seen that the effect of the propeller wake propagates for a distance of about $300[mm]$ for both methods and after which it dissipates and the flow becomes more uniform. The effect of the wake can be seen to be a bit more "scattered" for the SRF method, Figure 24, whereas a uniform profile was seen for the MRF method, Figure 23.

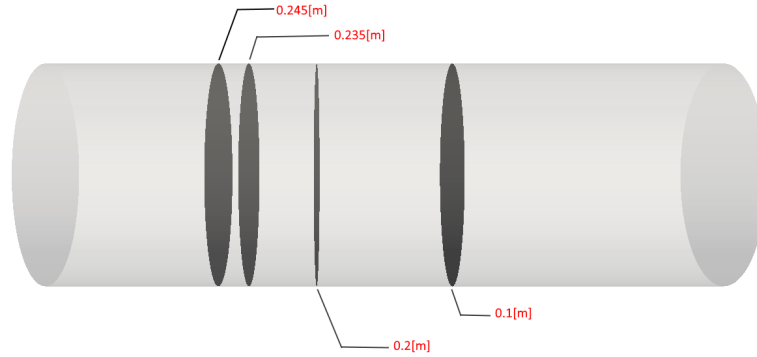


Figure 22: Sliced planes along domain length. Clockwise from left top: a) Slice plane at $0.245[m]$, b) Slice plane at $0.235[m]$, c) Slice plane at $0.2[m]$, d) Slice plane at $0.1[m]$

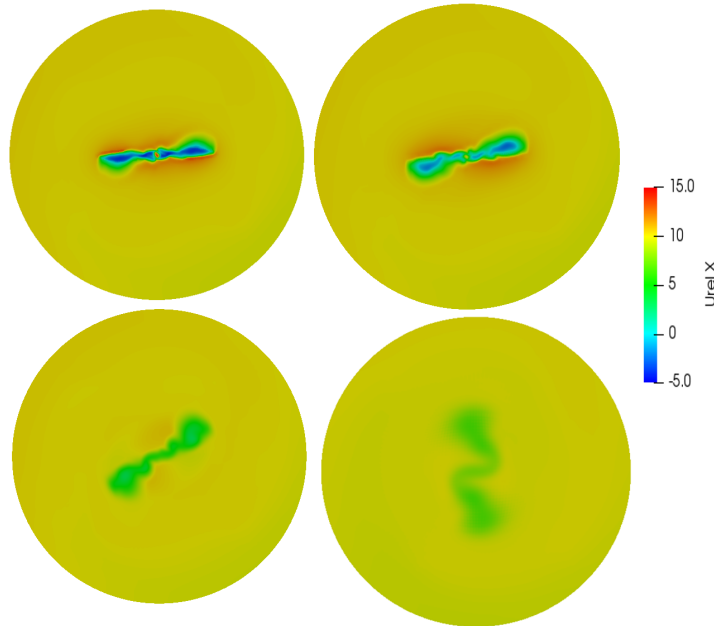


Figure 23: MRF- x velocity. Clockwise from left top: a) Slice plane at $0.245[m]$, b) Slice plane at $0.235[m]$, c) Slice plane at $0.2[m]$, d) Slice plane at $0.1[m]$

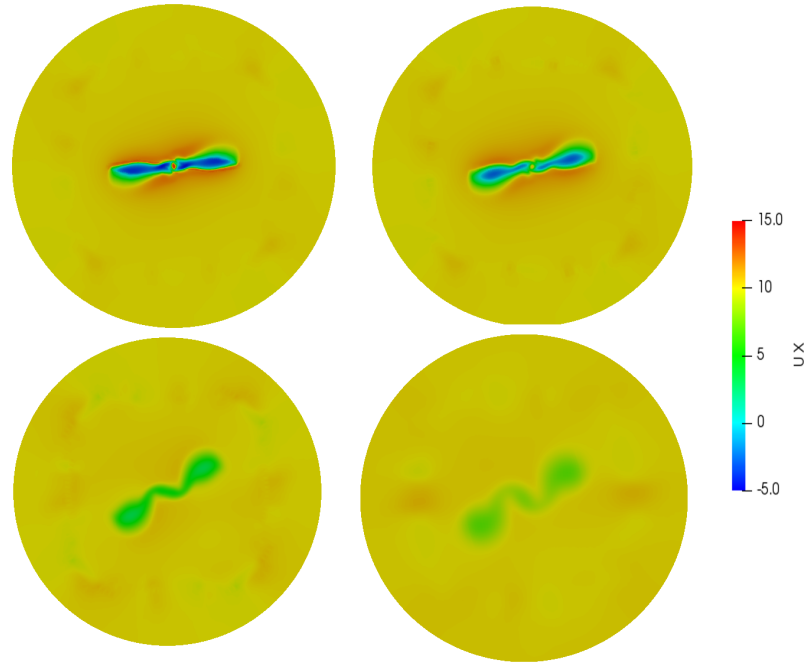


Figure 24: SRF- SST- x velocity. Clockwise from left top: a) Slice plane at 0.245[m], b) Slice plane at 0.235[m], c) Slice plane at 0.2[m], d) Slice plane at 0.1[m]

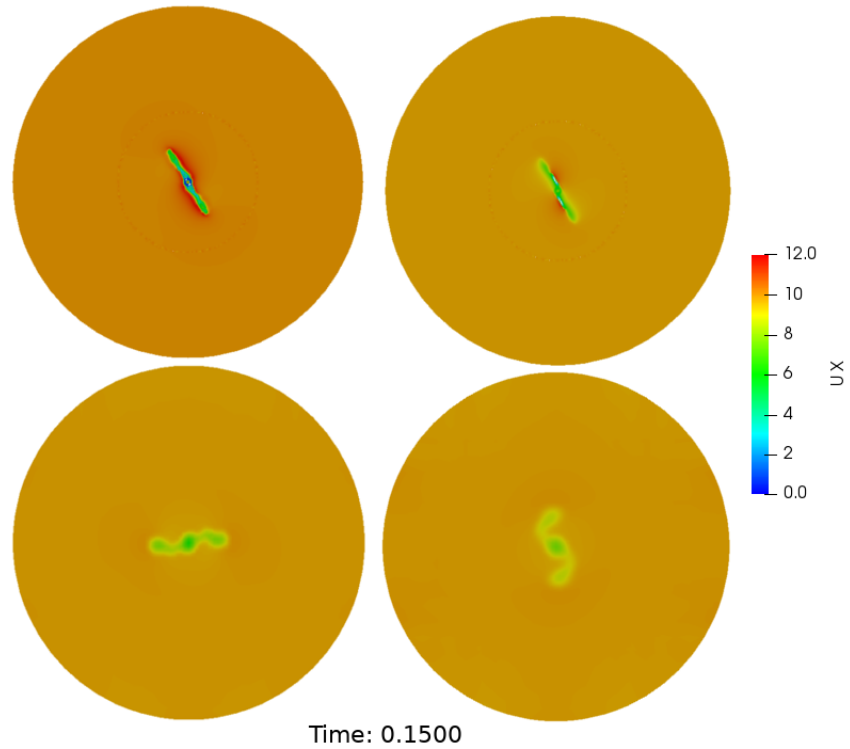


Figure 25: AMI- SST- x velocity. Clockwise from left top: a) Slice plane at 0.245[m], b) Slice plane at 0.235[m], c) Slice plane at 0.2[m], d) Slice plane at 0.1[m]

Figure 25 shows the wake after a time of 0.15[s] when the flow had become fully

developed. The wake for the AMI method is seen to have small peaks (red colour) nearer to the propeller. The profile of the wake downstream ($0.1[m]$) can be seen to include the effect of the propeller hub which was not evident in the other methods.

5.1.2 Experimental Comparison

Experimental results for the thrust and torque generated by the rotating propeller were compared with the numerical approach for the three methods. Experimental tests were conducted for 15 different RPMs at varying speeds and are available from [15]. 2 different RPMs were chosen while 4 varying inlet velocities were chosen for the comparisons and are shown in Figure 26. When comparing the thrust produced by the SRF method to the experimental thrust produced, a maximum deviation of around 66% was noticed for an RPM of 2000 with a speed of $5[m/s]$. The lowest deviation was seen for a speed of $2[m/s]$ with an RPM of 1000. Further, it can be seen that higher the RPM and the speed, more the deviation in results.

The MRF method showed significantly better results with a maximum deviation of around 12% for a speed of $1.5[m/s]$ with an RPM of 1000. The results improved when the speed was increased and the deviation fell down to around 7% for the remaining speeds and RPMs. The AMI method had great correlations with the experimental data as maximum deviations of around 5% were seen for an RPM of 2000.

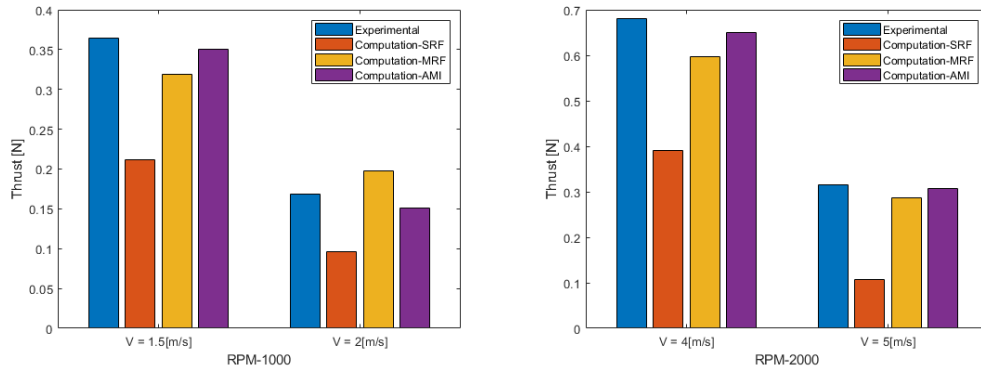


Figure 26: Experimental comparison of thrust produced at 2 chosen RPMs for different inlet velocities.

The results for the torque produced by the propeller for both the experimental and numerical approach are shown in Figure 27. A similar trend was seen for the torque produced for the SRF method when compared to the thrust produced. A maximum deviation of 57% was noticed for an RPM of 2000 with a speed of $4[m/s]$.

The MRF method had significantly better results with a maximum deviation of around 10% when compared to the experimental torque. For a lower speed and corresponding RPM, the deviation was found to be lesser than 6%. The AMI method, as seen from the thrust comparison had the highest accuracy. Very small deviations in the order of 4% were noticed.

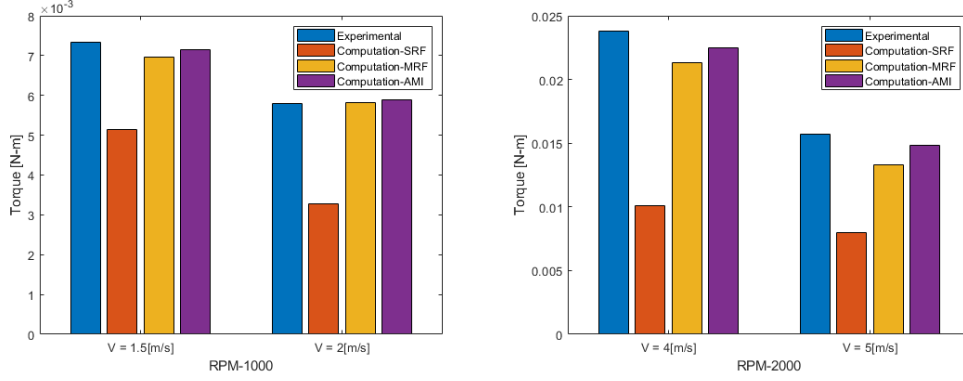


Figure 27: Experimental comparison of torque produced at 2 chosen RPMs for different inlet velocities.

5.1.3 Computational Time

The numerical simulations for the free-standing propeller were run on 32 cores of AirShaper’s platform. The time taken to run each simulation for the different methods is shown in Table 21. It can be seen from Table 21 that there are two time periods shown; the execution time and clock time. The execution time or CPU time is the elapsed CPU time or the time spent by the processor to run the simulations while the clock time is the elapsed wall clock time. The difference between the two time periods is due to the time spent to write the log files and also process the communication time between the processors in parallel run.

Table 21: Computational time taken to execute each respective method for the free standing propeller.

Method	Sub-divisions	Execution Time [s]	Execution Time [hr]	Clock time [s]
SRF	32	22787	6.329	22810
MRF($k-\omega SST$)	32	23076	6.41	23125
MRF($k-\epsilon$)	32	22613	6.28	22240
AMI	32	47283	13.13	50047

From Table 21, it is seen that there exists only a mere 9 minute difference between the execution times for the SRF and MRF methods. The AMI method on the other hand took 13 hours or roughly twice the time as the MRF and SRF methods, to execute and converge to a steady-state solution.

5.2 DJI Phantom 3

In this section the results obtained in the study of the DJI Phantom 3 drone are analyzed. First a single rotor of the Phantom 3 is studied followed by the simplified model for the different methods. Following this, the method that provides the best results among the three is chosen. Finally the selected method is applied to different scenario simulations in order to verify its capabilities.

5.2.1 Single Rotor

A single rotor of the Dji Phantom 3 was studied with the three methods and compared with experimental data for different RPMs in hover condition. RPMs of 3600, 4500, 5400, 6300 and 7200 were compared for the thrust produced and are shown in Figure 28. An excellent agreement is seen for the AMI method with a less than 3% difference when compared with the experimental results. The MRF method also shows good agreement with a maximum difference of 8.5% occurring at an RPM of 7200. The SRF method shows a good agreement with experimental data for low RPMs but the margin increases with the increase in RPM.

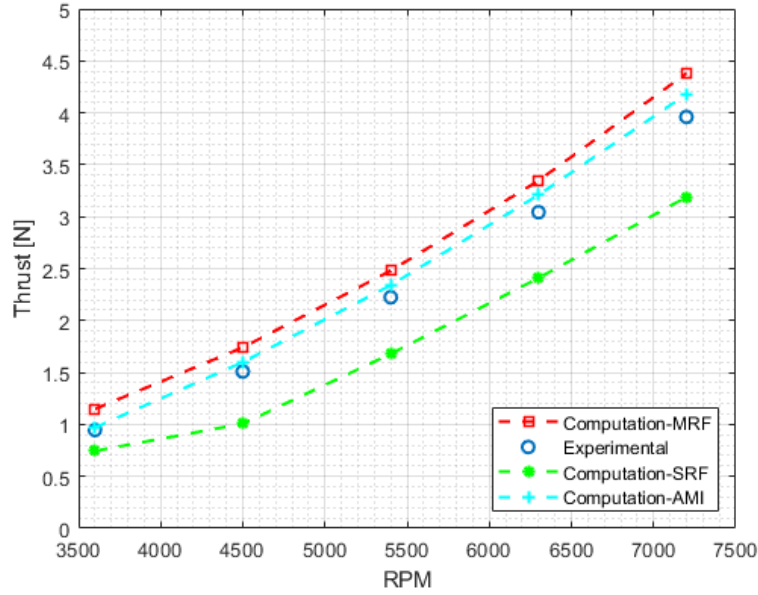


Figure 28: Comparison of Thrust with Zawondy’s experimental data[16] for the Dji Phantom 3 rotor at various RPMs.

Table 22 shows the computational time taken to execute each method. As discussed earlier, the difference between the MRF and SRF method is small and can be seen to be around 40 minutes for the execution time. On the other hand, the AMI method takes twice as much time as the MRF method. The time taken to write the log files and process the parallel computations was seen to increase by a big margin especially for the AMI method as the clock time took approximately 1 day to write and process the log files while the MRF and SRF method took 12-13 hours.

Table 22: Computational time taken to execute the SRF,MRF and AMI method for the propeller of the DJI model.

Method	Sub-divisions	Execution Time [s]	Execution Time [hr]	Clock time [s]
SRF	32	31187	8.66	48750
MRF	32	34082	9.46	51584
AMI	32	69741	19.37	87452

5.2.2 Simplified Dji Phantom 3

The simplified model of the Dji Phantom 3 was simulated with the MRF and AMI methods and is presented in this section. First, the thrust produced from the rotor and the quadcopter as a whole is compared against data from NASA’s modeling of small quadcopters[16]. Figure 29 shows the thrust produced by all 4 rotors at hover condition for an RPM of 3500. The difference in thrust produced between Numerical simulations from [16] and the MRF method is around 10%. Figure 30 shows the total thrust produced by the simplified model. A similar trend was seen for the total thrust as was seen for the rotor thrust.

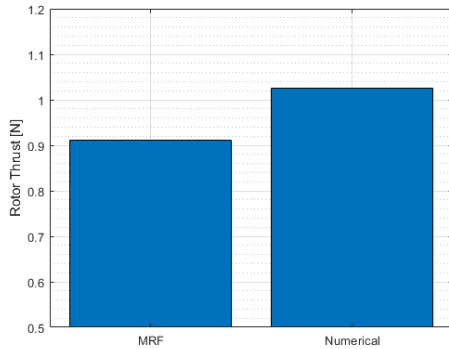


Figure 29: Rotor Thrust comparison for MRF and AMI methods.

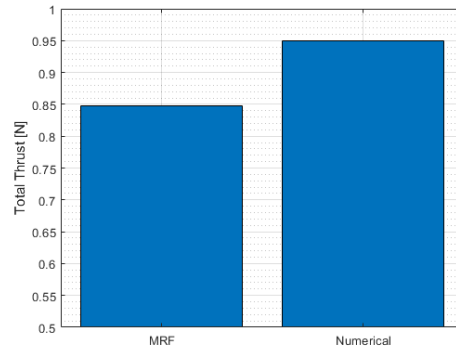


Figure 30: Total Thrust comparison for the MRF and AMI methods.

The simplified quadcopter model with the MRF method took approximately 17 hours to simulate with 32 cores. It can further be seen from Table 23 that an additional 4 hours was needed for the clock time execution for the MRF method.

Table 23: Computational time taken to execute the MRF method for the simplified DJI model.

Method	Sub-divisions	Execution Time [s]	Execution Time [hr]	Clock time [s]
MRF($k-\omega SST$)	32	61582	17.1	83260

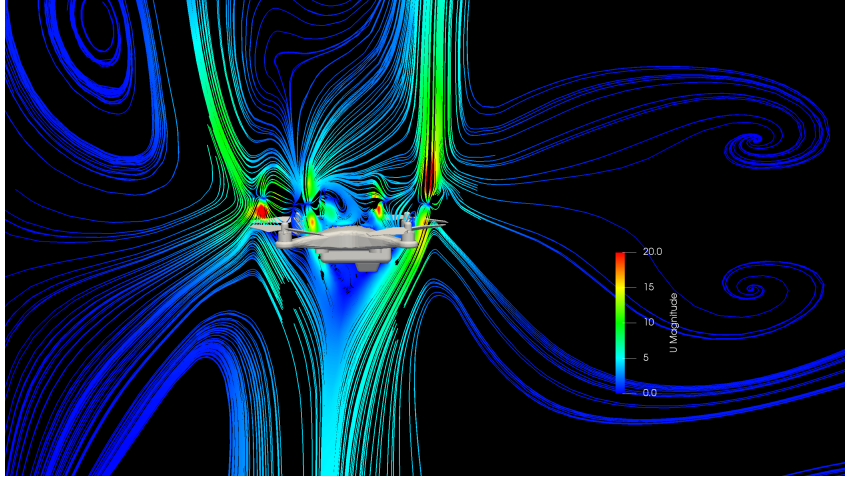


Figure 31: Velocity streamlines for the Simplified DJI Phantom 3 in hover implemented with the MRF model.

5.2.3 Comparison of different configurations

The complete and simplified models of the quadcopter were simulated with the MRF method at hover conditions. Velocity streamlines can be seen in Figure 32 and Figure 31 for the complete and simplified models. As expected, the velocity close to the propeller increases as the rotating propeller energizes the flow and the wake produced by the propeller is seen to propagate downstream. Further, the stand and camera can be seen to decelerate the flow and a small re-circulation zone is created behind the camera setup. This impacts the total thrust produced and is additionally the reason for the lower thrust seen in Figure 33. The simplified model on the other hand, has no battery and stands and has an almost undisturbed wake profile downstream which explains the small difference between the rotor and total thrust produced.

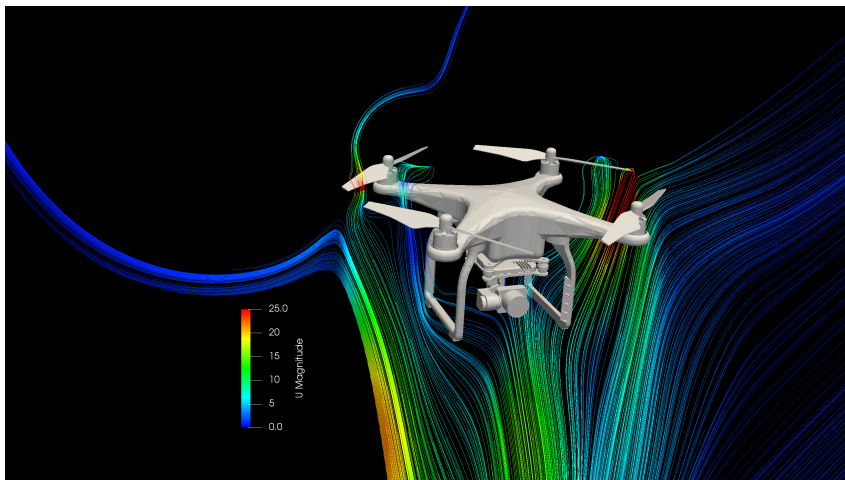


Figure 32: Velocity streamlines for the complete DJI Phantom 3 with the MRF method.

The thrust co-efficient is calculated using the formula:

$$c_T = \frac{T}{0.5 * \rho_{\infty} * A * (\sigma * R)^2} \quad (14)$$

where, T is the thrust, ρ_{∞} is the density, A is the reference area, σ is the rotational velocity and R is the radius of the propeller of the Dji Phantom 3.

Figure 33 shows a comparison between the thrust coefficient for the different components of the quadcopter model. It can be seen that the four rotors individually produce more thrust than the quadcopter as a whole. The thrust coefficient of the simplified Dji Phantom 3 is around 1.5% more than the complete model.

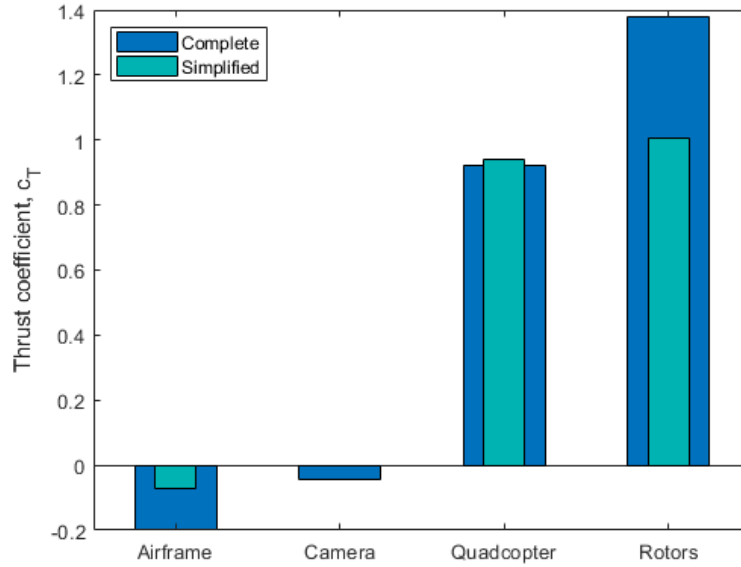


Figure 33: Comparison of thrust coefficient for the different components of the DJI Phantom 3 for the complete and simplified configurations.

Figure 34 shows the surface pressure on the entire body of the Dji Phantom 3 along with contours of the velocity magnitude on the model. High pressure zones can be seen in the battery and landing gear section which are in the wake region. This causes the quadcopter to be pushed down further reducing the overall thrust produced and this can be proved further from Figure 33, where the simplified quadcopter produces around 1.5% more thrust than the complete quadcopter.

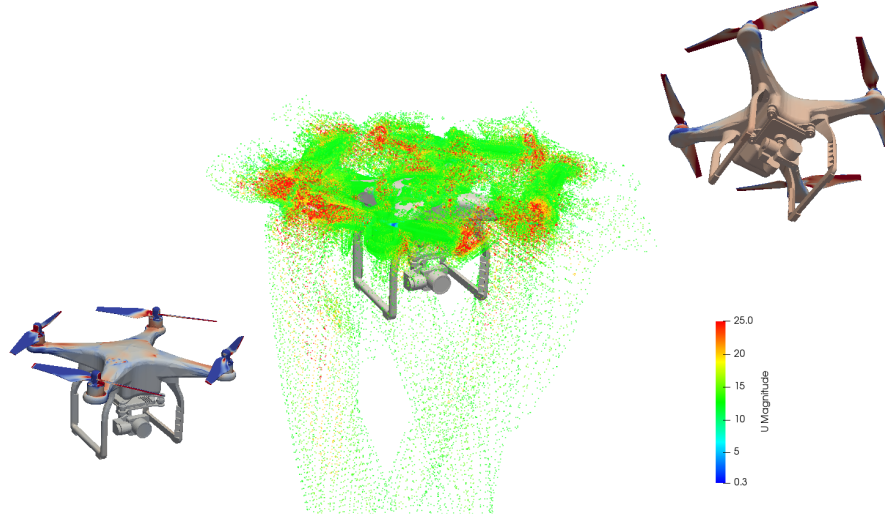


Figure 34: Complete DJI Phantom 3 in hover with velocity magnitude represented by points. Surface pressure is shown on the body with red indicating high pressure and blue indicating low pressure.

Figure 35 shows a similar flow behaviour as Figure 34 with the simplified model. The wake represented by contour points is seen to be a bit more uniform as the battery and landing gear section do not interfere with the wake and this in turn helps produce a bit more thrust.

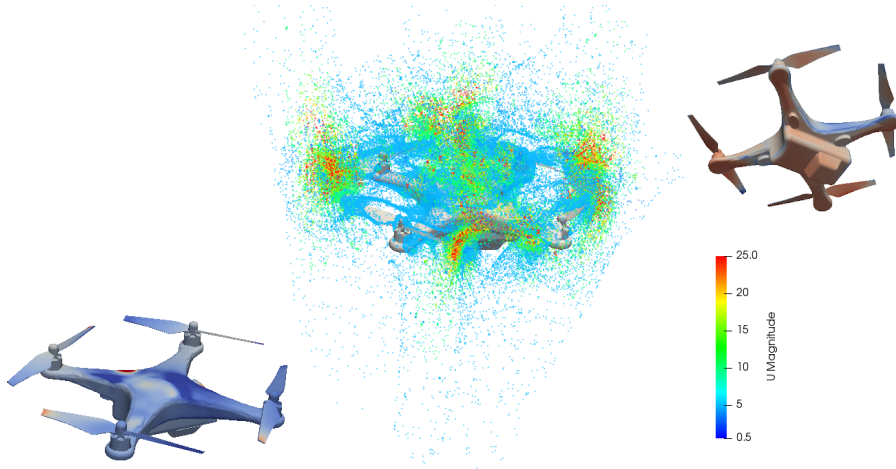


Figure 35: Simplified DJI Phantom 3 in hover with velocity magnitude represented by points. Surface pressure is shown on the body with red indicating high pressure and blue indicating low pressure.

5.2.4 Effects of variation in MRF Cylinder radius

The cylinder radius of the rotating region for the MRF method was varied to study its influence in the simulation. The original radius of the cylinder was $0.121[m]$ that enclosed the propeller perfectly. Two increments of 5% and 10% were tested to see if the results impacted the performance of the MRF method. The thrust and drag produced by the different variations are presented in Figure 36 and Figure 37. As

the radius of the rotation area increased, the drag increased consistently and the cylinder with an increase in 10% radius had an overall drag of $1.3[N]$. The thrust on the other hand had no difference in values for the original and the 10% radius increase case. The case with the 5% increase in radius saw a 15% increase in thrust.

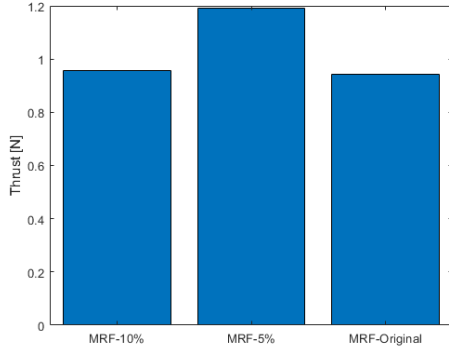


Figure 36: Total Thrust comparison for variation in MRF cylinder radius.

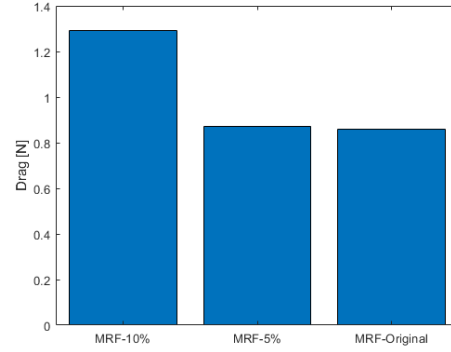
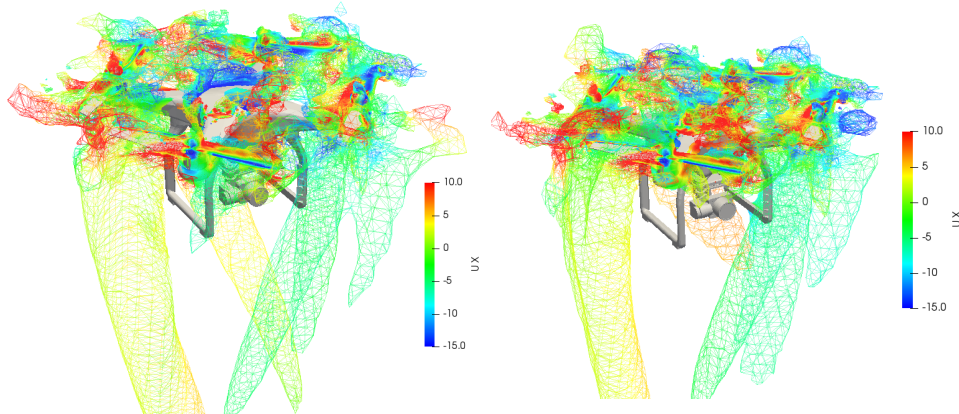


Figure 37: Drag comparison for variation in MRF cylinder radius.

x-velocity contours were plotted on the quadcopter and are shown in Figure 38. The aerodynamic interactions on top of the quadcopter body are seen to be a lot more chaotic for the case with a larger cylinder radius. The original radius shows the wake from the propeller rushing downwards with some interaction on the arms of the Dji Phantom 3. However, the interactions seem much stronger on the body and arms when the radius was increased.



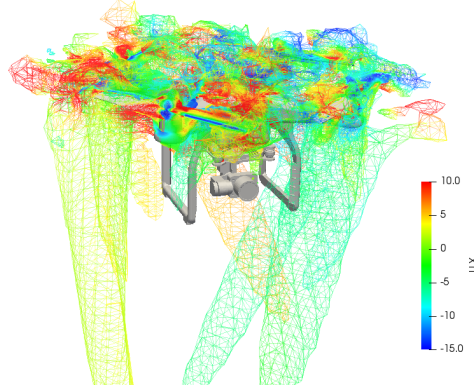


Figure 38: x-velocity contours for the three cylinder radius's. From left top, a)Original radius, right top b) 5% increase, bottom c) 10% increase.

5.2.5 Effects of weather

The complete quadcopter was simulated under effects of wind gusts for the hover condition. 3 different wind speeds were simulated with a maximum wind speed of 20[kts] or 10.28[m/s]. The maximum speed of 20[kts] was sustained as this was equivalent to a fresh breeze where leaves in trees begin to sway[17]. A wind speed beyond this would impact the performance of the quadcopter as small quadcopters are known to sway away with high wind speeds.

Figure 39 shows the wind speed direction and rotor indication. Figure 40 shows the thrust coefficient for each rotor for the different wind speeds. It was observed that the thrust co-efficient and in turn the thrust increased as the wind speed increased for rotors 1, 2 and 4. Rotor 3's thrust coefficient had a very slight increase with wind speed and almost remained constant.

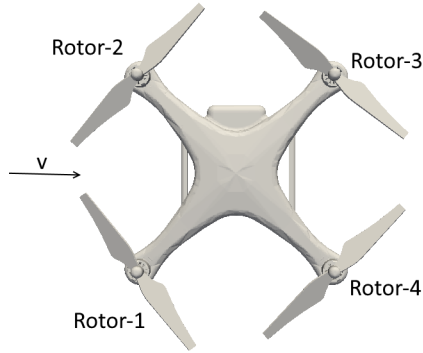


Figure 39: Rotor and wind speed indication.

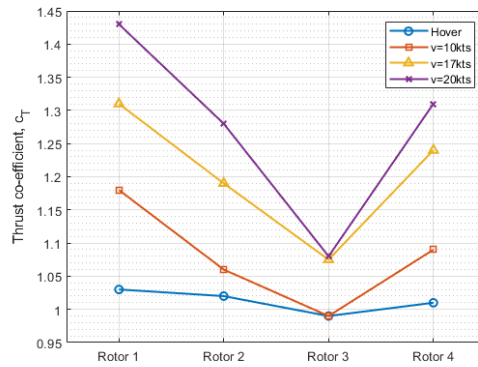
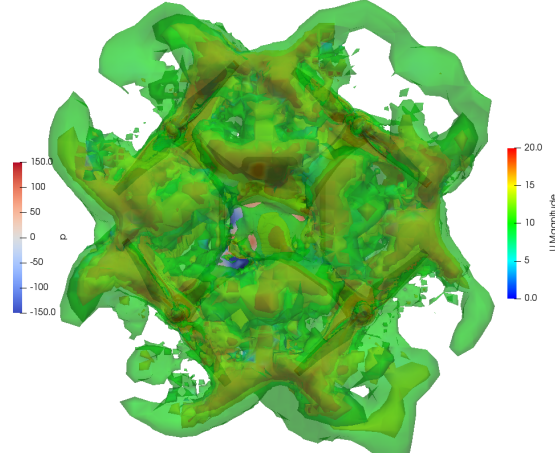


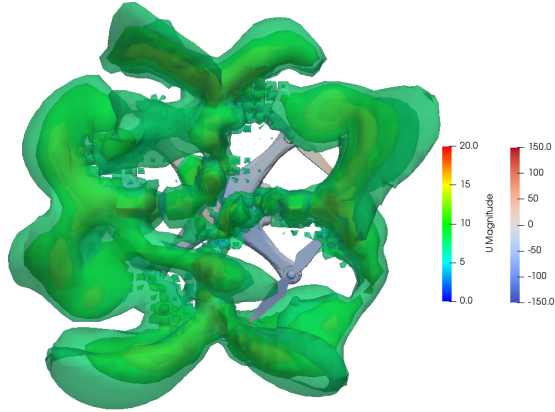
Figure 40: Thrust co-efficient for each rotor.

Top view contours of the velocity can be viewed from Figure 41 for the different wind speeds. Additionally, surface pressure was plotted on the body of the quadcopter with blue indicating low pressure and red indicating high pressure. It was seen from the figures, that a high velocity region exists on Rotors 1 and 2 for the wind speeds

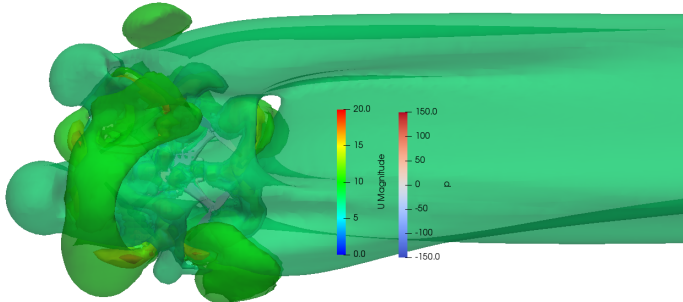
of 5, 17 and 20 [kts]. This explained the high thrust on these rotors. The thrust for the case with the hover condition and no wind speed can be seen to have an almost equal velocity profile all throughout the quadcopter which additionally be seen from Figure 40 where, the thrust co-efficients were almost the same for all 4 Rotors.



a) $V_{wind} = 0kts$



b) $V_{wind} = 10kts$



c) $V_{wind} = 17kts$

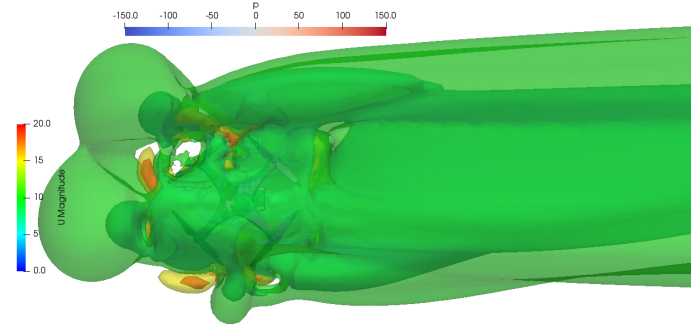
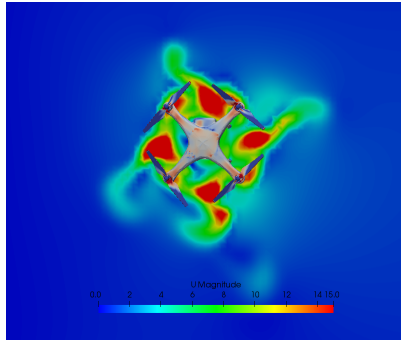
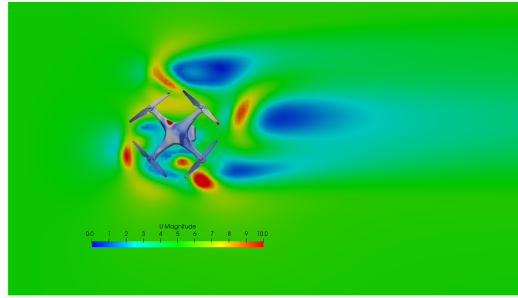


Figure 41: Dji Phantom 3 under effect of wind gusts of 10, 17 and 20 kts with the surface pressure shown on the Dji Phantom 3.

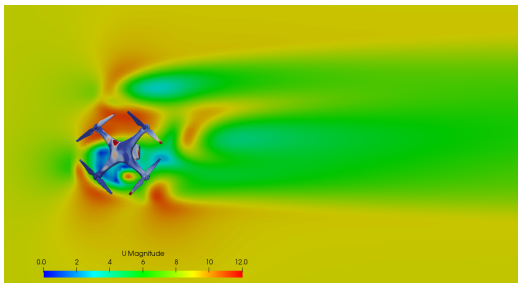
Figure 42 shows the velocity magnitude on a z-plane with the top view of the quadcopter painted with surface pressure. At a wind speed of $V_{wind}=0$ [m/s], the wake can be seen going downwards and extending only to the size of the quadcopter. As the wind speed increases, the wake can be seen pushed along the direction of the wind. Wind speeds of 17 and 20 [kts] show a large wake formation that is seen settling after a distance of about 500[mm].



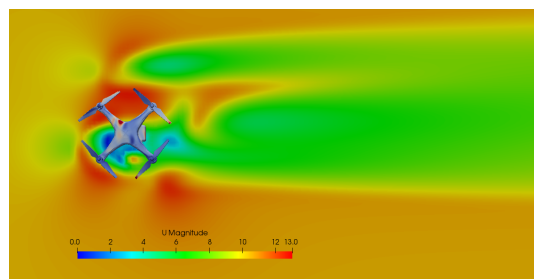
a) $V_{wind} = 0$



b) $V_{wind} = 10kts$



c) $V_{wind} = 17kts$



d) $V_{wind} = 20kts$

Figure 42: Velocity magnitudes shown on z-plane with different wind speeds and surface pressure shown on the Dji phantom 3(Top view).

5.2.6 Effects of pitch and weather on the Simplified Dji Phantom 3

Two pitch angles, 10° and 15° were tested for an RPM of 3600 with a wind velocity of $10[m/s]$. Streamline profiles around the Dji Phantom 3 for the two angles are presented in Figure 43 and Figure 44. The velocity profile on a z-plane in these figures shows the extent of the wake being pushed due to the wind velocity. Also, it was seen that for 15° pitch case, Rotors 1 and 2 have higher interactions and produce higher thrust compared to the 10° pitch case.

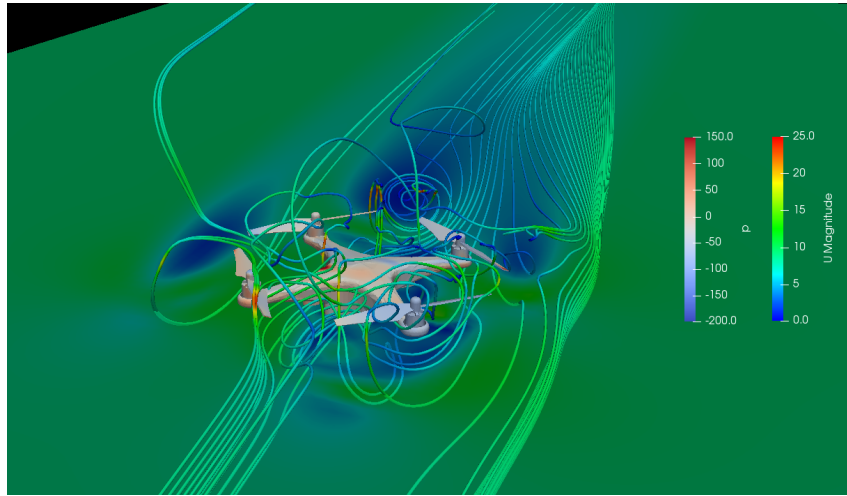


Figure 43: Surface pressure on the body with velocity streamlines on a z-plane of velocity magnitude for 10° pitch angle.

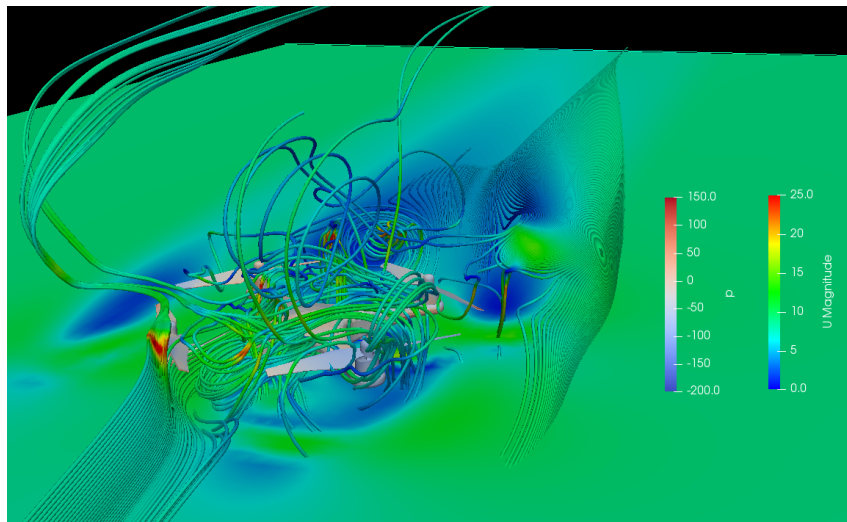


Figure 44: Surface pressure on the body with velocity streamlines on a z-plane of velocity magnitude for 15° pitch angle.

5.2.7 Overmount and Undermount configuration comparisons

The overmount and undermount configurations were compared for the thrust produced in this section. The undermount configuration had the rotors placed on the underside of the fuselage as seen from Figure 46. Table 24 shows the comparison of the thrust produced for the two different configurations along with the comparison of the numerical data from [16]. Firstly, there was around an 8% difference in total thrust produced between the two configurations while the difference between the thrust produced by the rotors was around 2%. The downforce produced by the undermount configuration was larger than the downforce produced in the overmount case. The numerical comparison between the overmount and undermount showed a difference of around 13% and 15% for the total thrust produced.

Table 24: Total thrust for the two quadcopter configurations along with a comparison from [ref] for the total thrust produced.

Configuration	Rotor Thrust [N]	FuselageDown-force [N]	Total Thrust [N]	Numerical Comparison [16][N]
Overmount	0.869	-0.051	0.818	0.949
Undermount	0.847	-0.06	0.787	0.936

Velocity streamlines for the simplified Dji Phantom 3 model were plotted for the overmount and undermount cases and are shown in Figure 45 and Figure 46.



Figure 45: Velocity streamlines on the simplified overmount Dji Phantom 3.

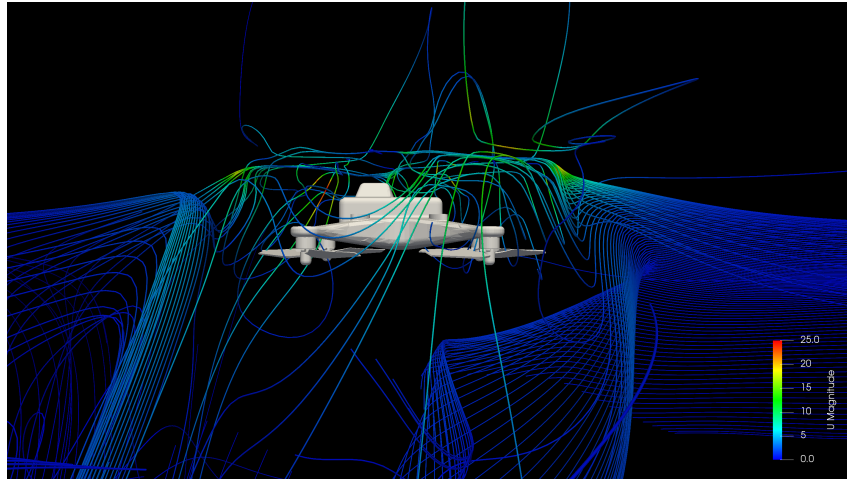


Figure 46: Velocity streamlines on the simplified undermount Dji Phantom 3.

6 Discussion

6.1 FSP

6.1.1 Velocity magnitudes

The principle behind the SRF method was to rotate the flow in the entire domain for the RPM specified and this lead to the velocity along the sides of the domain to remain at that RPM ,thus, producing a more "scattered" velocity profile as seen in Figure 24. Additionally, the wake generated by the SRF method, Figure 20, was observed to propagate nearly to the outlet indicating flow dissipation which additionally can be substantiated with Figure 6. The MRF and AMI methods on the other hand had a more uniform wake profile that dissipated before it reached the end the outlet section, Figure 20. The $k-\epsilon$ model with the MRF method produced a wake with a very small order of velocity, whereas the $k-\omega$ SST model's wake profile was more realistic, Figure 23. Further, Figure 21 showed the narrowest wake for the AMI method with short peaks occurring at regular intervals which extended to the size of the propeller.

6.1.2 Experimental Comparisons

The thrust and torque produced by the APC propeller was compared for 2 different RPMs and 4 different inlet velocities. As seen in Figure 26 and Figure 27, the SRF method was observed to have the largest difference in values. This could be explained by SRFs need to solve the governing equations for rotating regions in the entire domain and not specifically pertain to the propeller region. The MRF method proved to be much more reliable when compared to the experimental data as deviations of a maximum of around 9% were reported. The deviations were smaller for lower RPMs and velocities. The MRF method proved more viable as it was able to solve the governing equations for the static and rotating regions independently providing a higher degree of accuracy for the flow. The AMI method showed the best agreement compared to the other methods with a highest deviation of around 3%. The deviation remained constant even when the RPM and speed was increased owing to the solver's capability to model the flow accurately.

6.1.3 Computational Time

The computational time, in terms of clock-time and CPU time, taken to study the free-standing propeller showed the behaviour of the SRF and MRF methods to be alike. Very small differences in the order of 9 minutes was noticed between the two methodologies as seen in Table 21. On the other hand, the time taken to execute the AMI method was twice as much as the MRF method. This difference in time is mainly attributed to two reasons; firstly the AMI method used a transient solver which took a longer time to converge to a steady state solution. Secondly, the mesh at the area where the rotating region and stationary region meet had more

cells closely packed as this was of prime importance to allow for the rotation of the cylinder to take place.

6.1.4 Outtakes

The $k-\epsilon$ model did not perform adequately as the results in thrust and torque produced and velocity profiles showed a large deviation. The thrust and power values used as a comparison between the $k-\omega$ SST model had a nearly 10% difference. The use of the $k-\epsilon$ model is not advised for external aerodynamics and thus, it was not tested further in this study. The SRF method is considered time and cost effective and should be implemented for multi-parameter studies according to Wilhelm[18]. However, the differences in computational time taken to solve the SRF and MRF were very small in this case and hence, it would be beneficial to use the MRF method instead. The AMI method proved to have the highest degree of accuracy but was computationally very expensive and thus, should be used only in cases that require a high degree of accuracy.

6.2 Dji Phantom 3

6.2.1 Single rotor

The single rotor of the Dji Phantom 3 was tested for different RPMs at the hover condition with the three methods. At higher RPMs the deviation between the experimental and SRF method results grew larger. One reason for the deviation could be explained by the fact that the SRF method had convergence to a lower order as the RPM increased. The MRF and AMI methods on the other hand, performed much better than the SRF method with a maximum deviation of 8.5% and as these methods implement a much smaller rotating region, the forces calculated are more accurate.

The SRF and MRF tests took once again comparable CPU times whereas the AMI method took twice the time, see Table 22. As the SRF under-performed for both the FSP and the Single rotor case of the quadcopter taking the same computational time as the MRF method, it was not implemented to conduct further tests.

6.2.2 Simplified Dji Phantom 3

The MRF and AMI methods were compared for the simplified Dji Phantom 3 model for the total thrust and the rotor thrust. The MRF method had a 8% deviation for the total thrust and 6% for the rotor thrust when compared with numerical data from [16]. Mesh problems encountered during the setup of the AMI method led to stability issues which is why the AMI method did not work unfortunately. The MRF method took roughly 21 hours to simulate for the simplified model whereas if the AMI method worked, it would definitely take a longer time to converge to a stable solution as the mesh at the interface region of the rotating and stationary parts were of higher order. This would probably result in a higher accuracy for the results compared but as seen from previous cases, the MRF method gave good

results without a high cost to simulate. Different mesh approaches for the AMI method were tested but this unfortunately, did not lead to a stable solution and therefore, was considered a drawback for the current study. .

6.2.3 Comparison of different configurations

It was observed in Figure 33 that the rotors of the simplified Dji model produced more thrust than the complete Dji model. Rotor interaction reduced because of the presence of the battery and stands which lead to the higher thrust production for the rotors of the complete model. Since, the landing gear and battery were in the wake region, see Figure 34, they were contributing to the downforce which lead to a loss in total thrust. The total thrust for the simplified model showed only a small drop(about 1%) from the rotor thrust because of the presence of the airframe. Figure 35 also shows the wake from the simplified model not being obstructed in its path and thus no significant thrust drop was noticed.

The MRF method was able to capture the flow physics accurately and the results were seen consistent with findings from Yeon and Diaz[19]. The study also showed similar findings for the comparison of the complete and simplified models. Differences in values were observed owing to the fact that they used a high-fidelity mesh with around 350 million cells for unsteady flows whereas, the current study had a much smaller mesh size for a steady case.

6.2.4 Effects of variation in MRF Cylinder radius

The case with the original MRF cylinder radius enclosed the propeller perfectly, thus the wake propagated downwards with limited interactions with the main body of the quadcopter. As observed in Figure 38, contours of velocity can be seen on the propellers and the arms of the quadcopter. With an increase of 5% of the radius, the thrust increased significantly while the drag only increased by 1%, Figure 37. However, the vortices can be seen extending beyond the arms of the quadcopter to the body. The 10% increase in radius had the vortices covering the entire body of the quadcopter. This decreased the thrust, which was comparable to the original case but also increased the drag. As the flow slowed down on the body of the quadcopter and led to bigger re-circulation zones near the camera setup, the drag increased. Also, as the rotating region had a larger radius, the wake produced was stronger and propagated for a slightly bigger distance. This would create a problem for setups where the domain is extremely precise as a larger radius of rotation would result in a larger wake and thus, would require a bigger domain for the flow to settle down thereby, increasing the computational cost.

Based on these cases, it would be advised to enclose the rotating region with a 1% or 2% tolerance level for modeling MRF methods. Any increase in the area of the rotating region would complicate the flow physics further especially in cases with drones and wheels of cars as they are very compact and closely bound.

6.2.5 Effects of weather

The quadcopter was tested for hover conditions at different RPMs but to extend the capability of the method established, effects of 3 different wind velocities on the quadcopter were also tested. Velocity contours and the velocity profile on a z-plane(top view) were shown for the different wind speeds. The wake was seen to be pushed in the direction of the wind. Wakes produced for velocities of 17 and 20[kts] were much stronger and Figure 41 shows the interaction of the flow between the rotors to be much stronger compared to the hover condition. Figure 40 shows the thrust produced by each rotor. It was observed that the higher the wind speed, the larger the thrust produced by each rotor. Surface pressure on the body of the quadcopter showed low pressure regions on the top portion where the wake generated by the propeller was pushed towards the body for higher wind speeds.

The MRF methodology performed considerably well when it came to study the effects of weather conditions. Results such as the increase of rotor thrust with increase in wind speed and the propagation of the wake produced by the rotation of the propellers were seen to be consistent with data from Yeon and Diaz[19]. Experimental data comparisons if available would further, justify the accuracy of the methodology.

6.2.6 Effect of weather and pitch angle

Two pitch angles, 10° and 15° were tested for a wind speed of 20[kts]. Testing of the pitch angle was necessary to see if the MRF method was able to adapt to on-coming wind from different directions as a pitch angle can help determine forward flight characteristics. Figure 43 and Figure 44 show a significant part of the flow get deflected upwards from the rotor area, visible with the help of streamlines. The velocity profile on a z-plane shows extent of the wake and the case with the 15° pitch angle was seen to have a wider wake region.

No thrust results were available to compare the numerical setup with other data as this would indicate how accurate the MRF method was able to predict the flow. This test along with a test for differential rotor RPMs would narrow down on the performance of the accuracy of the MRF method. Weather and pitch angle testing is essential to also judge the control system movements of the quadcopter [19].

6.2.7 Overmount and undermount comparisons

The overmount configuration produced more thrust according to Table 24 and the difference in thrust was around 2% compared to the undermount configuration. The downwash generated from the overmount configuration, Figure 45 was similar to the low pressure generated for the undermount configuration, Figure 46 which is why the fuselage downforce for the two are nearly same in Table 24. Rotor thrust alone had around a 2% difference between the two configurations. The study of the undermount configuration is useful in case where rotors need to be placed above and below the motors [16].

The MRF method predicted the performance of both the configurations pretty accurately with a maximum deviation of around 15%. The overall wake profile and thrust results were seen consistent with findings from [16].

6.2.8 Outtakes

Three methods were tested for different scenario simulations on the Dji Phantom 3. Firstly, a single rotor of the Dji Phantom 3 was tested for different RPMs. All three methods performed considerably well with the SRF method showing the highest deviation in results. The SRF and MRF methods took almost the same time to perform the simulations but the accuracy of the MRF method was much better. Owing to this and previous simulation comparisons, the SRF method was dropped from the methods to compare.

Further, the MRF and AMI methods were tested on the simplified model of the Dji Phantom 3. Rotor thrust, total thrust and wake profiles were compared with numerical data from [16]. The AMI method was dropped due to complications arising from the mesh. It would be advised to use the AMI method in cases only where the accuracy of the results needed plays a more vital role than the cost required to simulate it. Additionally, as AirShaper's current platform does not cater to unsteady simulations, the AMI method was dropped henceforth and more focus was given to the MRF method. In the future, when the platform implements unsteady simulations and clients require accuracy, the AMI method can be built into the platform's code. Furthermore, the point of using the AMI method was to quantify the potential accuracy losses that arise from the implementation of simplified models such as MRF in the platform.

With the MRF method decided as the best method to implement into the platform, 4 scenario simulations were carried out and the capability of the method was established. Different scenarios helped predict how the method behaved and if it was able to capture the flow physics accurately. Results ranging from how much bigger should the rotation region for the method be to weather and pitch angle scenarios were shown and discussed. The MRF method was seen to predict the flow physics pretty accurately being a steady flow method and it was seen that implementing the method into the platform could help analyze rotating flows.

7 Future Work

The following list contains the scope of the future work that can be carried out:

1. OpenFOAM has a limitation with the MRF method as it only operates with a steady state solution. If the development of an unsteady solver is incorporated for the MRF method in OpenFOAM, the results could provide a better degree of accuracy.
2. As the MRF method was established for propellers of a quadcopter, it could be useful to check if the method predicted data accurately for other cases like rotation of wheels of a car. Results from these should be checked before implementing the code into the platform to make sure that the method works for all cases.
3. Another limitation was that the source code for the AMI method did not allow for multiple rotating flows to take place and modification would be required to enable multiple rotation. The modification of the source code to allow for multiple rotation would be extremely insightful and it could provide valuable results like the rotor-rotor interaction for the AMI method. Although very expensive to simulate, the results could help make modifications in the initial design stages leading to a more aerodynamic model.
4. The final step would be to implement the MRF code onto the platform and allow the client to setup with the click of a button. Additionally, when Air-Shaper uses transient solvers, the AMI method should also be implemented and allowed as an option for clients seeking higher accuracy, but they should be warned about the time taken to simulate the case.

8 Conclusions

OpenFOAM was used to study three methods that allowed rotating flows; the SRF, MRF and AMI methods. The three methods were applied first for a free-standing propeller and then for propellers of the Dji Phantom 3. Mesh independence studies were done for the three methods and for the different turbulence models as well. Experimental comparisons with the free-standing propeller were made using performance data available from the manufacturer, APC propellers. It was observed that the SRF method under-performed compared to the MRF while taking more or less the same computational time. Two turbulence models using the MRF method were compared, and the $k-\epsilon$ model was judged not suitable to continue with. The AMI method preformed most accurately but took twice the time the MRF method took.

With the rotating methods validated and the $k-\epsilon$ model removed from further tests, the three approaches were tested for the propellers of the Dji Phantom 3. The single-propeller was tested first and again it was noticed that the SRF method under-performed compared to the MRF taking the same computational time and hence, the SRF method was dropped and not implemented for further tests. Then a simplified model of the Dji Phantom 3 without the camera and landing stands was tested with the MRF and AMI method and compared with numerical data from NASA's research. A lot of complications were noticed while setting up the simulations as the mesh for the AMI had to be extremely refined for the method to work. Even with a refined mesh, the AMI method ran into stability issues and was dropped at this stage owing to the large amounts of time taken to setup the method. A large amount of setup time apart from the computational time needed to solve the AMI method would be needed to perform other tests that were planned for the duration of this thesis. It was because of this reason that the AMI method was dropped and the MRF method was selected to perform the other tests.

Finally, the MRF method was tested for different scenario simulations and compared with numerical data available. The method was tested for weather conditions, pitch angle conditions and undermount configuration conditions for the entire Dji Phantom 3 model. It was seen that the MRF was able to predict the flow physics quite accurately and the performance was very good for a steady-state solver and this was the justification to rate it as the best method among the 3 methods tested.

In terms of accuracy alone, the AMI method outperformed against the other two methods and results with good agreement with experimental and numerical simulations were noticed. However, the accuracy came at a cost which was computational time to perform the simulations and setup time needed for the mesh and this proved to be too expensive. For a company whose marketing tagline was to provide fast results, the use of the AMI method would not permit this and hence, the MRF method was judged the best method as it provided fairly accurate results without being too computationally heavy. Recommendations for the platform to provide the MRF method as the default simulation setup should be implemented and the AMI method should be offered in case the client prefers accuracy over cost.

References

- [1] Henningson Dan S , Berggren Martin. Fluid Dynamics:Theory and Computation; 2005. https://www.mech.kth.se/~henning/CFD/CFD_main.pdf.
- [2] Menter FR. Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications. 1994;.
- [3] Online C. SST k-omega model; 2011. https://www.cfd-online.com/Wiki/SST_k-omega_model.
- [4] AutoDesk. SST k-omega turbulence model; 2015. <https://knowledge.autodesk.com/support/cfd/learn-explore/caas/CloudHelp/cloudhelp/2014/ENU/SimCFD/files/GUID-0F5C4828-9F91-46B6-A16A-2578D72DCFCC-htm.html>.
- [5] Guide CT. Shear stress transport model;. https://www.sharcnet.ca/Software/Ansys/16.2.3/en-us/help/cfx_thry/i1302321.html#i1302649.
- [6] Bakker A. Introduction to Computational Fluid Dynamics. 2008;.
- [7] Online C. SIMPLE Algorithm; 2016. https://www.cfd-online.com/Wiki/SIMPLE_algorithm.
- [8] Olle P. CFD with Open source software; 2011. http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2011/OlofPenttinen/projectReport.pdf.
- [9] Fumiya N. CFD for rotating machinery; 2015. <https://www.slideshare.net/fumiyanozaki96/cfd-for-rotating-machinery-using-openfoam>.
- [10] T Maric, J Høpken, K Mooney. The OpenFOAM technology primer. SorceFlux GmbH; 2014.
- [11] Magnus W. Benchmark and validation of OpenSource CFD codes, with focus on compressible and rotating capabilities, for integration on the SimScale platform; 20153. <http://publications.lib.chalmers.se/records/fulltext/198965/198965.pdf>.
- [12] OpenFOAM Foundation. OpenFOAM v7 User Guide;.
- [13] Ayachit U. The ParaView Guide: A Parallel Visualization Application. KitWare USA; 2015.
- [14] Propellers A. APC Propellers- Technical Information.;. <https://www.apcprop.com/technical-information/>.
- [15] Performance Data;. <https://www.apcprop.com/technical-information/performance-data/>.
- [16] Diaz V Patricia, Yoon Seokkwan, C M William . Computational Aerodynamic Modeling of Small Quadcopter Vehicles. AHS Forum 73; 2017.

- [17] WMO. The Beaufort Scale of Wind Force: (Technical and Operational Aspects). 1970;.
- [18] D W. Rotating Flow Simulations with OpenFOAM. International Journal of Aeronautical Science Aerospace Research (IJASAR); 2015.
- [19] Diaz V Patricia, Yoon Seokkwan . High-Fidelity Computational Aerodynamics of Multi-Rotor Unmanned Aerial Vehicles. AIAA SciTech Forum 2018, Kissimmee, Florida; 2018.

A OpenFOAM Solvers and Script

A.1 OpenFOAM ./Allrun script

The ./Allrun script allows to run commands directly with the help of a script and can be used by the command *./Allrun* in the terminal window. The following commands were generally included in the ./Allrun script and the script shown below is to execute the AMI method:

```
blockMesh
surfaceFeatureExtract
decomposePar
mpirun -np 32 snappyHexMesh -overwrite
reconstructParMesh -constant
cp -rf 0.orig 0
createPatch -overwrite
checkMesh
decomposePar
mpirun -n 32 renumberMesh -overwrite -parallel
pyFoamPlotRunner.py mpirun -np 32 pimpleDyMFoam -parallel
reconstructPar
paraFoam
```

A.1.1 fvSchemes Dictionary

The finite volume schemes used by the $k-\omega$ SST model for the MRF method are shown below:

```
ddtSchemes
{
    default steadyState;
}
gradSchemes
{
    default Gauss linear;
    grad(U) cellLimited Gauss linear 1;
}
divSchemes
{
    default none;
    div(div(phi,U)) Gauss linear;
    div(phi,U) bounded Gauss linearUpwindV grad(U);
    div(phi,k) bounded Gauss upwind;
    div(phi,omega) bounded Gauss upwind;
    div(nuEff*dev2(T(grad(U)))) Gauss linear;
    div(R) Gauss linear;
```

```

laplacianSchemes
{
    default Gauss linear corrected;
    laplacian(yPsi) Gauss linear corrected;
}
interpolationSchemes
{
    default linear;
}
snGradSchemes
{
    default corrected;
    grad(U) corrected;
}
wallDist
{
    method meshwave;
}

```

A.1.2 fvSolution Dictionary

The finite volume solutions used by the $k-\omega$ SST model for the MRF method are shown below:

```

solvers
{
    p
    {
        solver GAMG;
        tolerance 1e-9;
        relTol 0.0001;
        smoother GaussSeidel;
        nPreSweeps 0;
        nPostSweeps 2;
        cacheAgglomeration on;
        agglomerator faceAreaPair;
        nCellsInCoarsetLevel 10;
        mergeLevels 1;
    }
}
SIMPLE
{
    nNonOrthogonalCorrectors 1;
    residualControl;
    p 1e-9; U 1e-9; k 1e-9;
}
relaxationFactors

```

```
{  
  equations;  
  U 0.9; k 0.7; omega 0.7;  
}
```

It is to be noted that solvers for k , ω , U , y_{Psi} and Φ were assigned the same way as the solver for p was assigned.

B Meshing in OpenFOAM

B.1 blockMesh

Screenshots of the *blockMesh* file used to create the domain for the Dji Phantom 3 are shown below.

```
1  /*-----* C++ */
2  |
3  | \ \ / / F i e l d   | OpenFOAM: The Open Source CFD Toolbox
4  | \ \ / / O p e r a t i o n   | Version: 2.3.1
5  | \ \ / / A n d   | Web: www.OpenFOAM.org
6  | \ \ / / M a n i p u l a t i o n   |
7  |
8  FoamFile
9  {
10     version      2.0;
11     format        ascii;
12     class          dictionary;
13     object          blockMeshDict;
14 }
15
16 // *****
17
18 convertToMeters 1;
19
20 vertices
21 (
22
23     (-0.865 -1.55 -1.23375) //0
24     ( 3.465 -1.55 -1.23375) //1
25     ( 3.465 1.55 -1.23375) //2
26     (-0.865 1.55 -1.23375) //3
27     (-0.865 -1.55 0.95175) //4
28     ( 3.465 -1.55 0.95175) //5
29     ( 3.465 1.55 0.95175) //6
30     (-0.865 1.55 0.95175) //7
31 );
32
33
34 blocks
35 (
36     hex (0 1 2 3 4 5 6 7) (210 168 140) simpleGrading (1 1 1)
37 );
38
39 edges
40 (
41 );
42
43 boundary
44 (
45     left
46     {
47         type patch;
48         faces
49         (
50             (3 7 6 2)
51         );
52     }
53
54     right
55     {
56         type patch;
57         faces
58         (
59             (1 5 4 0)
60         );
61     }
62
63     bottom
64     {
65         type patch;
66         faces
67         (
68             (0 3 2 1)
69         );
70     }
71
72     top
73     {
74         type patch;
75         faces
76         (
77             (4 5 6 7)
78         );
79     }
80
81     inlet
82     {
83         type patch;
84         faces
85         (
86             (0 4 7 3)
87         );
88     }
89
90     outlet
91     {
92         type patch;
93         faces
94         (
95             (2 6 5 1)
96         );
97     };
98
99 mergePatchPairs
100 (
101 );
102
103 // *****
```

Figure 47: *blockMesh* dictionary used to create the domain for the Dji Phantom 3.

B.1.1 snappyHexMesh

Screenshots of the *snappyHexMesh* file used for the *Dji Phantom 3* are shown below.

```
1  /*-----* C++ *-----*/
2  {
3      // F ield
4      // O peration
5      // A nd
6      // M anipulation
7  }
8
9  FoamFile
10 {
11     version    2.0;
12     format      ascii;
13     class      dictionary;
14     object      snappyHexMeshDict;
15 }
16 // *****
17
18 // Which of the steps to run
19 castellatedMesh true; // castellatedMeshTriger
20 snap true; // snapTriger
21 addLayers false; // addLayersTriger
22
23 geometry
24 {
25     object.obj
26     {
27         //scale 1;
28         type triSurfaceMesh;
29         name object;
30     }
31     refinementBoxOuter
32     {
33         type searchableBox;
34         min (-0.5 -0.5 -0.6);
35         max ( 1 0.5 0.6);
36     }
37     refinementBoxInner
38     {
39         type searchableBox;
40         min (-0.3 -0.25 -0.25);
41         max ( 0.3 0.25 0.25);
42     }
43 }
44
45
46 // Settings for the castellatedMesh generation.
47 castellatedMeshControls
48 {
49     maxLocalCells 10000000;
50     maxGlobalCells 200000000; // snapMaxGlobalCellsTriger
51     minRefinementCells 10;
52 }
```

```
56     maxLoadUnbalance 0.10;
57     nCellsBetweenLevels 2;
58     features
59     {
60         {
61             file "object.eMesh";
62             level 5; // snapSurfaceLevelExpTriger
63         }
64     };
65
66 // Surface based refinement
67 refinementSurfaces
68 {
69     object
70     {
71         level (4 5); // snapSurfaceLevelTriger
72         patchInfo
73         {
74             type wall;
75         }
76     }
77 }
78
79 // Resolve sharp angles
80 resolveFeatureAngle 50;
81
82 refinementRegions
83 {
84     refinementBoxOuter
85     {
86         mode inside;
87         levels ((1E15 2)); // refRegionsStartTriger
88     }
89     refinementBoxInner
90     {
91         mode inside;
92         levels ((1E15 3)); // refRegionsEndTriger
93     }
94 }
95
96 // Mesh selection
97 // -----
98 locationInMesh (2 0 0);
99 allowFreeStandingZoneFaces true;
```

```

113 // Settings for the snapping.
114 snapControls
115 {
116     nSmoothPatch 3;
117     tolerance 2.0;
118     nSolveIter 30;
119     nRelaxIter 5;
120     // Feature snapping
121     nFeatureSnapIter 10;
122     implicitFeatureSnap false;
123     explicitFeatureSnap true;
124     multiRegionFeatureSnap false;
125 }
126
127 addLayersControls
128 {
129     relativeSizes true;
130     layers
131     {
132         "(lowerWall|object).*"
133         {
134             nSurfaceLayers 5;
135         }
136     }
137     expansionRatio 1.1;
138     finalLayerThickness 0.2;
139     minThickness 0.005;
140     nGrow 0;
141     // Advanced settings
142     featureAngle 60;
143     slipFeatureAngle 30;
144     nRelaxIter 3;
145     nSmoothSurfaceNormals 1;
146
147     nSmoothNormals 5;
148     nSmoothThickness 10;
149     maxFaceThicknessRatio 0.5;
150     maxThicknessToMedialRatio 0.3;
151     minMedianAxisAngle 90;
152     nBufferCellsNoExtrude 0;
153     nLayerIter 50;
154 }
155
156 // Generic mesh quality settings. At any undoable phase these determine
157 // where to undo.
158 meshQualityControls
159 {
160     #include "meshQualityDict"
161     nSmoothScale 4;
162     errorReduction 0.75;
163 }
164
165 // Advanced
166 // Write flags
167 writeFlags
168 {
169     scalarLevels
170     layerSets
171     layerFields // write volScalarField for layer coverage
172 };
173
174 mergeTolerance 1e-6;
175
176 // *****

```

Figure 48: *snappyHexMesh* dictionary used for the Dji Phantom 3.