CFD Simulations and Physics Engine Enchancing for Quadrotors

Documentation of CFD simulations and the implementation of the flow for MuJoCo physics engine.

Author:
Ádám Weinhardt-Kovács

Developers:
Ádám Weinhardt-Kovács
email: weini@sztaki.hu or wkadam9@gmail.com
Máté Koroknai
email: koroknai.mate@sztaki.hu or mate.koroknai93@gmail.com

Version: **1.0**
Date: 2023.08.03.

AIMotionLab SZTAKI
Számítástechnikai és Automalizálási Kutatóintézet
Institute for Computer Science and Control
Budapest, Hungary

Contents

1 Motivation

Our goal is to implement the effect of the flow to our MuJoCo physics environment. In this way we can achieve a more realistic model and further analyse payload transportation projects. During payload transportation we use our Bumblebee quadrotor at AIMotionLab and we have the twin model implemented in our MuJoCo environment. We observed instability and vibrations during real world transportation, even hovering with the Bumblebee, thus a controller where the effect of the flow plays a role would be useful.

Figure 1: Bumblebee quadrotor

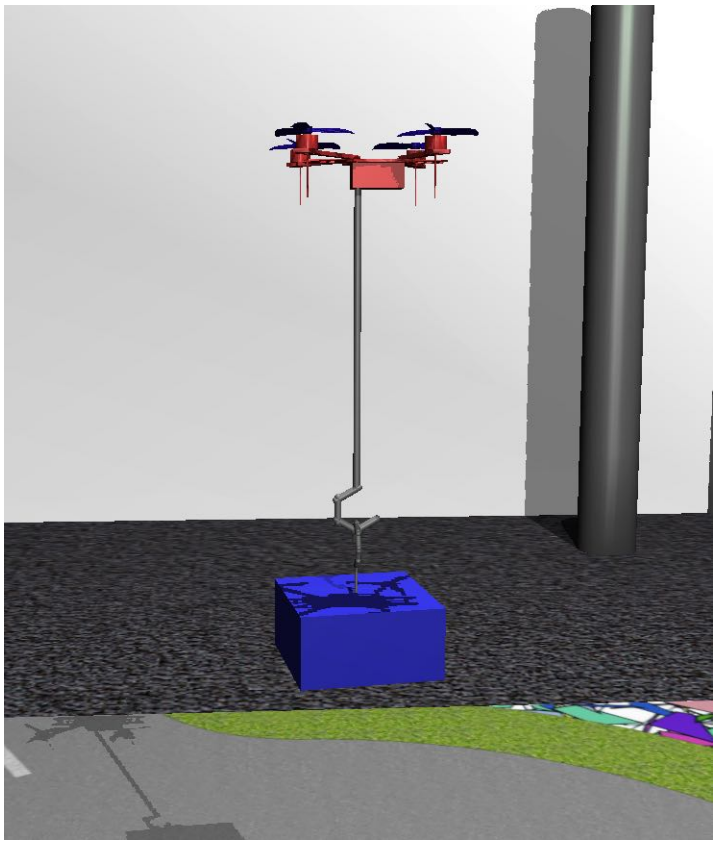

Figure 2: Bumblebee quadrotor in MuJoCo environment

2 Introduction to CFD

Computational Fluid Dynamics (CFD) is a finite volume method which can simulate fluid dynamics numerically. CFD relies on the principles of conservation of mass, momentum, and energy, which are expressed as partial differential equations (PDEs) known as the Navier-Stokes equations. These equations describe the motion of fluids and are notoriously difficult to solve analytically. CFD methods enable the approximation and solution of these equations through discretization and numerical techniques. More in: [1]. Downloadable english version: *Numerical modeling of fluid flows-Dr. Gergely Kristóf, Department of Fluid Mechanics, BME*

2.1 Workflow of CFD

During the modeling of flows the following steps are necessary (the used softwares are in the brackets). The ANSYS workbench is a project environment where the user can complete the whole simulation, but in our case only the geometry, the meshing and the visualization was made in ANSYS workbench, the simulation was made in Fluent separately (the simulation in the workbench environment is slower and issues can occur).

1. Geometrical modeling (ANSYS workbench-Space Claim software)
2. Meshing (ANSYS workbench-ANSYS mesher)
3. Problem setup + Solution (Fluent)
   (a) Boundary Conditions
   (b) Physical model specification
   (c) Specification of material properties
   (d) Choosing solver
   (e) Initialization
   (f) Iteration till convergence
4. Visualization (ANSYS workbench)

3 Single rotor model (cylindrical)

To access the source files to this version go to: SZTAKI Nextcloud: CFD/single rotor cylindrical.

Since the modelling of the full quadcopter is very expensive and we have licence limitations, one of the rotors of the quadcopter is simulated. In this first single rotor version the domain is cylindrical, because with a Sliding Mesh model where the rotating fluid is a cylinder, a comfortable option to set the whole domain as a cylinder. Furthermore the motivation was in this version is to see make the process work and achieve realistic results. Some help can be found in: *YouTube: Playlist of useful videos for CFD simulations*.

3.1 Geometry (Space claim)

We can import our clock wise rotor geometry (.stl format) into SpaceClaim.

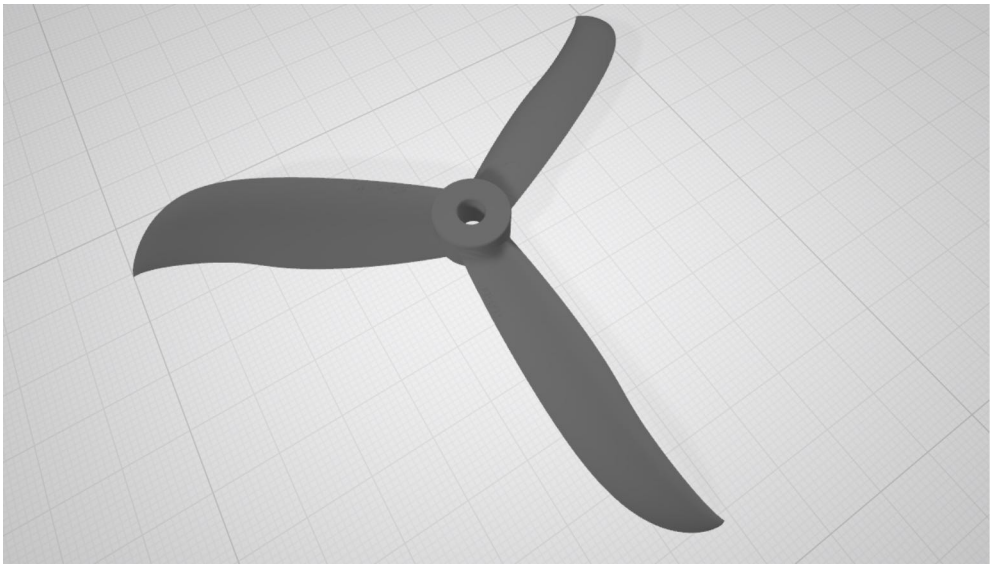

Figure 3: CW rotor

The geometry will conclude 2 areas: the fluid field and the rotating fluid field. The method that we will use to rotate the propeller is called Sliding Mesh model. In this model we can assign an angular velocity to a mesh area. Thus in the geometry generation the area of the rotor is extruded from the rotating cylinder. The diameter of the rotating fluid cylinder is $140mm$ and the height is $16mm$.

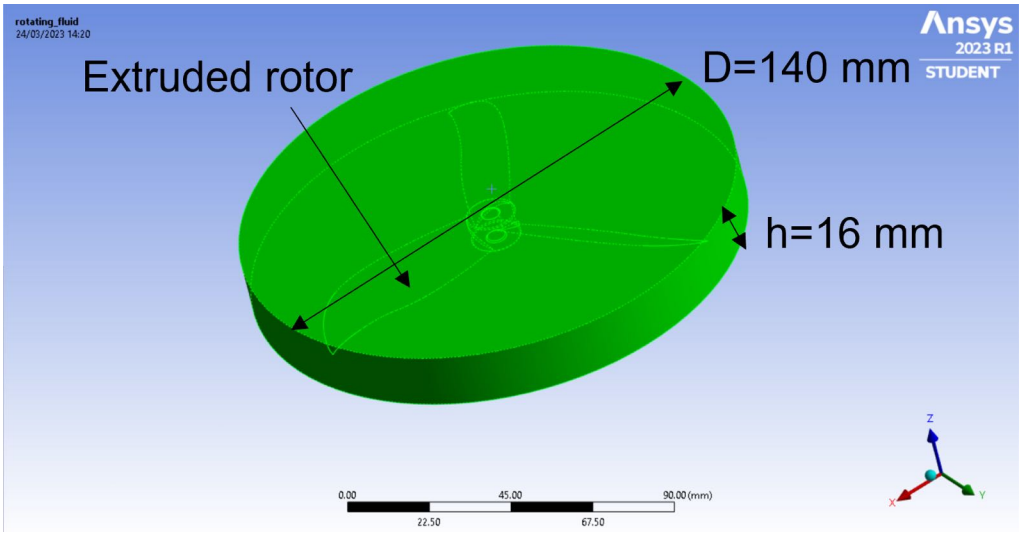

Figure 4: Rotating fluid cylinder

During the solution a *wall* boundary condition will be assigned to the surface of the extruded rotor. The other area is the fluid field with a $0.5m$ diameter and also the bottom is $0.5m$ far from the rotor and the top is $0.25m$ above the rotor.

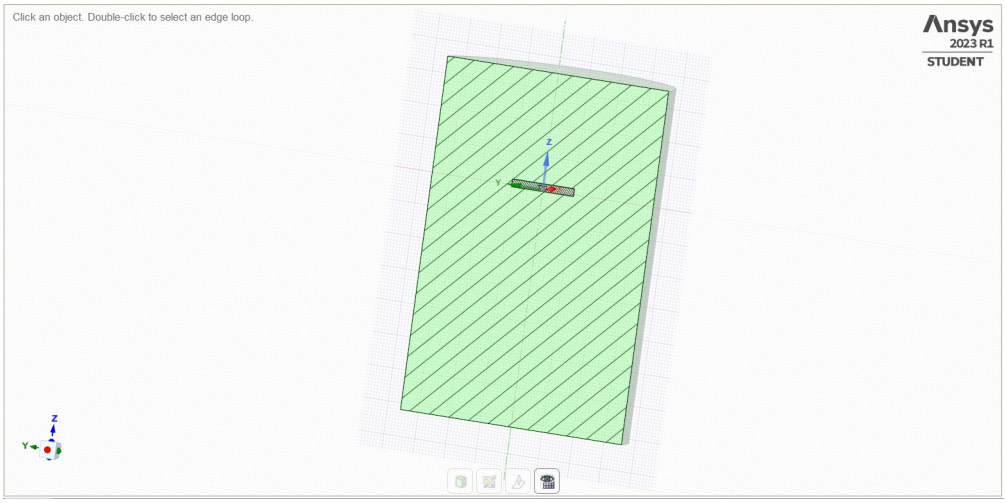

Figure 5: Cross section of the geometry

3.2 Mesh and boundary conditions (ANSYS mesher)

After the geometry creation we can mesh the domain. The meshing will divide the domain in to elements where the numeric solver can calculate the field variables.

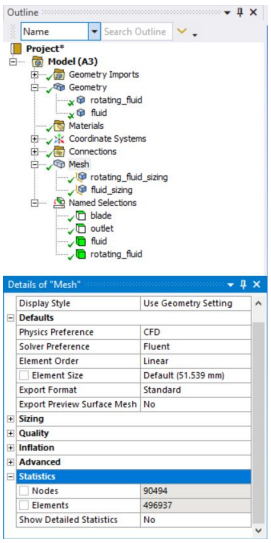

Figure 6: Mesh details

CFD is a finite volume method thus the domain is divided into little volume elements. CFD only stores one value at the middle of the cell and the node values are extrapolated from them. During the meshing we can assign different criteria to different domains and surfaces. Rule of thumb is to refine the mesh where we have big gradients, i.e. where the geometry changes rapidly. I assigned two different volume mesh sizing to the fluid and the rotating fluid domain. The fluid field was assigned with the element size of $15mm$ and the rotating fluid was assigned with a $4mm$ element size value. These numbers were arbitrary with the consideration that the rotating fluid domain should have the finer mesh and the whole meshing size shall be under $512,000$ element number for the limitation of the student license. Also in further development the cylindrical mesh should be refined homogeneously to achieve better results.

To validate the mesh we can use different metrics. The most common one is called skewness (Based on the internal angles of an element: for regular polygon elements, 0, if one internal angle is much smaller than the others, skewness approaches 1. A perfectly shaped element would have a skewness value of 0, indicating no distortion or deformation. On the other hand, a highly skewed element would have a skewness value close to 1, indicating significant distortion or irregularity.). A rule of thumb regarding the criteria is the skewness should not exceed 0.85 for hexa elements and 0.95 for tetra elements.

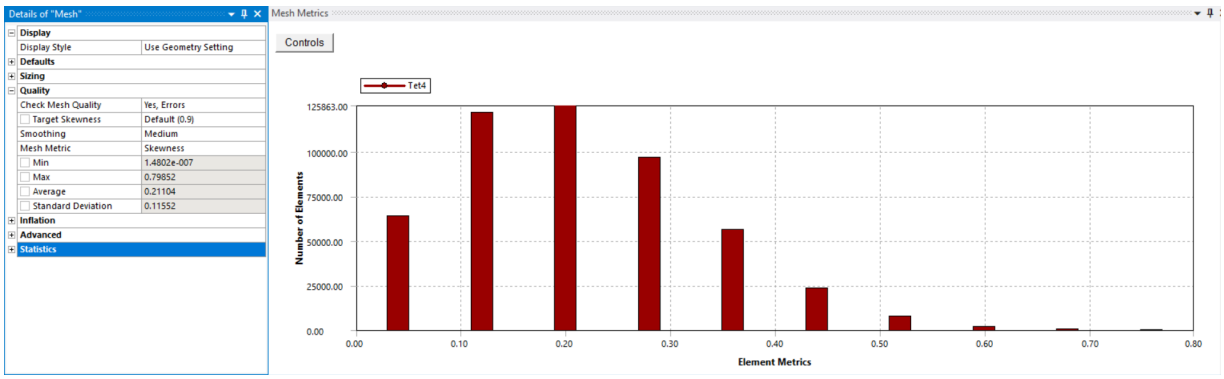

Figure 7: Skewness of the mesh

As we can see the maximum skewness in the domain is 0.7985 thus our mesh is fine enough. We can also depicts the skewness in the domain and we can show that the rotating disk cylinder has a finer mesh than the fluid field.

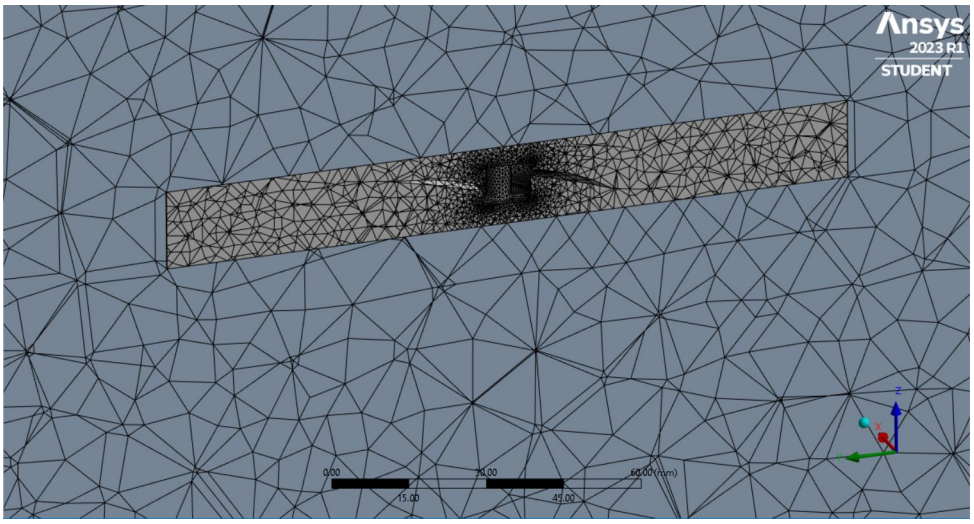

Figure 8: Mesh cross section

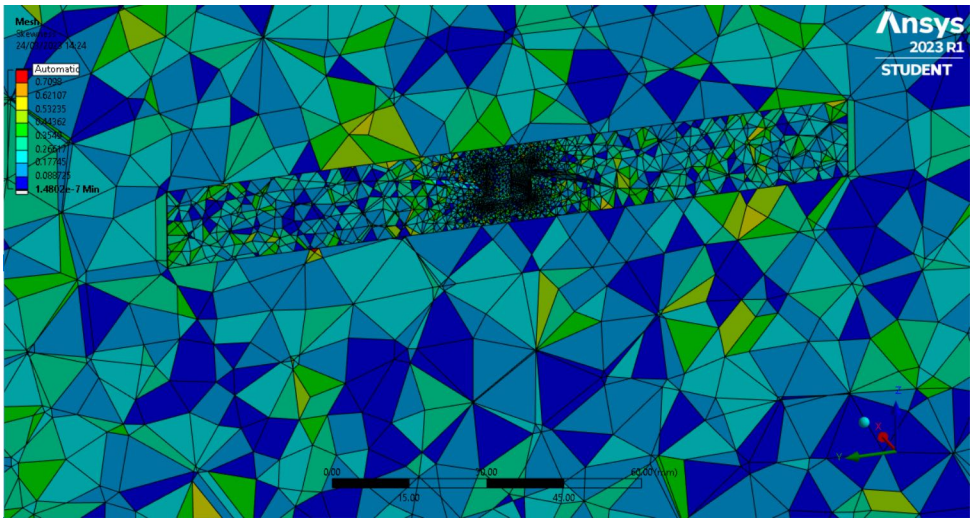

Figure 9: Mesh cross section colored with skewness

The other thing that we can do during the meshing, is assign name selection to domains. As we can see in

Figure 6 the *fluid* and *rotating fluid* names were assigned to the two domains. Moreover an *outlet* is assigned to the surface of the fluid cylinder and a *blade* is the the surface of the inner rotating fluid cylinder. The advantage of naming the domains is that when we import the mesh file into Fluent the software will detect the names selections and assign a proper boundary condition to them. In our case the two geometry domain will be detected as a fluid domain, the surface of the big cylinder as a far field pressure outlet and the surface of the rotor as a wall boundary condition.

3.3 Solution (Fluent)

After the meshing we can import our mesh to Fluent where the numerical solution is made. The limitation of the student licence is that we can only use 4 computer core for parallel computing which slows down the computation significantly. We can also start our simulation with an option called *double-precision* which make the solver store twice as much decimal number which correct the rounding error of the solver.

The following things were assigned in fluent for the simulation, with only the consideration that these options were choosed in the CFD/quadrotor literature [5], [2], [4]:

1. A pressure-based solver with gravity and transient time.
2. The viscous model is a $k-\epsilon$ realizable model with enhanced wall treatment.
3. The fluid was assigned to be air with a constant density.
4. An $1200\frac{rad}{s}$ angular velocity was assigned to the rotating fluid disk. (Approximately the operational area for hovering the drone)
5. Coupled solver with second-order spatial discretization.
6. The solver was initialized via. hybrid initialization.
7. The field variables were saved in every 15 time-step for post-processing and the thrust force was monitored during the iterations.
8. The iterations was started with a $0.0001s$ time step size and with a number of 2000 time steps overall.

During the iterative numerical solver the residuals are monitored automatically by Fluent.

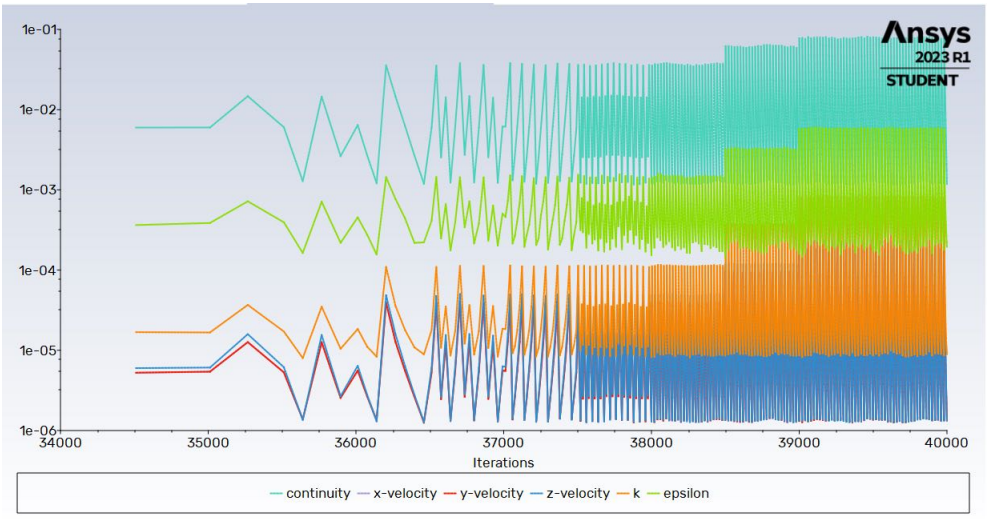

Figure 10: Residuals

Next to the residuals, it is important to monitor some value which represent the physical behaviour of the system. In our case the thrust force was monitored.

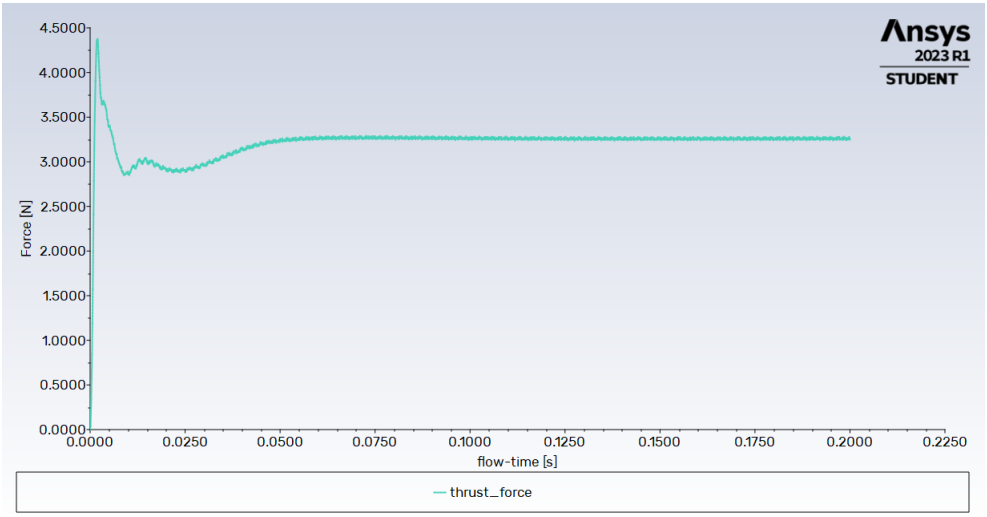

Figure 11: Thrust force of the rotor

The overall thrust force converged for approximately $3.25N$ which is in the domain of the measured value (further validation can be done according the documentation of the bumblebee).

3.4 Visualization (ANSYS workbench)

In the post-processing section we can visualize the field variables. The most common ones are the velocity and pressure contours/streamlines. First the values for the rotor:

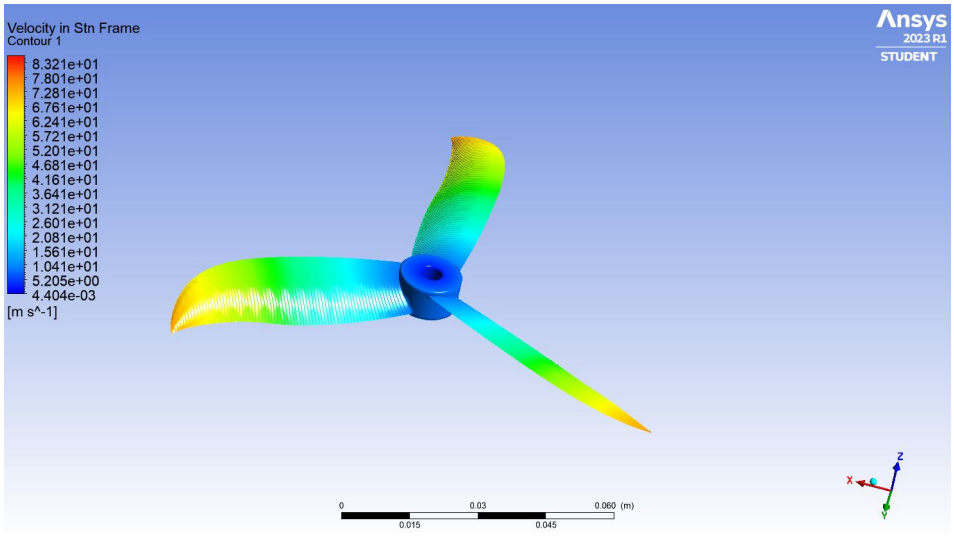

Figure 12: Velocity contour of the rotor

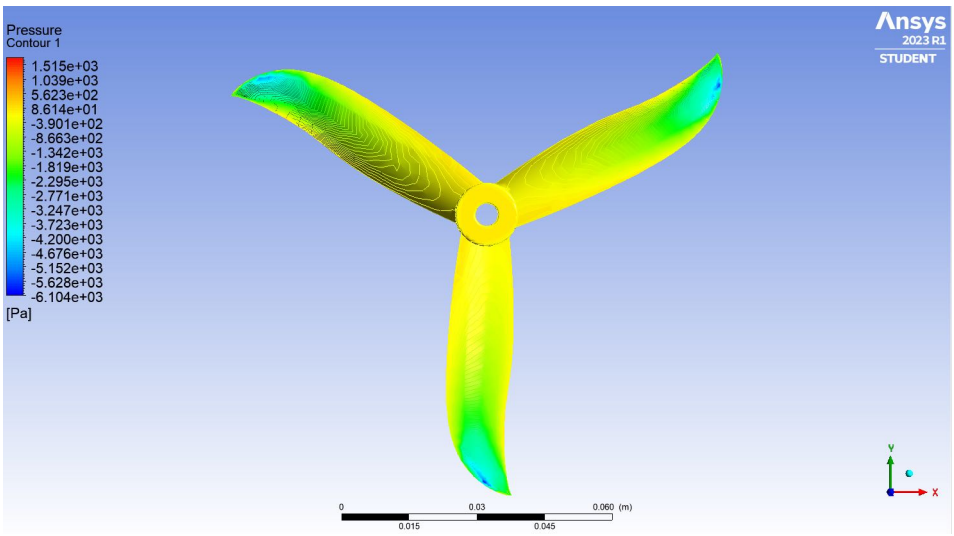

Figure 13: Pressure contour of the rotor (upper view)

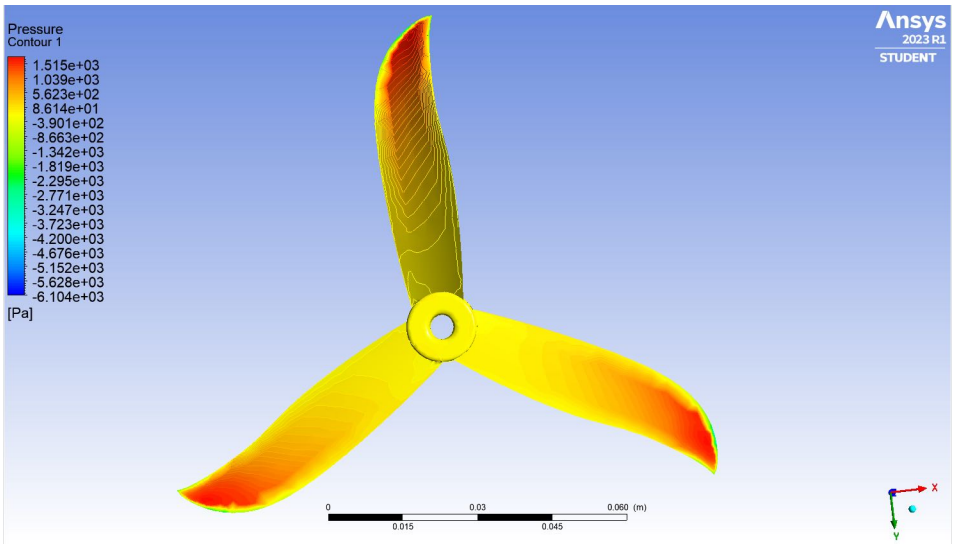

Figure 14: Pressure contour of the rotor (bottom view)

As we can see from the figures the biggest velocity occurs at the end of the blades and the pressure is bigger at the bottom of the blade which validates real life rotor behaviour. The other post-processing videos can be find at the Sztaki cloud.

1. *Full video: Velocity contour at a cross section*

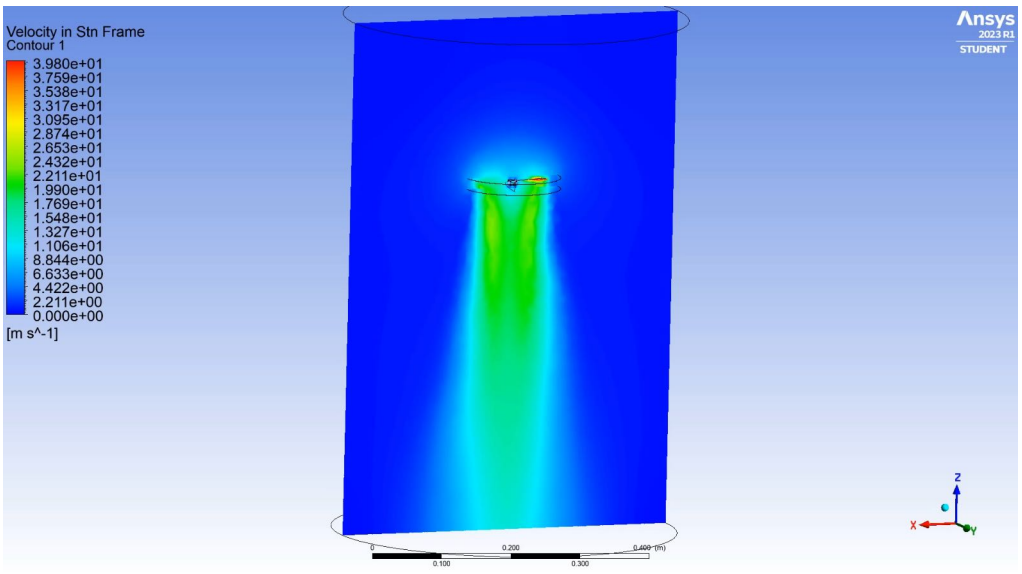

Figure 15: Velocity contour at a cross section

2. *Full video: Pressure contour at a cross section*

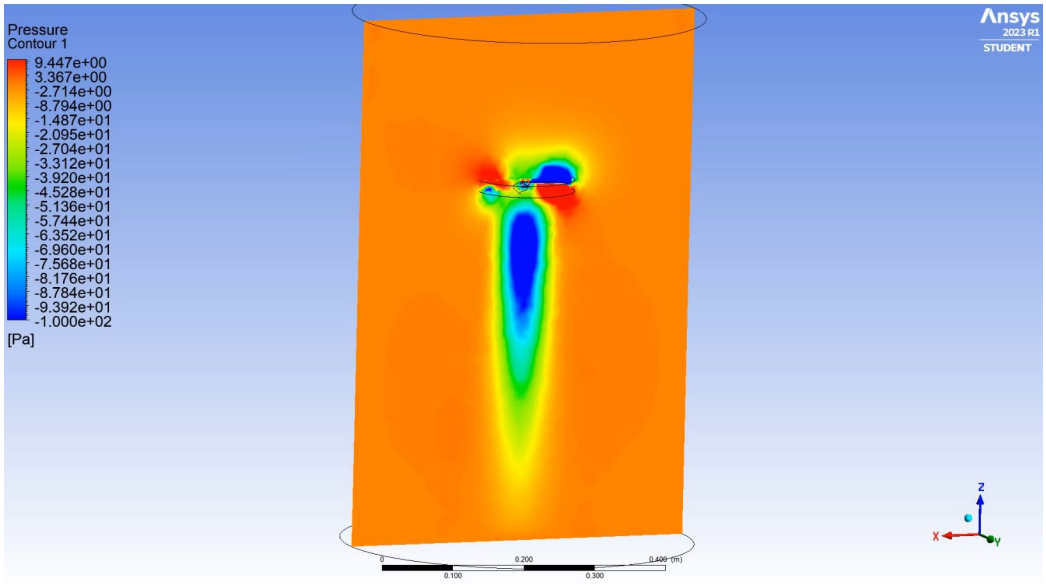

Figure 16: Pressure contour at a cross section

3. *Full video: Velocity streamlines in the whole field*

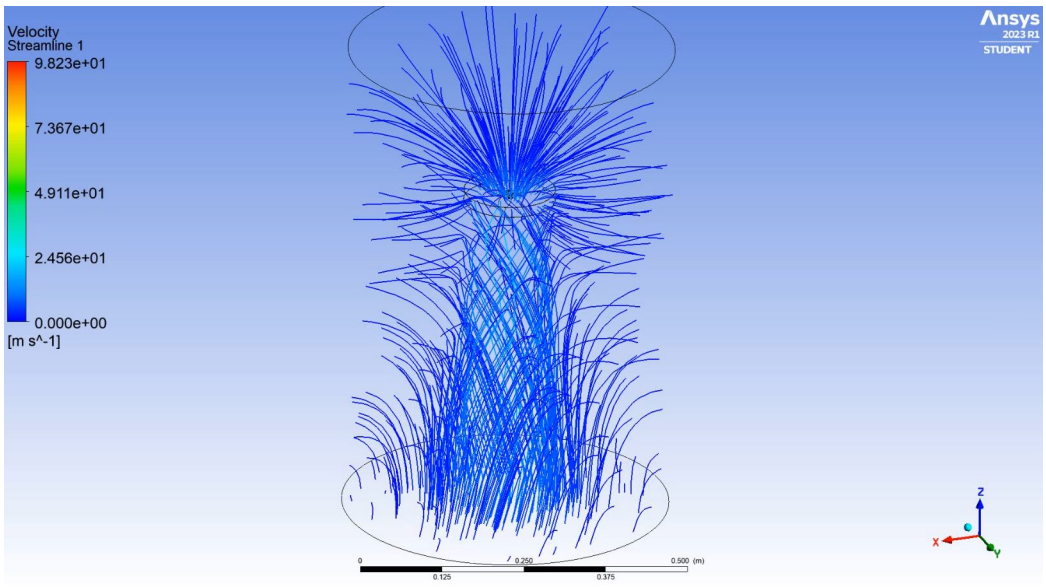

Figure 17: Velocity streamlines in the whole field

4. *F*ull video: Velocity streamlines from the rotating fluid domain

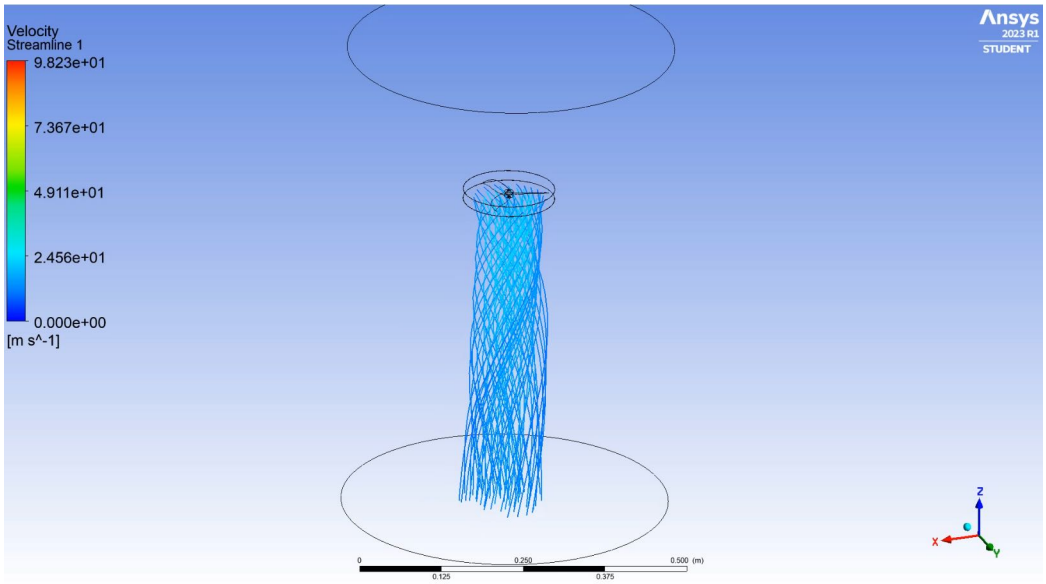

Figure 18: Velocity streamlines from the rotating fluid domain

4 Single rotor model (rectangular)

To access the source files to this version go to: SZTAKI Nextcloud: CFD/single rotor rectangular.

Our goal is to implement the flow of the rotor to the MuJoCo physics engine. In the previous section we made a simple simulation with a cylindrical fluid domain. For implementation reasons in this version the fluid domain will be rectangular. In this section only the geometry and the mesh will be different, the CFD solution options (see Section 3.3) will be the same as in the previous section.

4.1 Geometry (Space claim)

The geometry of the rotor is the same. The flow domain:

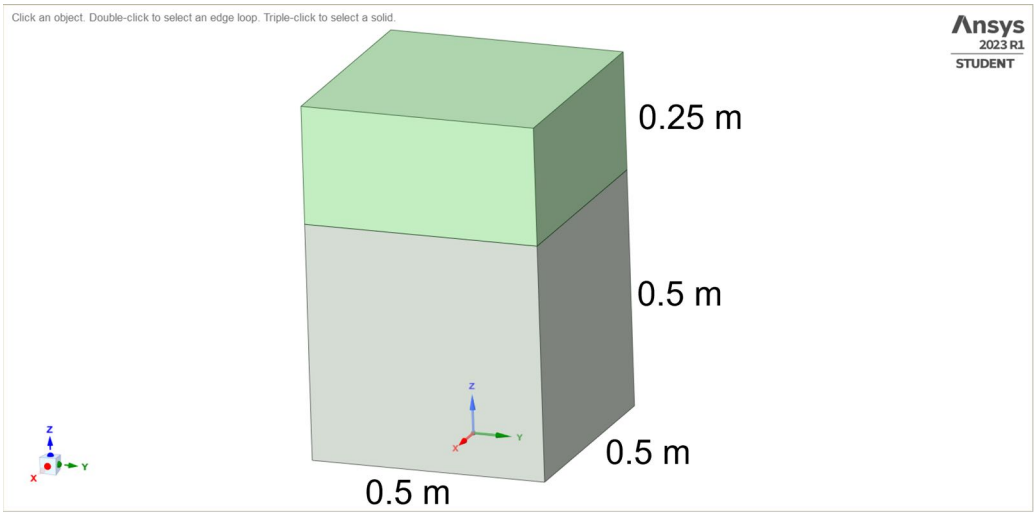

Figure 19: Geometry

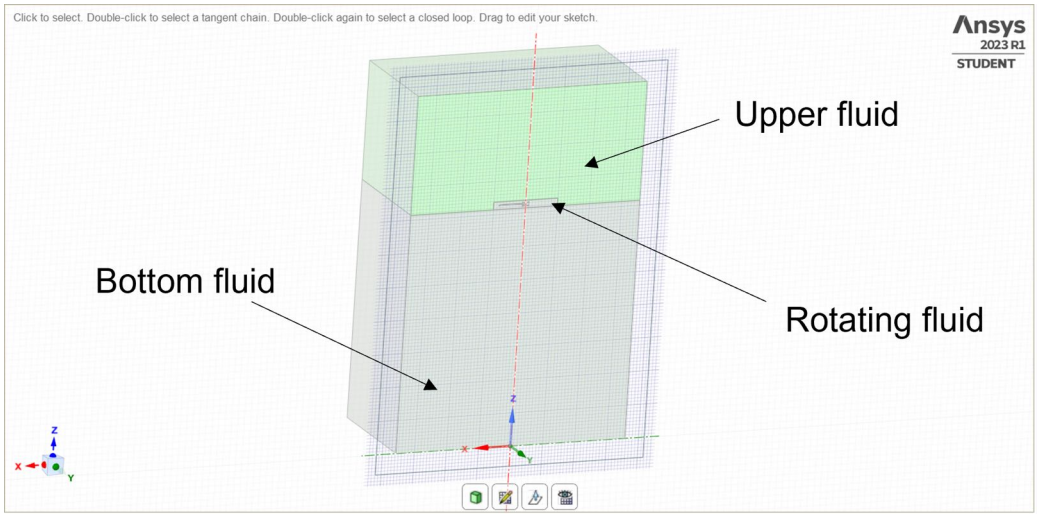

Figure 20: Geometry cross section

As we can see the fluid domain is rectangular. There is an upper fluid body and the bottom fluid flow domain. Our main interest is to export the dynamic pressure values under the rotor, thus it's a practical solution to separate the flow under the rotor and use a homogeneous mesh in that domain. In this way the pressure exportation will be simple and we can avoid further interpolation in the domain. The fluid flow domain is a cube with $0.5m$ edge size and the upper fluid is a $0.5m$ by $0.5m$ by $0.25m$ rectangle.

4.2 Mesh and boundary conditions (ANSYS mesher)

The domain construction is similar to the previous version. The difference is the separated fluid domains. The boundary conditions can be seen in the following Figures:

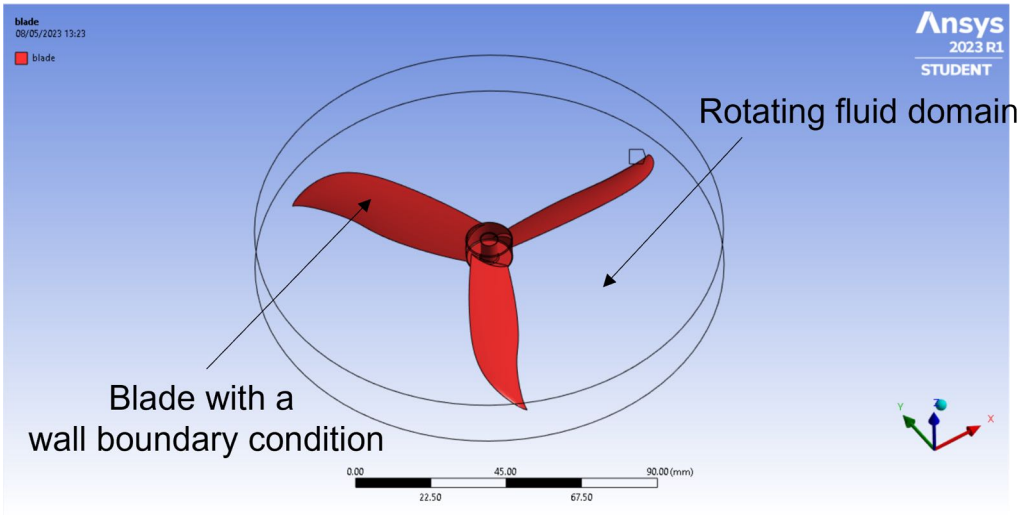

Figure 21: Blade boundary condition

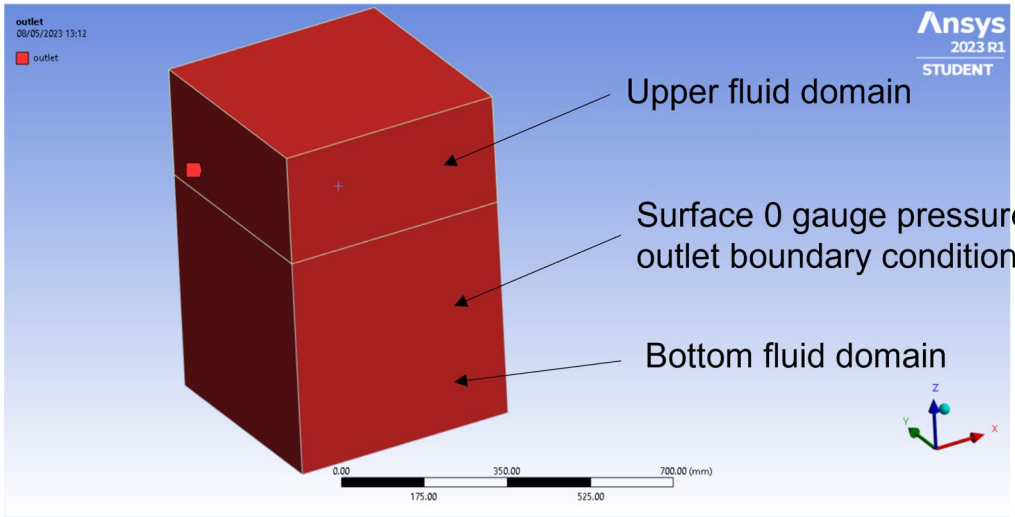

Figure 22: Domain boundary condition

As we can see the element size are still under our limit.

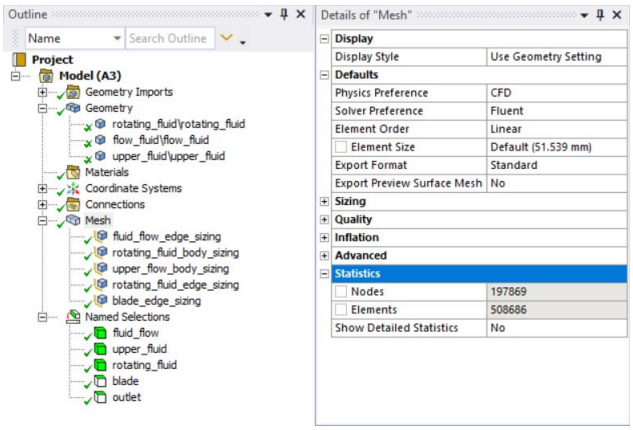

Figure 23: Mesh details

To achieve a homogeneous mesh for the fluid flow we can apply an edge sizing to the edge of the domain

with the same element size. The edge of the cube is $0.5m$ and the applied edge sizing is 50 division, thus one cubic element will be $0.01m$ by $0.01m$ by $0.01m$. The blade, the rotating fluid domain and the upper fluid domain was meshed with the consideration of the quality and the limitation of the elements.

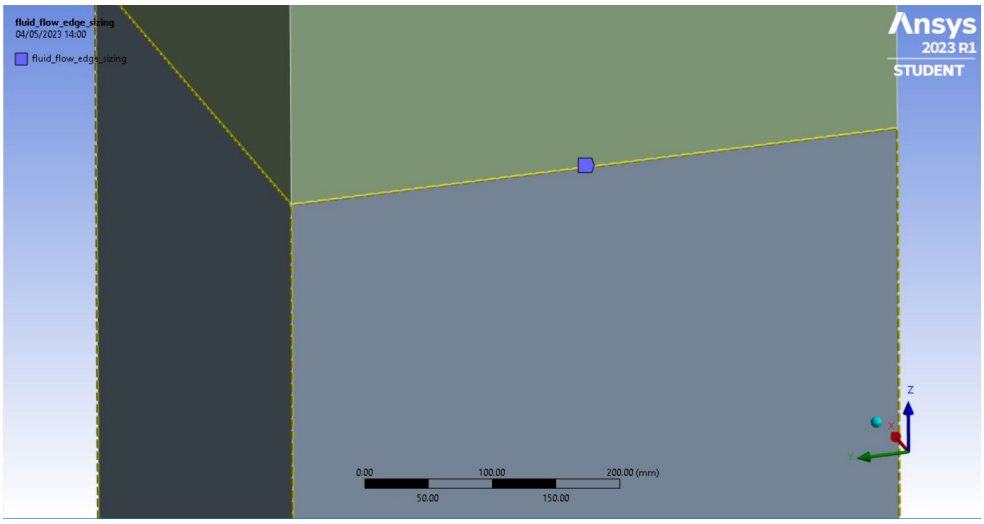

Figure 24: Fluid flow edge sizing

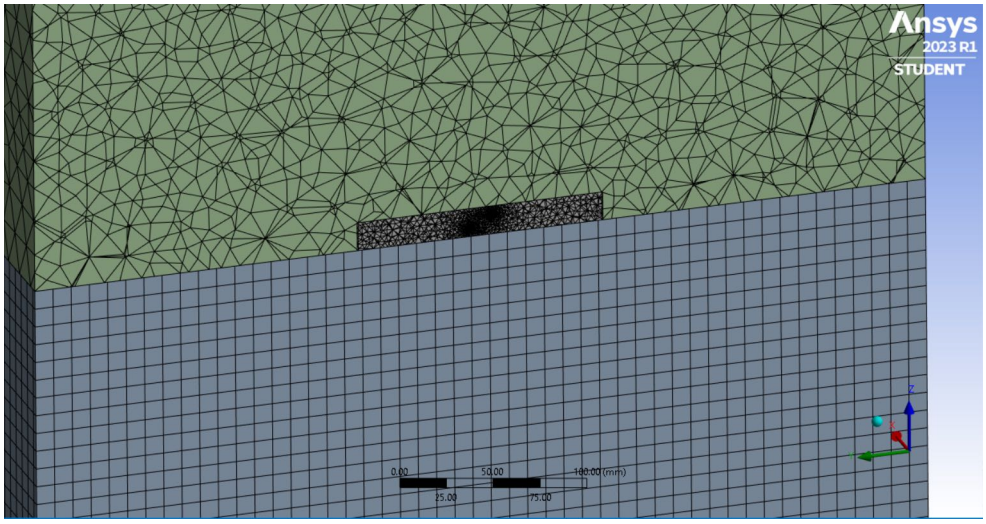

Figure 25: Mesh cross section

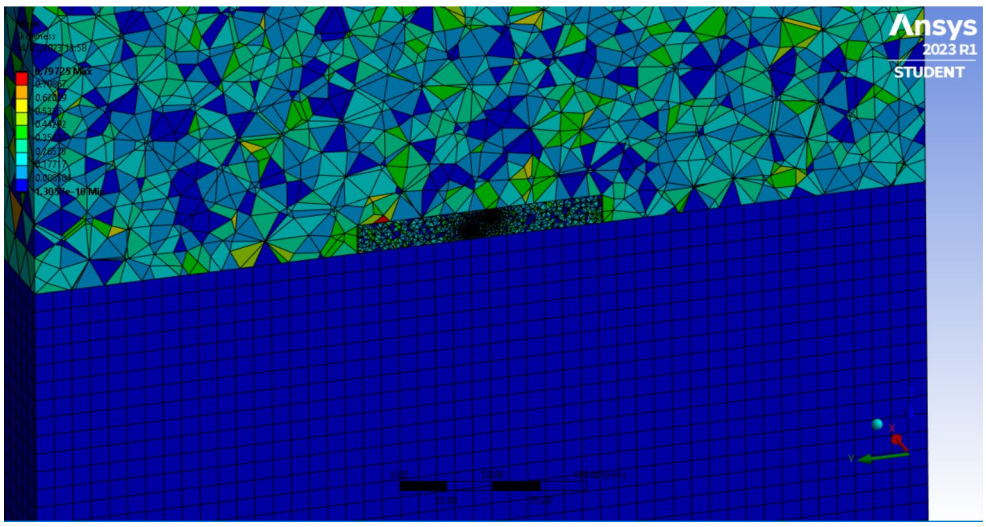

Figure 26: Mesh cross section colored with skewness

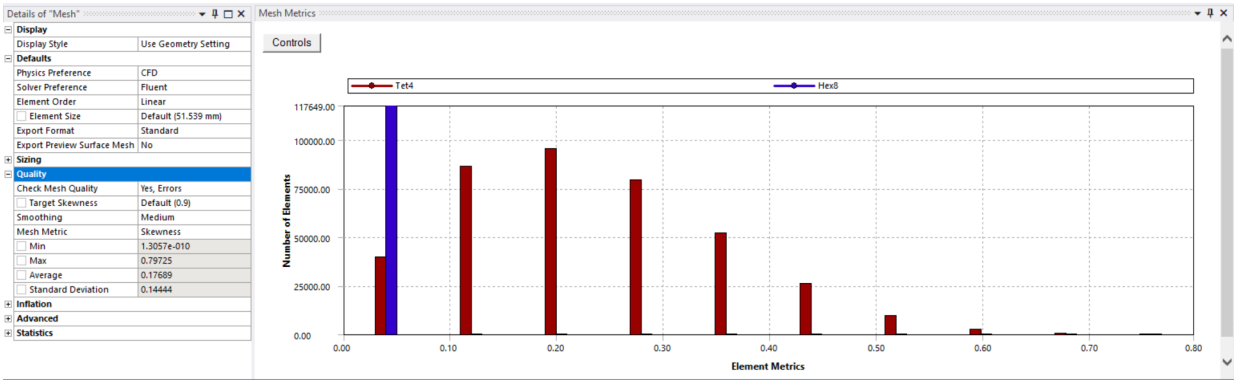

Figure 27: Mesh quality

As we can see the maximum skewness is 0.79 and the average is 0.17, thus our mesh is perfectly fine. Furthermore as we can see in Figure 26. the skewness of the fluid flow elements are 0 since they are perfectly cubical.

4.3 Solution (Fluent)

The problem setup and solution options are the same as in Section 3.3. (We just wanted a more convenient geometry in this whole Section 4., with the same solution).

4.4 Visualization (ANSYS workbench)

Since our solution options are the same as in the first version the results are the same just in a different domain. The flow animations:

1. *Full video: Velocity contour at a cross section*

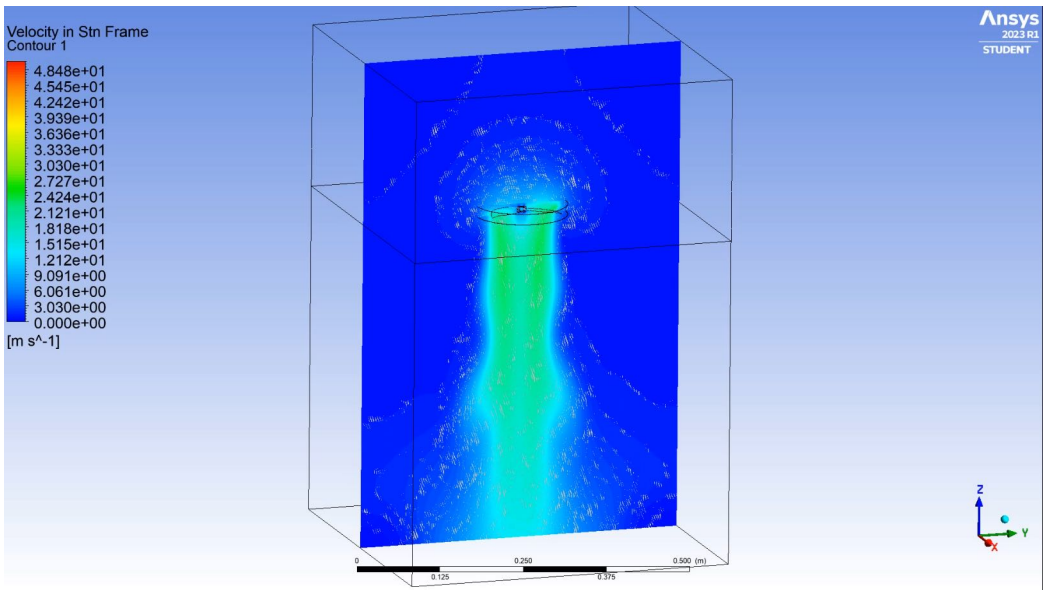

Figure 28: Velocity contour at a cross section

2. *Full video: Dynamic pressure contour at a cross section*

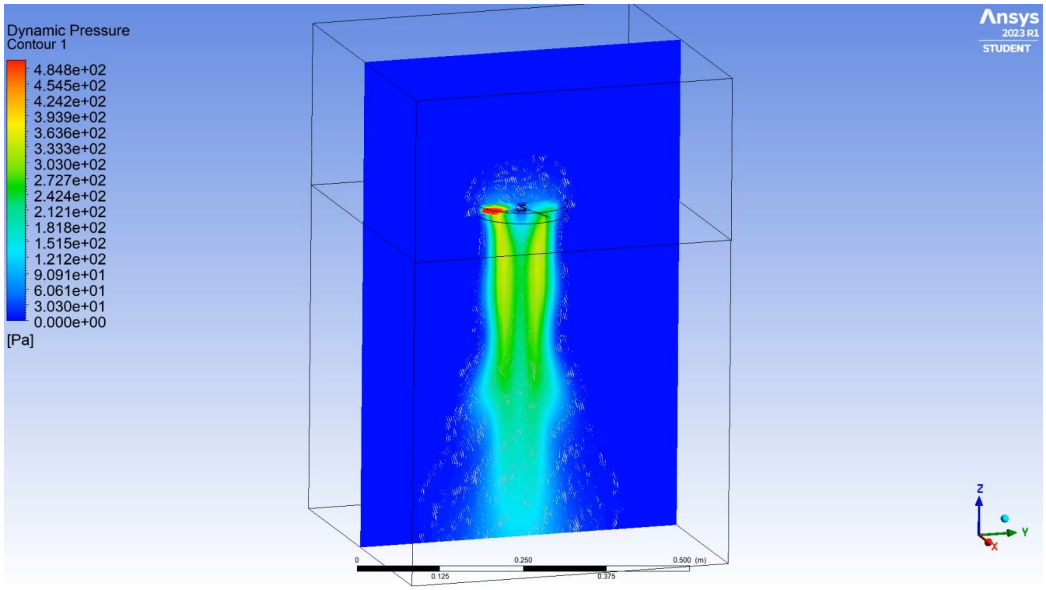

Figure 29: Dynamic pressure contour at a cross section

3. *Full video: Velocity streamlines*

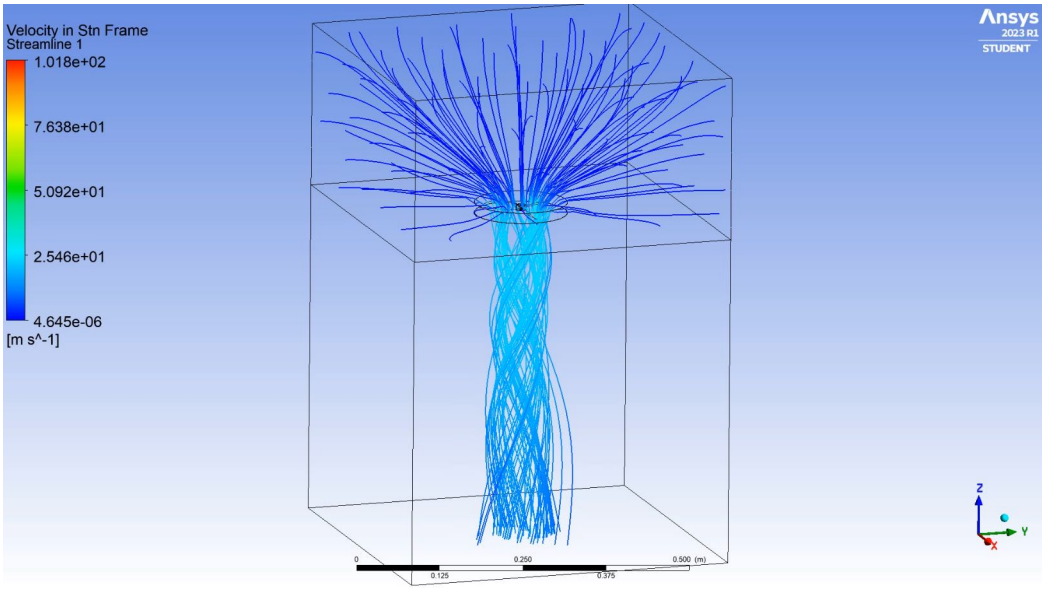

Figure 30: Velocity streamlines

5 Single rotor model (rectangular) with 45° gravity

To access the source files to this version go to: SZTAKI Nextcloud: CFD/single rotor rectangular 45 degree grav.

In this section we will examine what happens to the flow if the rotor is tilted. The rotor will be tilted by 45° which is equivalent with a 45° angle of gravitational acceleration. The motivation to this part is to see if there are big changes in the flow and if we can approximate the flow all time with the vertical one.

The geometry, the mesh and the solution options are the same as the previous version. Only the gravitational acceleration is $-6.93[m/s^2]$ in the x and z component, instead of the $-9.81[m/s^2]$ only in the z direction. Note that $9.81 \cdot \frac{\sqrt{2}}{2} = 6.93$ for the 45° rotation.

To investigate the differences aside the visualization we can compare the thrust force acting on the rotor in the two different scenarios.

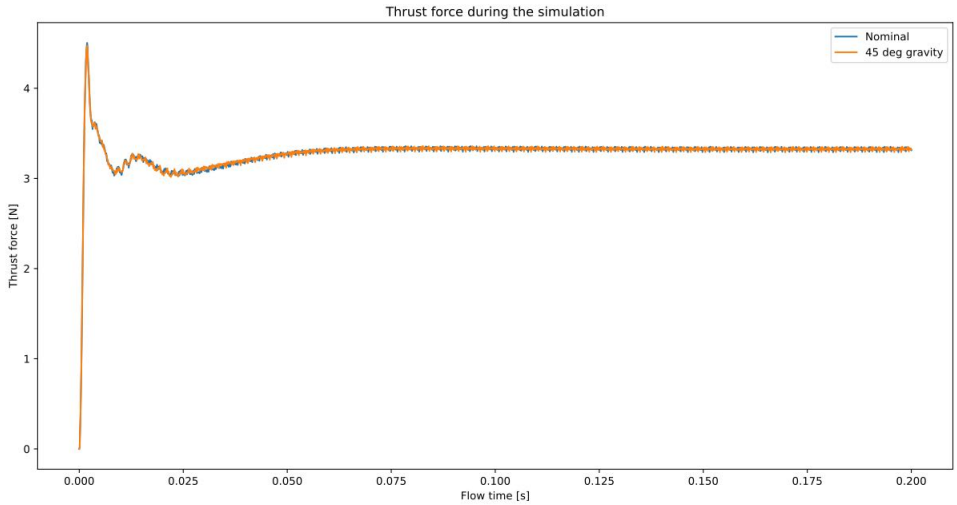

Figure 31: Thrust force comparison

As we can see the time series of the thrust force are nearly identical. We can also validate the flow since the

angular velocity of the rotor is 1200$[rad/s]$ which is a relatively fast speed, and the gravity does not affect our flow much if we use this high speed values.

Furthermore we can see the flow animations:

1. *Full video: Velocity contour at a cross section (z-x plane)*

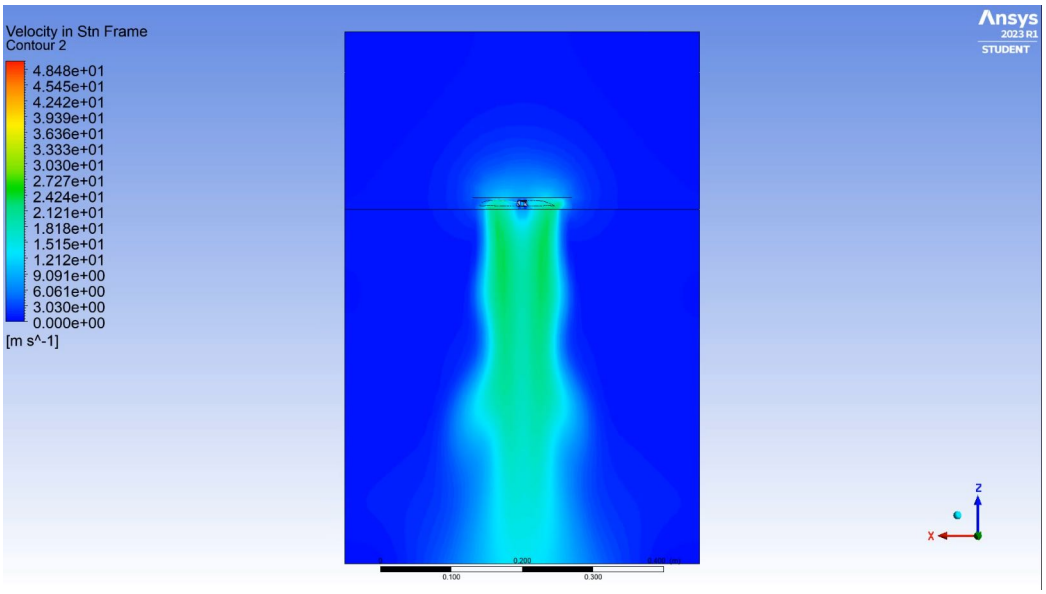

Figure 32: Velocity contour at a cross section (z-x plane)

2. *Full video: Velocity contour at a cross section (z-y plane)*

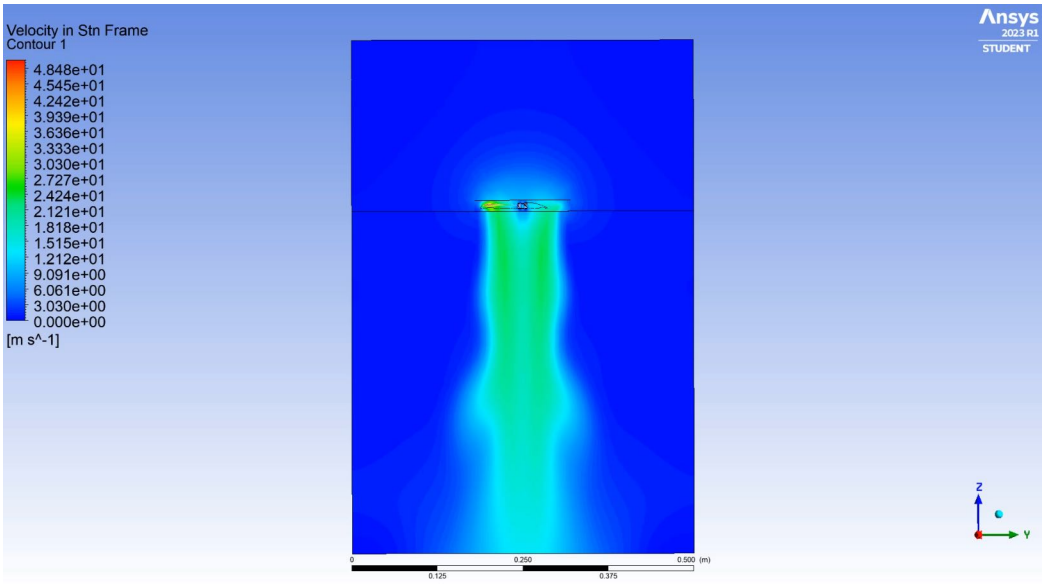

Figure 33: Velocity contour at a cross section (z-y plane)

3. *Full video: Dynamic pressure contour at a cross section (z-x plane)*

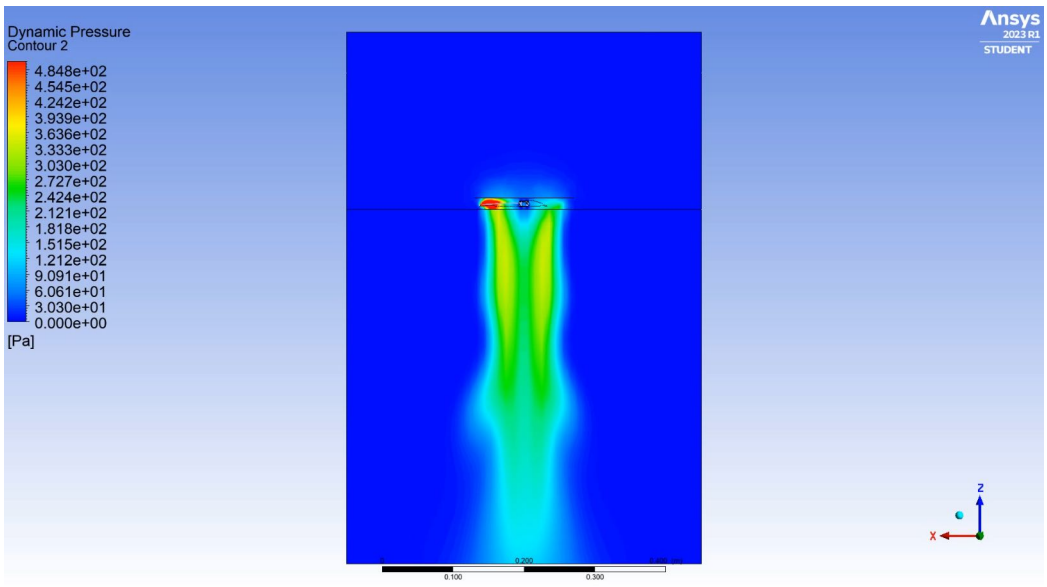

Figure 34: Dynamic pressure contour at a cross section (z-x plane)

4. *Full video: Dynamic pressure contour at a cross section (z-y plane)*

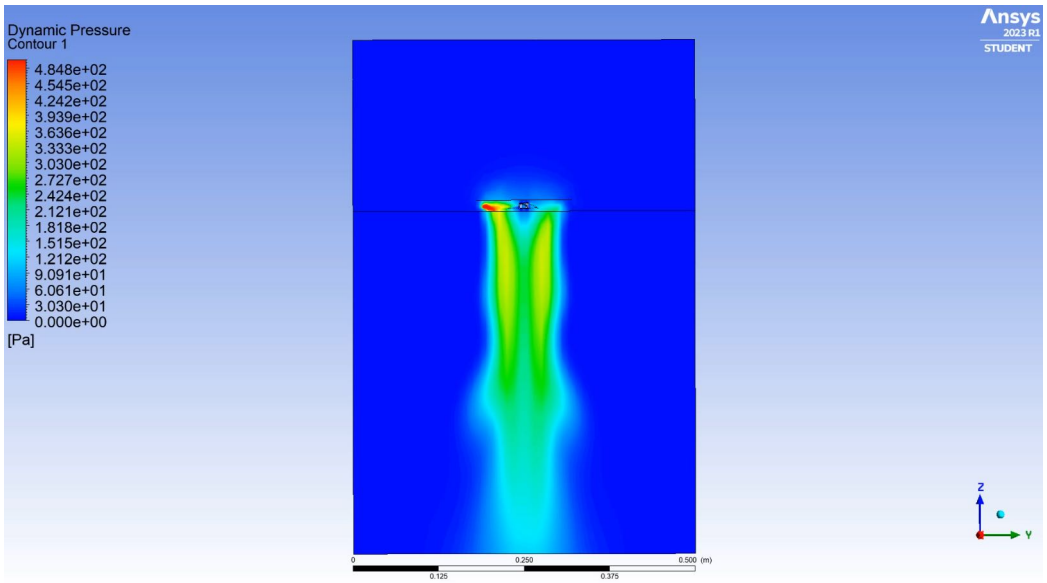

Figure 35: Dynamic pressure contour at a cross section (z-y plane)

5. *Full video: Velocity streamlines from the top*

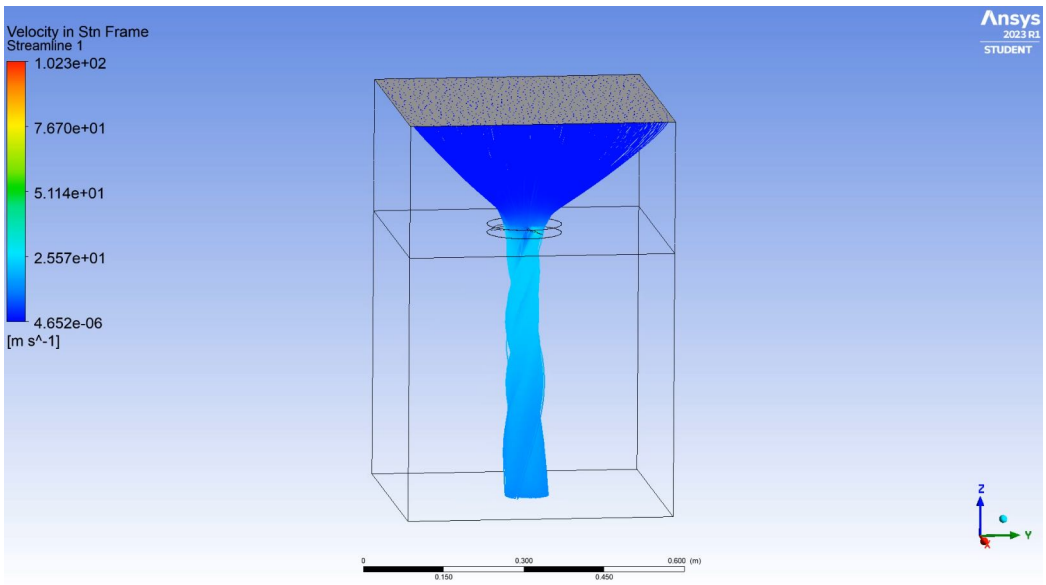

Figure 36: Velocity streamlines from the top

6. *Full video: Velocity streamlines from the rotor*

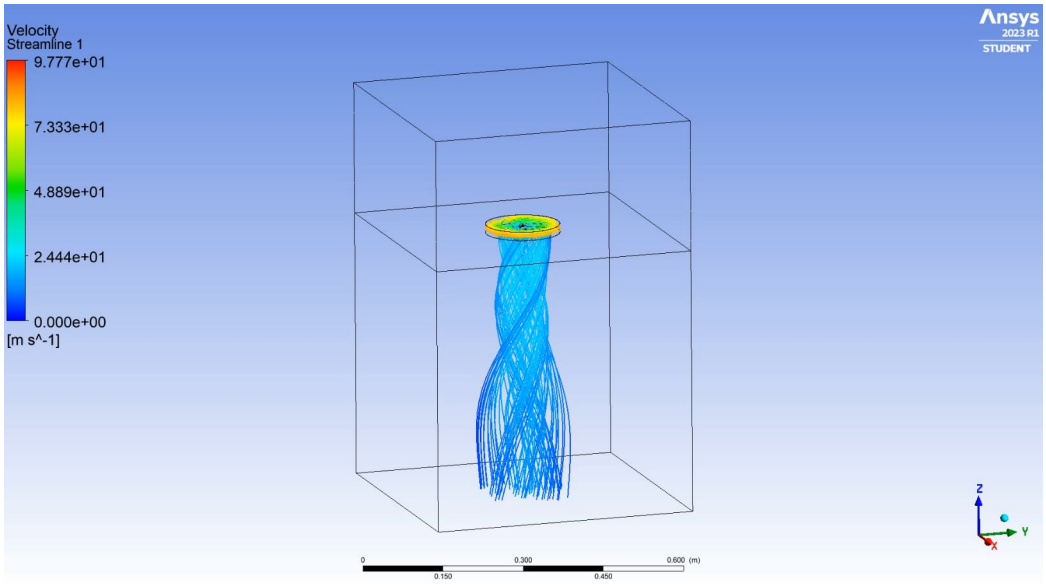

Figure 37: Velocity streamlines from the rotor

Also the flow in the z-y and z-x plane are nearly identical. We can plot the dynamic pressure values as a heat map to further visualize the flow.

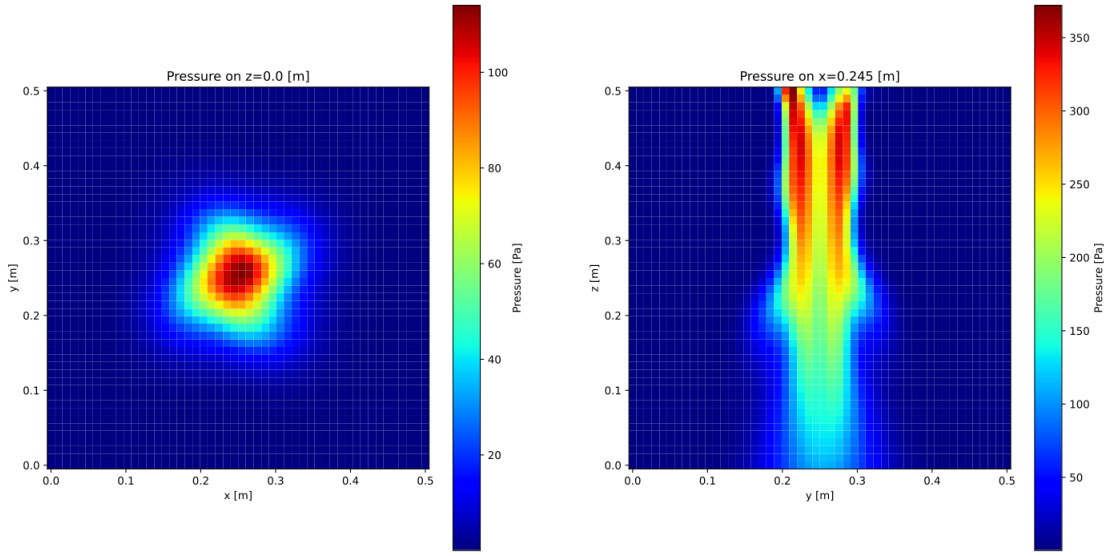

Figure 38: Dynamic pressure heat map of the flow

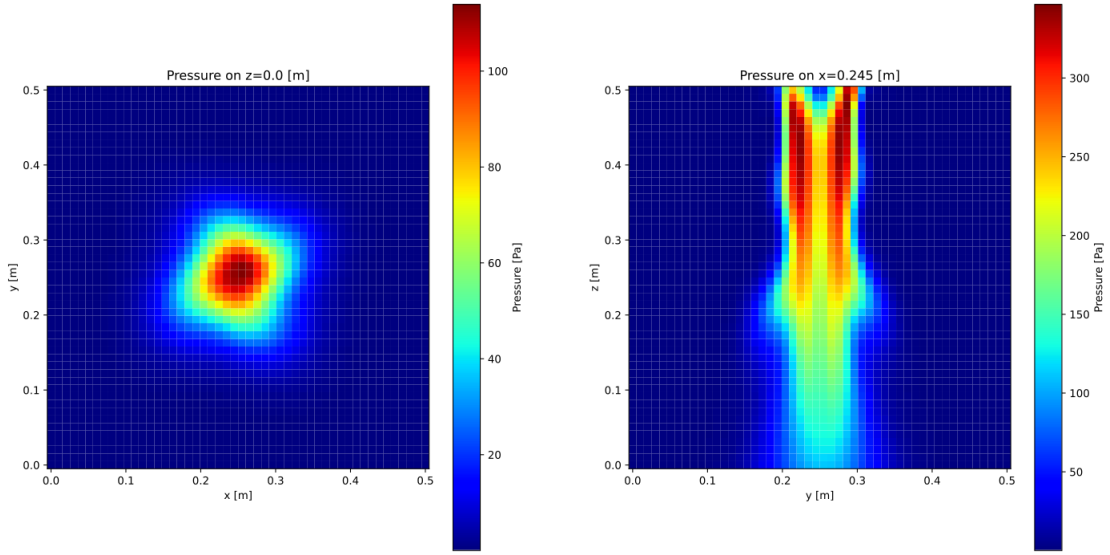

Figure 39: Dynamic pressure heat map of the flow (45° gravity)

The dynamic pressure (which will be implemented) heat maps are also very similar to each other. To measure the difference quantitatively, the error between the thrust forces are less than 0.1% (from Figure 31). Thus in conclusion we can say that the rotation of the flow does not affect much and we can use the original flow simulation with great approximation if the quadrotor goes through aggressive maneuvers.

6 Flow implementation

6.1 Pressure

To account for the effect of the flow we can export the pressure values from the *fluid flow domain* (which was meshed homogeneously). In fluid dynamics there are three important concepts related to pressure [3]:

- **Static Pressure** (p_s): Static pressure refers to the pressure exerted by a fluid when it is at rest or in equilibrium. It represents the force per unit area exerted by the fluid on a surface due to molecular collisions. Static pressure is isotropic, meaning it acts equally in all directions. It can be measured using a pressure sensor or gauge.

- **Dynamic Pressure** (p_d): Dynamic pressure is the pressure exerted by a fluid due to its motion or flow. It represents the kinetic energy per unit volume of the fluid. Dynamic pressure arises from the conversion of the fluid's velocity into pressure as it flows through a constricted region or speeds up. It is directly related to the fluid's velocity and density. The dynamic pressure is given by the equation:

$$p_d = 0.5 \cdot \rho \cdot v^2 \tag{1}$$

  where: ρ is the density of the fluid, v is the velocity of the fluid and p_d is the dynamic pressure.

- **Total pressure** (p_t): Total pressure, also known as stagnation pressure or pitot pressure, is the sum of the static pressure and the dynamic pressure of a fluid. It represents the maximum pressure that a fluid would have if it were brought to rest (brought to zero velocity) adiabatically (without heat transfer) and isentropically (without entropy change). Total pressure is measured using a device called a pitot tube, which is often used in aerodynamics and fluid flow measurements.

$$p_t = p_s + p_d \tag{2}$$

To summarize, static pressure represents the pressure of a fluid when it is at rest, total pressure represents the sum of the static and dynamic pressures, and dynamic pressure represents the pressure exerted by a fluid due to its motion. These pressure concepts are fundamental in fluid dynamics and are used to analyze fluid behavior, design efficient systems, and calculate forces and flow rates in various engineering applications. If we talk about the effect of the flow on an object created by moving rotor, the **dynamic pressure** is the best option to work with, since we want to account for the effects of the fluid motion. In Fluent we can export all of the pressure values on our mesh (we can export the values at the center of the cells or at the nodes).

6.2 Velocity

The velocity vector field is a fundamental output of a fluid flow simulation, representing the flow velocity at each point in the domain. It provides a complete picture of the flow behavior, showing both the magnitude and direction of the velocity at any given location. The velocity vector field:

- The velocity vector field is a 3D field that describes the flow velocity at every point in the domain.
- At each location (X, Y, Z), the velocity vector is represented by three components: U (velocity in the X direction), V (velocity in the Y direction), and W (velocity in the Z direction).
- The magnitude of the velocity vector (V) at a point represents the flow speed at that location. It tells you how fast the fluid is moving at that specific point.
- The direction of the velocity vector indicates the flow direction at that point. The arrow's orientation points in the direction of the fluid flow at that location.
- The color of the velocity vectors can also be used to visualize additional information, such as the flow speed (by scaling the arrow length) or other flow properties.

6.3 1 rotor flow implementation

We can use our simulated one rotor flow to implement the flow created by a single rotor. In this case the dynamic pressure values are directly exported from the CFD simulation as a look-up table. During the implementation the middle of the flow is constrained under just 1 rotor. See the results of the implementation for hovering in Section 6.6.1.

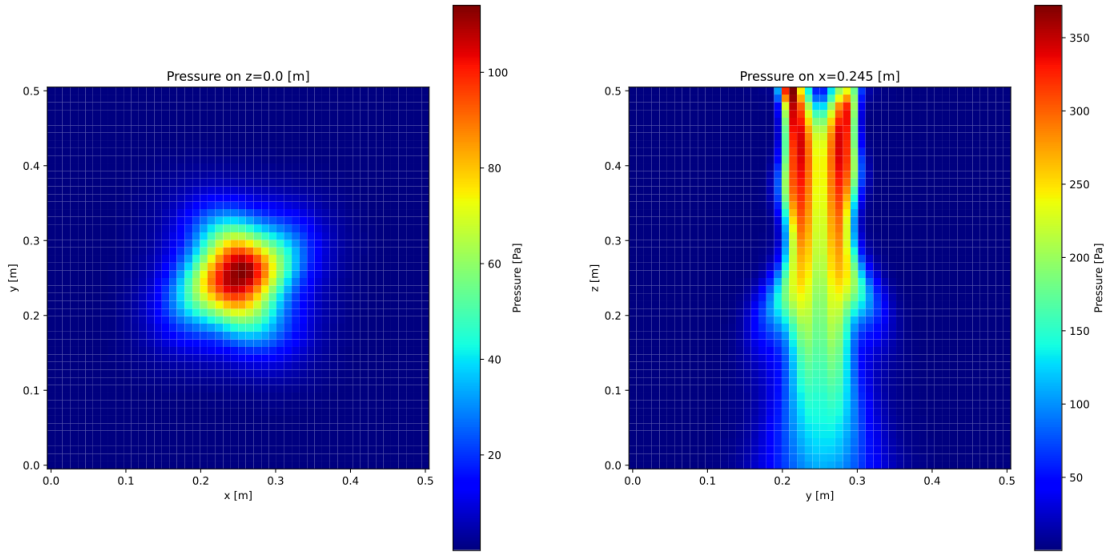

Figure 40: Dynamic pressure heat map of a single rotor

6.4 4 rotor flow implementation

Naturally we want to model the whole flow under the quadrotor, but thanks to licence limitations we can't. Thus the next best step is to construct the flow from our 1 rotor simulation. The concept is to slide and add together the dynamic pressure values which were simulated by only 1 rotor to get a dynamic pressure look-up table which approximates a flow created by 4 rotors. We can slide the origin of the flow by $l = 0.085m$ which is the offset of the rotors relative to the origin of the Bumblebee quadrotor.

6.4.1 Separate construction

In this construction we just simply slide the pressure values by our offset to get the flow domain. In each quarter the proper pressure values are assigned under the individual rotors. Thus we in this case the pressures under the individual rotors does not influence each other.

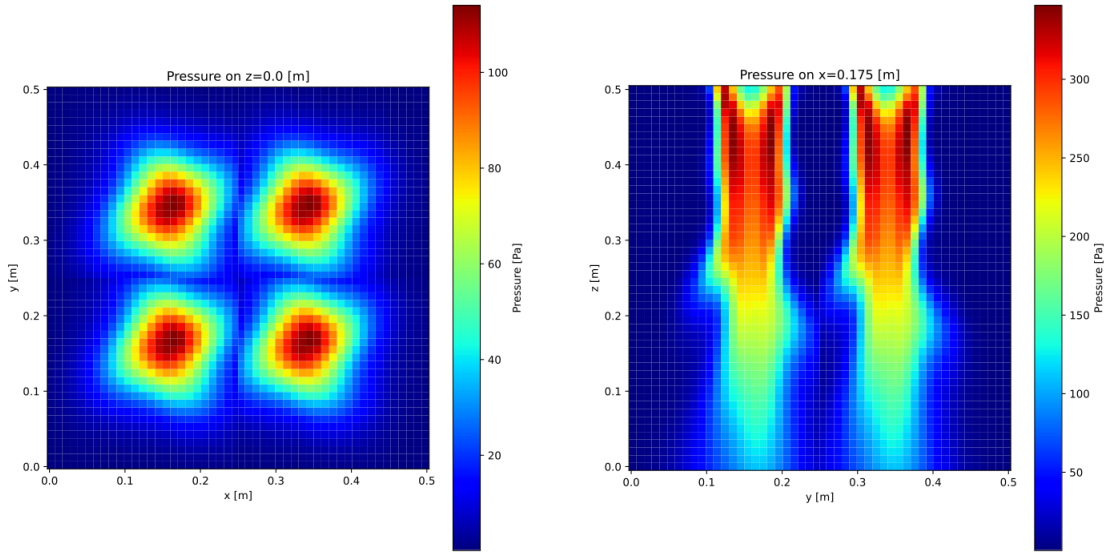

Figure 41: Dynamic pressure heat map of 4 rotors

6.4.2 Influenced construction

We can construct a version where we add the overlapping pressure values of each rotor to approximate the effect of the rotors to each other. This construction try to approximate more the influence of the rotors to each other, but the whole CFD simulation can validate it only.

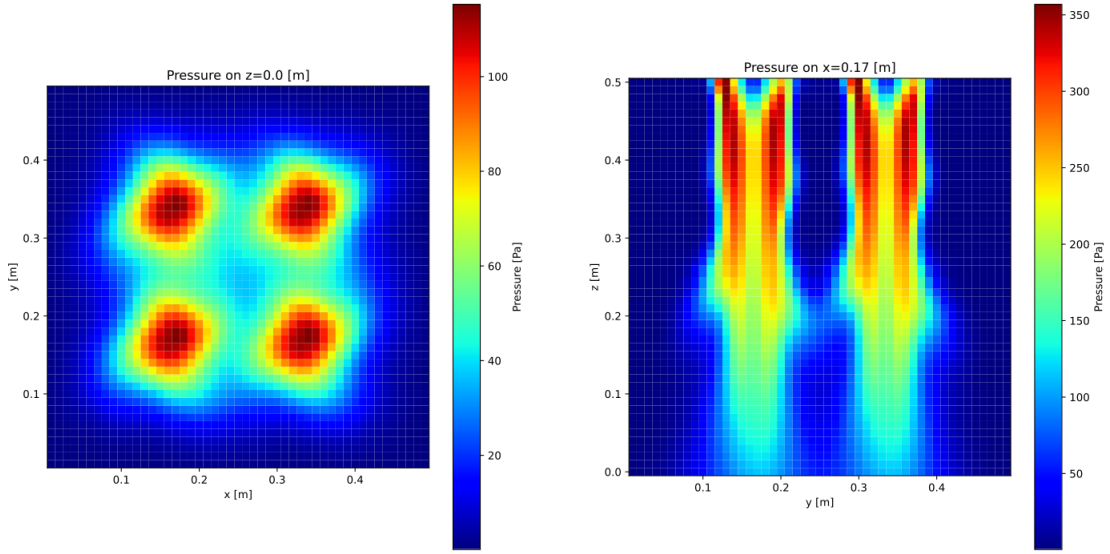

Figure 42: Dynamic pressure heat map of 4 rotors (influenced by each other)

As we can see we can get a more continuous pressure map. One further improvement is to mirror 2 diagonal values during the construction of the 4 rotor look up table, to model the cw and ccw construction of the rotors.

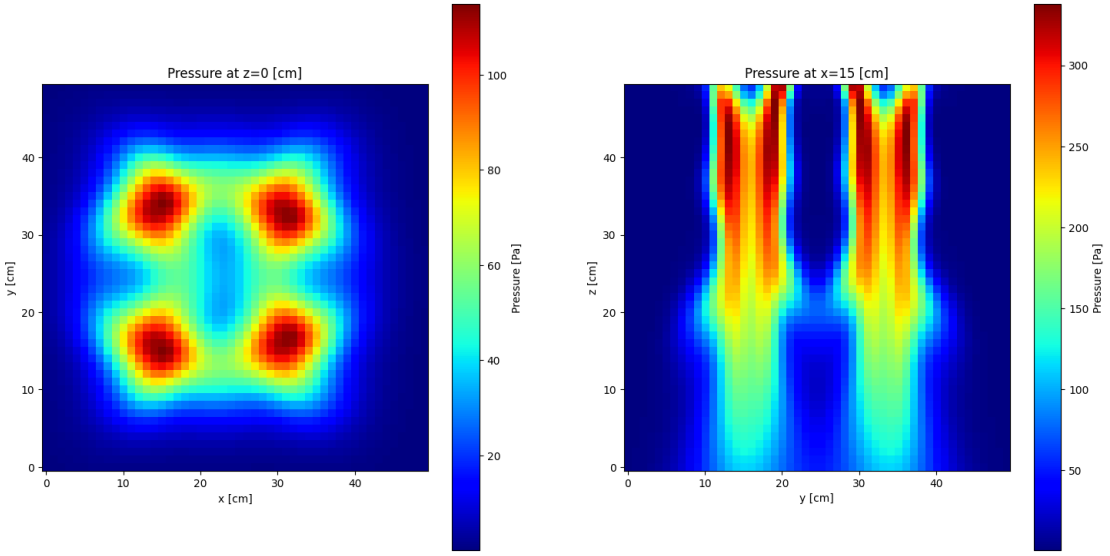

Figure 43: Dynamic pressure heat map of 4 rotors (influenced by each other), mirrored

Also we can export the velocity values as a vector field from all of the meshes is the flow field.

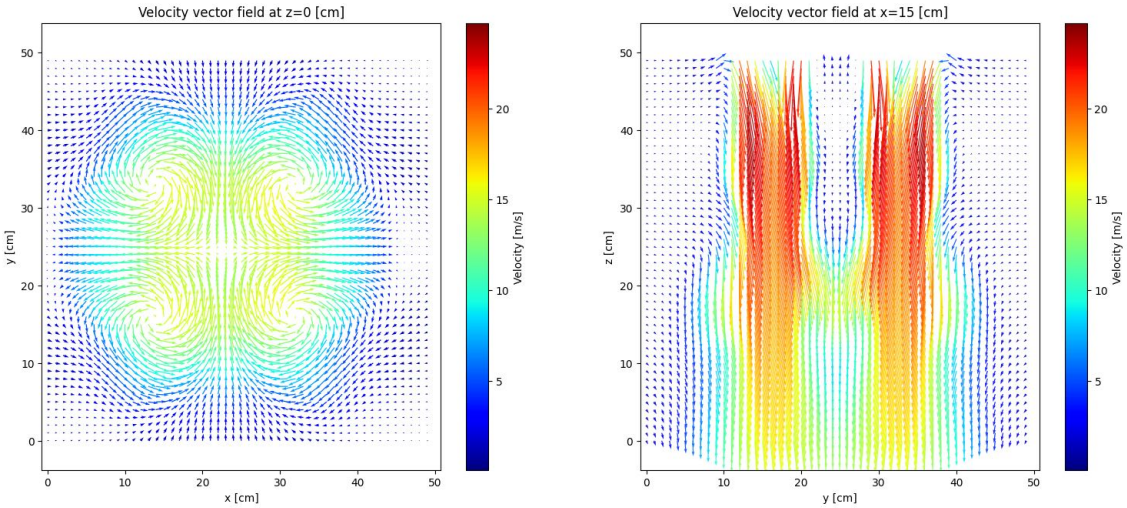

Figure 44: Velocity map of 4 rotors (influenced by each other), mirrored

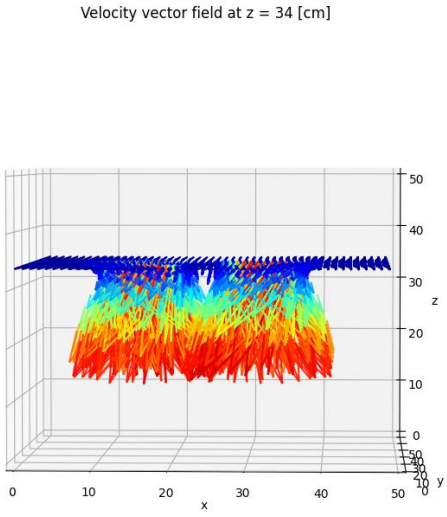

Figure 45: Velocity vector field, from the side

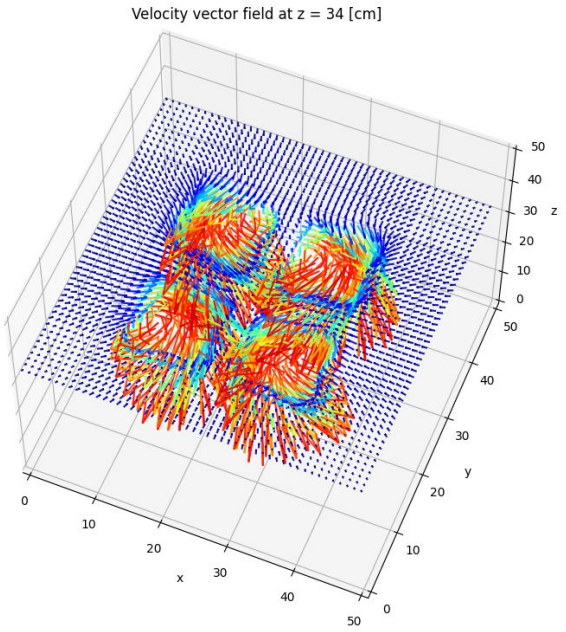

Figure 46: Velocity vector field

Some gifs for better understanding:

1. *Velocity vector field, 3D, from top to bottom*
2. *Velocity vector field, 2D cross section, from top to down*
3. *Velocity vector field, 3D cross section, from top to down*
4. *Velocity vector field, 3D cross section, from left to right*

In the MuJoCo 4 rotor implementation, this *influenced construction* version was used. See the results of the implementation for hovering in Section 6.6.2 and for payload transportation see Section 6.6.3.

6.5 MuJoCo implementation

For the flow implementation the following steps are necessary:

1. Define the system setup and define the coordinate frames, which includes the calculation of the position and rotation of the *Flow frame* in the *World frame*.
2. Mesh the surface of our payload.
3. For every surface mesh calculate the positions and rotations in the *World frame* and their normal vector.
4. For every surface mesh calculate the positions in the *Flow frame*.
5. Convert the positions in the *Flow frame* to indecies and sample the pressure flow with that indecies.
6. Finally calculate the force and torque acting on the payload via. Momentum Integral Equation [3].

6.5.1 System setup and frame definitions: Step 1.

Remark: During the implementation the flow is shifted up by 15 cm, because the length of the hook is around 0.5m, thus the payload cannot even fit into the flow. A bigger flow domain is advised in further development.

During the MuJoCo implementation the only difference between the 1 and 4 rotor implementation is the constrained place of the flow (plus the actual dynamic pressure look up tables). The setup of the system:

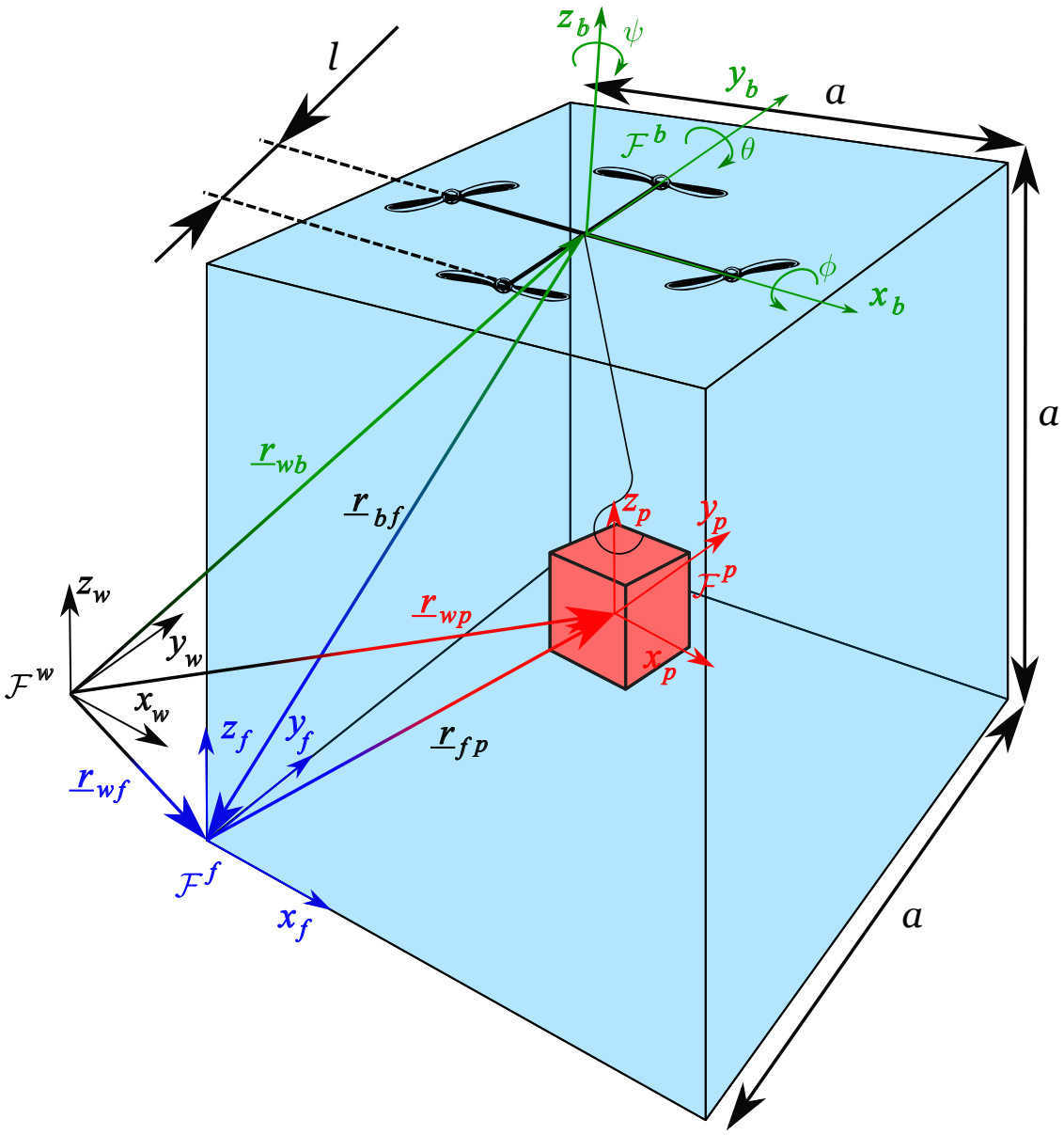

Figure 47: System setup

Geometric notations:

- $l = 0.085m$: Is the offset length of the rotors from the center of mass of the quadrotor.
- $a = 0.5m$: Edge length of the flow.

Coordinate frame notations:

- F^w: *World frame* (or inertial).
- F^b: *Body frame* of the quadrotor, with the proper Euler angles: ϕ (roll), θ (pitch) and ψ yaw.
- F^p: *Payload frame.*
- F^f: *Flow frame.*

Vector notations (all in the World F^w frame):

- $\mathbf{r}_{wb}$: Translational vector from F^w to F^b.
- $\mathbf{r}_{wp}$: Translational vector from F^w to F^p.
- $\mathbf{r}_{wf}$: Translational vector from F^w to F^f.

- $\mathbf{r}_{bf}$: Translational vector from F^b to F^f.
- $\mathbf{r}_{fp}$: Translational vector from F^f to F^p.

The frames are defined by their translation and rotation from the World F^w frame. The translation vectors denoted by $\mathbf{r} \in \mathbb{R}$ and the rotation is denoted by $q \in \mathbb{C}$, where q is a quaternion.

Frame definitions:

- The *Body frame* F^b: Body frame of the quadrotor is defined by the $\mathbf{r}_{wb}$ translational vector and the q_w^b quaternion rotation. Both are measured in the MuJoCo environment and also in the real world by the OptiTrack motion capture system.
- The *Payload frame* F^p: The frame of the payload is defined by the $\mathbf{r}_{wp}$ translational vector and the q_w^p quaternion rotation. Both are measured in the MuJoCo environment and also in the real world by the OptiTrack motion capture system.
- The *Flow frame* F^f: The frame of the flow is defined by the $\mathbf{r}_{wf}$ translational vector which can be calculated by:

$$\mathbf{r}_{wf} = \mathbf{r}_{wb} + \mathbf{r}_{bf} \tag{3}$$

where $\mathbf{r}_{bf}$ is the offset vector in the *World frame* (See: Vector 6.5.1. in Figure 51.), but the offset in the *Body frame* ($\mathbf{r}_{bf}^b$) is a constant value. This offset value is the difference between the 1 rotor and 4 rotor implementation. In both cases this offset is constant since the flow is constrained to be under the quadrotor. The offsets are:

$$\mathbf{r}_{bf,1rotor}^b \begin{bmatrix} x_b & y_b & z_b \end{bmatrix} = \begin{bmatrix} -\frac{a}{2} & -l - \frac{a}{2} & -a \end{bmatrix}^T \tag{4}$$

and

$$\mathbf{r}_{bf,4rotor}^b \begin{bmatrix} x_b & y_b & z_b \end{bmatrix} = \begin{bmatrix} -\frac{a}{2} & -\frac{a}{2} & -a \end{bmatrix}^T \tag{5}$$

where $l = 0.085m$ is the distance of the rotor from the center of mass of the quadrotor and $a = 0.5m$ is the edge size of the flow. As we can see the only difference is the $-l$ offset in Equation 4. Note that the choosing of the rotor where the flow will be implemented, depends on the sign and the component of the vector. The plus and minus l offsets in the x_b and y_b direction describes all the 4 rotors. Note that this offset is in the *Body frame*, thus we have to convert it in the *World frame*:

$$\mathbf{r}_{bf} = \mathbf{r}_{bf}^w = q_w^{b*} \otimes \mathbf{r}_{bf}^b \otimes q_w^b \tag{6}$$

where q_w^{b*} is the conjugate quaternion of the original rotation and $\otimes$ is a special multiplication operator for quaternions. And to rotate a $\mathbb{R}^3$ vector, we can quaternion multiply the $\mathbb{R}^3$ vector from left to the conjugate of the rotation and quaternion multiply it from right with the original rotation, where the scalar part of the vector is always considered zero. Finally:

$$\mathbf{r}_{wf} = \mathbf{r}_{wb} + q_w^{b*} \otimes \mathbf{r}_{bf}^b \otimes q_w^b \tag{7}$$

For the rotation:

$$q_w^f = q_w^b \tag{8}$$

hence the rotation of the flow is the same as the rotation of the quadrotor. With these constrains the flow is always below the quadrotor.

6.5.2 Payload mesh: Step 2.

To mesh the surface of our payload we simply made a grid at the top of the payload. This process was made in python environment where the top surface of the payload was divided to equal square elements along the edges. In the current version a 10 by 10 grid, thus 100 surface mesh elements are implemented. Note that the number of mesh elements of the payload affect the computational cost of the calculation significantly.

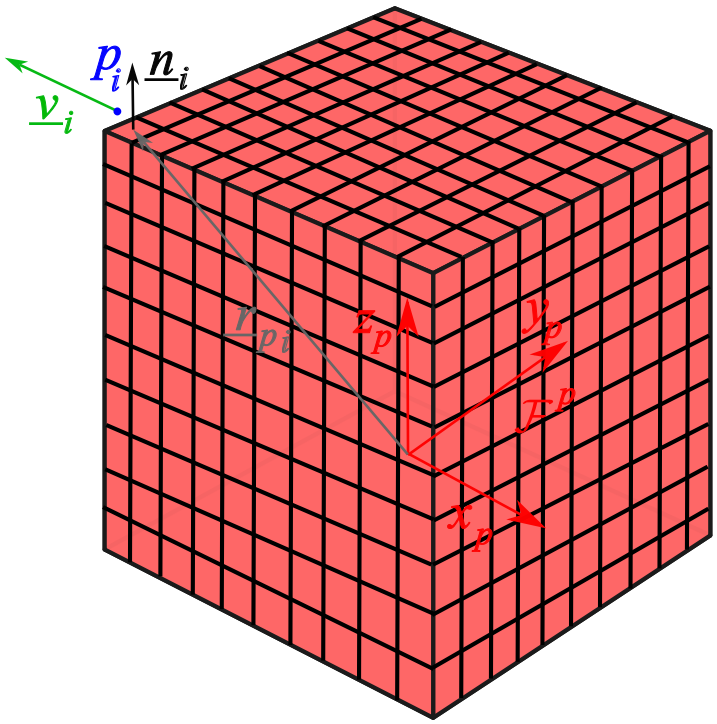

Figure 48: Meshed payload

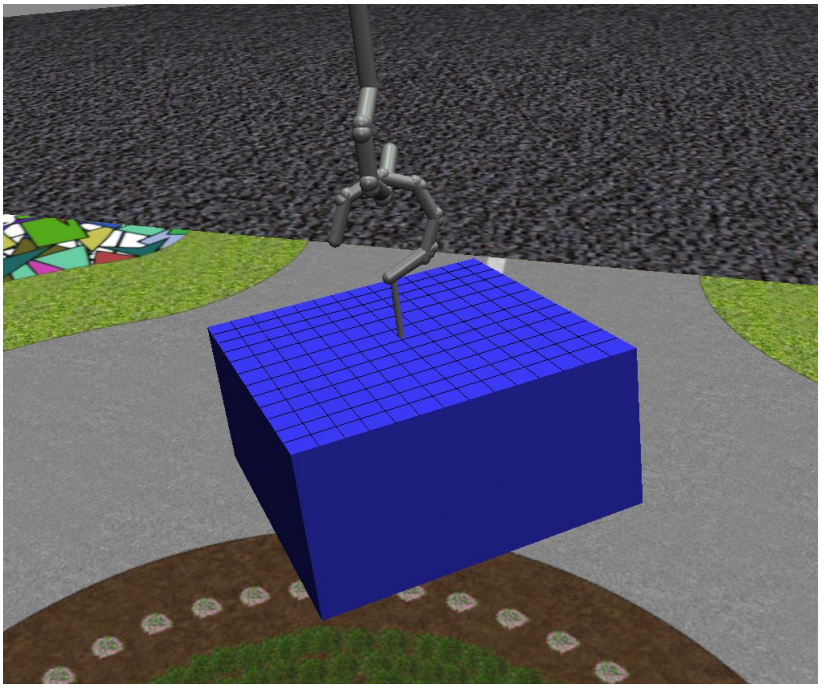

Figure 49: Meshed payload in MuJoCo

6.5.3 Surface mesh calculations: Step 3.

To calculate the positions of the surface meshes in the *World frame* we can add the translational vector of the payload to the position of the mesh grid in the *Payload frame*:

$$\mathbf{r}_{wi} = \mathbf{r}_{wp} + \mathbf{r}_{pi} \tag{9}$$

where $\mathbf{r}_{pi}$ is the position of the mesh grid in the *World frame*, but it is a constant value in the *Payload frame* ($\mathbf{r}^p_{pi}$) for each surface mesh, see Figure 48. Thus we have to convert it into the *World frame*:

$$\mathbf{r}_{pi} = \mathbf{r}^w_{pi} = q^{p*}_w \otimes \mathbf{r}^p_{pi} \otimes q^p_w \tag{10}$$

hence:

$$\mathbf{r}_{wi} = \mathbf{r}_{wp} + q^{p*}_w \otimes \mathbf{r}^p_{pi} \otimes q^p_w \tag{11}$$

We also have to calculate the normal vector of the meshes, but since our payload is rectangular the normal vector is same for all grids. The normal vector is the *Payload frame*:

$$\mathbf{n}^p_{pi} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tag{12}$$

We can convert it into the *World frame*:

$$\mathbf{n}_{pi} = \mathbf{n}^w_{pi} = q^{p*}_w \otimes \mathbf{n}^p_{pi} \otimes q^p_w \tag{13}$$

6.5.4 Transformation: Step 4.

For Step 4, which is the transformation of the *Payload frame* to the *Flow frame*, we have to determine the position of the individual meshes ($\mathbf{r}^w_{wi}$, Equation 11.) in the *Flow frame*: $\mathbf{r}^f_{wi}$. We can write up the transformation of the payload:

$$\mathbf{r}^w_{fp} = \mathbf{r}^w_{wp} - \mathbf{r}^w_{wf} \tag{14}$$

for $\mathbf{r}^w_{fp}$, see: Vector 6.5.1. in Figure 51. But we can write up this equation for the individual meshes, which are already include the rotation of the payload (Equation 11).

$$\mathbf{r}^w_{fi} = \mathbf{r}^w_{wi} - \mathbf{r}^w_{wf} \tag{15}$$

This is the positions of the individual meshes from the *Flow frame* in the *World frame*. We can convert them into the *Flow frame* itself:

$$\mathbf{r}^f_{fi} = q^f_w \otimes \mathbf{r}^w_{fi} \otimes q^{f*}_w \tag{16}$$

6.5.5 Array sampling: Step 5.

The main idea to sample the pressure and velocity arrays for each individual mesh on the payload, is to convert the position vectors of the individual meshes into the *Flow frame* (whole reason for Step 4) and round it up to get an equivalent index in the arrays (Step 5). This process determines the closest pressure and velocity values for each surface mesh elements without a complex algorithm, which speeds up the process drastically. As mentioned our flow is a 50 by 50 by 50 array along the 3 dimensional space which is equivalent to the coordinate value of the flow since one index represent a $0.01m$ spatial step ($50 \cdot 0.01m = 0.5m$). In this way we can transform the position vectors of the mesh grids to an index value which represents the closest dynamic pressure and velocity value to that position vector. To do that we can multiply the position vectors (in the coordinate system of the flow) with 100 and round them to an integer and we can get the corresponding dynamic pressure and velocity value of our flow array.

$$\begin{bmatrix} b & c & d \end{bmatrix} = round(\mathbf{r}^f_{fi} \cdot 100) \tag{17}$$

where $a, b, c = 1, 2, .., 50$ represents the index value in the dynamic pressure or velocity array. The array sampling is simply:

$$a_i = \mathbf{A} \begin{bmatrix} b & c & d \end{bmatrix} \tag{18}$$

where $\mathbf{A}$ is our 50 by 50 by 50 array pressure and velocity look up table (Section 6.4). Note that for one flow mesh cell point correspond to dynamic pressure and velocity value. Also see Figure **??** for p_i and $\mathbf{v}_i$.

An example (for pressure) for better understanding:

$$\mathbf{r}^f_{fi} = \begin{bmatrix} 0.3715 & 0.4824 & 0.073 \end{bmatrix} m \tag{19}$$

is the position value of the i^{th} mesh element is the *Flow frame*, thus the position value after Step 4. Then to achieve the index value:

$$\begin{bmatrix} i & j & k \end{bmatrix} = round(\mathbf{r}^f_{fi} \cdot 100) = round(\begin{bmatrix} 0.3715 & 0.4824 & 0.073 \end{bmatrix} \cdot 100) = round(\begin{bmatrix} 37.15 & 48.24 & 7.3 \end{bmatrix}) = \begin{bmatrix} 37 & 48 & 7 \end{bmatrix} \tag{20}$$

Pressure sampling:

$$p_i = \mathbf{P} \begin{bmatrix} 37 & 48 & 7 \end{bmatrix} \tag{21}$$

6.5.6 Force and torque calculations: The Momentum Integral Equation [3] (Step 6.)

Finally we can calculate the forces acting on the individual mesh grids on the payload and calculate their torque from the center of mass of the payload. The individual force values can be added together for the center of mass, but the torque values must be calculated for each mesh point individually before adding them together. We can apply the Momentum Integral equation to calculate the total force and torque values. The Momentum Integral equation express the equation of motion in an integral form. To apply this equation we have to define a control volume in the space where we want to calculate the volume and surface integrals. If we want to calculate the force acting on a rigid body in a flow, we should include the rigid body in the control volume. Since the flow cannot go through the rigid body we exclude it's volume from the control volume.

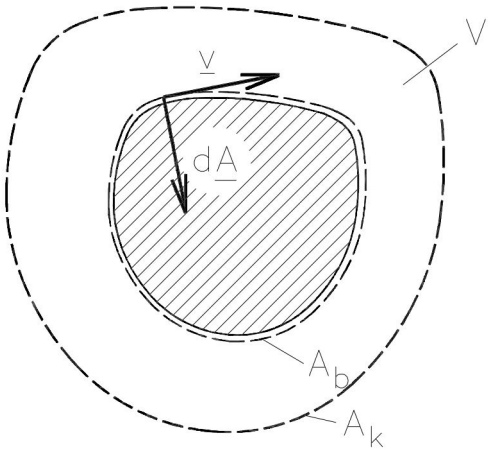

Figure 50: Rigid body in the control volume [3]

The Momentum Integral equation:

$$\frac{\partial}{\partial t}\int_V (\rho \mathbf{v})dV + \int_A \mathbf{v}\rho(\mathbf{v}d\mathbf{A}) = \int_V \rho \mathbf{g}dV - \int_A pd\mathbf{A} - \mathbf{F} - \mathbf{S} \tag{22}$$

where:

- V: Is our control volume.
- A: Surface of our control volume.
- t: Time.
- $\mathbf{v}$: Velocity of the flow.
- ρ: Density of the flow. In our case constant everywhere. $\rho_{air} = 1.1839\frac{kg}{m^3}$ in room temperature.
- p: Dynamic pressure created by the flow.
- $\mathbf{g}$: Gravitational acceleration.
- $\mathbf{F}$: The force acting on the rigid body inside the control volume.
- $\mathbf{S}$: The force created by friction (we can neglect it).

The left hand side of the equation express the change of the flow momentum and the right hand side express the forces acting on the fluid. In the left hand side where change of the flow momentum is expressed, the first term indicates the rate of change of the velocity (or density) and the second term express the spatial moving of the flow in time. In the right hand side the first term expresses the gravitational force created by the flow, the second expressed the force created by the pressure of the flow, the third is the force acting on the rigid body and the fourth is the friction. As mentioned before the control volume where we apply this equation is our choice, thus we can define our control volume infinitesimally (by a very little ϵ) close to our rigid body, and also the control volume has the same rotation and position as our rigid body (payload).

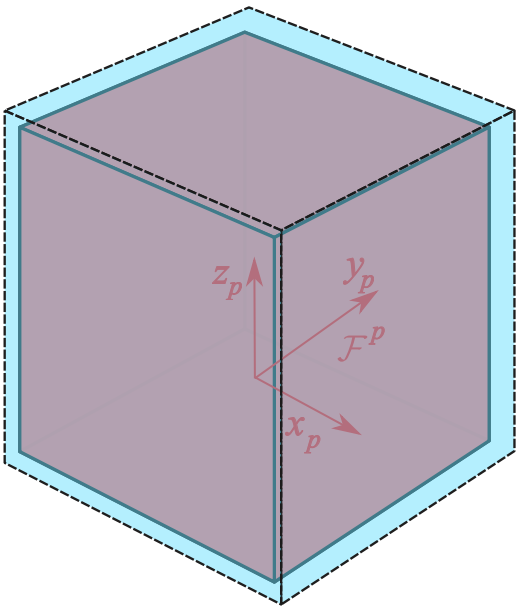

Figure 51: Control volume defined around our payload

In this way the Integral Momentum equation can be simplified as follows:

$$\int_A \mathbf{v}\rho(\mathbf{v}d\mathbf{A}) = -\int_A pd\mathbf{A} - \mathbf{F} \tag{23}$$

If we choose our control volume very close to our rigid body, and the volume of the rigid body must be extracted from the control volume, we can neglect our volume integrals in Equation 22. Also we want to calculate the forces acting on our rigid body, thus:

$$\mathbf{F} = -\int_A pd\mathbf{A} - \int_A \mathbf{v}\rho(\mathbf{v}d\mathbf{A}) \tag{24}$$

Notice that the direction of the force created by the pressure has the same direction as the normal vector of the surface, but in the second term the force will always point to the same direction as the velocity of the flow. To calculate the force numerically, we can add together the calculated forces on each surface mesh in our payload (see Figure 48) to get the total force acting on our payload.

$$\mathbf{F}_t = \sum_{i=1}^{N} \mathbf{F}_i = \sum_{i=1}^{N} -p_i d\mathbf{A}_i - \mathbf{v}_i\rho(\mathbf{v}_i d\mathbf{A}_i) \tag{25}$$

where:

- F_i and F_t: Is the force acting on the i^{th} mesh element and the total force acting on the payload, respectively.
- N: Number of mesh elements (resolution) in the surface of our payload.
- p_i: Is the sampled dynamic pressure value (Section 5) for the i^{th} mesh element.
- $\mathbf{v}_i$: Is the sampled velocity value (Section 5) for the i^{th} mesh element.
- $d\mathbf{A}_i$: Is surface area for the i^{th} mesh element, with the direction of it's normal vector.
- ρ: Density of the air (constant everywhere).

We can add together the individual calculated force values to get the total force acting on the payload, but we have to calculate the total torque in the following way:

$$\mathbf{M}_t = \sum_{i=1}^{N} \mathbf{M}_i = \sum_{i=1}^{N} \mathbf{r}_{pi} \times \mathbf{F}_i \tag{26}$$

where:

- M_i and M_t: Is the torque acting on the i^{th} mesh element and the total torque acting on the payload, respectively.
- $\mathbf{r}_{pi}$ is the position vector of the i^{th} single surface mesh from the center of mass of the payload.

After the calculation of the force and torque acting on the center of mass of the payload, we can apply this in the MuJoCo environment to simulate the effect created by the flow. This whole process describes one static position in time, thus this process can be repeated iteratively to simulate the transient effects.

6.6 Results

6.6.1 1 rotor flow: Hovering

The visualization in the MuJoCo environment after implementation: (the flow is implemented under the pink rotor)

1. *1 DoF hook attached to the quadrotor*
2. *2 DoF hook attached to the quadrotor*

6.6.2 4 rotor flow: Hovering

The visualization in the MuJoCo environment after implementation:

1. *2 DoF hook, hovering, 10cm by 10cm by 10cm payload*

6.6.3 4 rotor flow: Transportation

Then for robustness and validation we tested the flow implementation on a payload transportation path. During the simulation the payload is $20g$. During the transportation a 2 DoF hook is attached to the quadrotor:

1. *Transportation, without flow, 10cm by 10cm by 10cm payload*
2. *Transportation, with flow, 10cm by 10cm by 10cm payload*

7 Summary

7.1 Conclusion

As a conclusion we can say that the simulation of the 1 rotor flow and the implementation to our MuJoCo environment is a success, but requires validation from experts.

7.2 Future work

The future work can include the validation of the results. One major goal is to use an open source CFD environment such as OpenFoam, where licence and hardware limitations do not occur. Furthermore the refinement of the mesh, the validation of the solution settings and a more comprehensive geometry (eg.: all the 4 rotors at the same time) is a future goal. More complex payload geometry and even a real time force and torque calculation also can be an option.

Appendix

- *Github/AIMotionLab-SZTAKI/CFD-post-process*: Data post-processing. (to create the dynamic pressure look-up tables)
- *Github/AIMotionLab-SZTAKI/Mujoco-Simulator/classes/pressure-sampler.py*: Flow implementation in MuJoCo.
- *SZTAKI Nextcloud*: All data files.
- *Presentation*: Power point presentation of the process.

References

[1] Kristóf Gergely. "Áramlások numerikus modellezése. Electronic book". In: *BME* (2014).

[2] Daxiong Ji et al. "Dynamic modeling of quadrotor AUV using a novel CFD simulation". In: *Ocean Engineering* 237 (2021), p. 109651.

[3] Tamás Lajos. "Az áramlástan alapjai". In: *Műegyetemi Kiadó, Budapest* 600 (2004).

[4] T Oktay and Y Eraslan. "Computational fluid dynamics (Cfd) investigation of a quadrotor UAV propeller". In: *International Conference on Energy, Environment and Storage of Energy (ICEESEN 2020)*. 2020, pp. 1–5.

[5] C Paz et al. "Assessment of the methodology for the CFD simulation of the flight of a quadcopter UAV". In: *Journal of Wind Engineering and Industrial Aerodynamics* 218 (2021), p. 104776.