Universitat Polytechnica de Catalunya
MSc Computational Mechanics
Summer 2018

Industrial Training Report

Technical University of Munich

Due 05/09/2018

Alexander Keiser

Supervisor: Aditya Ghantasala

Contents

1 Introduction

1.1 Initial Study of Kratos Framework and Compilation

In the beginning stages of the internship, I was tasked with learning how Kratos is structured, how the different files interact with one another, and compiling my own version of it on the machine assigned to me at the Technical University of Munich for further use during the internship. Kratos can be run through both GID and the terminal, with the latter providing faster model adjustment and more information during calculation of simulations. The main file that is run by both the terminal and GID is MainKratos.py. This is a python file that executes the simulation, however it needs several other sub-files to perform correctly. These sub-files are the application.py python file, the sample.mdpa model part file, and the ProjectpParameters.json file. The application file tells Kratos which kind of problem is to be solved along with the relevant variable information, element information, and the condition information from other files previously created. The sample.mdpa model part file has all the geometry information (stored as a list of numbered nodes with their x,y,z coordinates and a list of these nodal connectivities for element generation), loads, materials etc. information stored that the main python script reads. This file can be hand coded in very simple cases but usually will be generated by GID once meshing has been complete once the proper boundary conditions, loads, materials and etc. have been applied and the mesh generated thereafter. Next is the ProjectParameters.json file, this file stores all the project parameters such as the type of solver being used, output frequency, time step size, load magnitude, input/output velocities, output pressure, the echo level to get more information during the solving of the simulation, the start/stop times and custom time segments, and which groups all this information is assigned to. This allows the main python script to know which nodes have what conditions assigned to them during solving of the problem. Once I felt that I had a good general understanding of all this, I compiled my own up to date version ofKratos on the Linux machine assigned to me for the internship.

2 Initial Kratos Test Case

Now that I had an up to date version of KRATOS, I had to familiarize myself with using GID to create executable Kratos test cases, running Kratos from the terminal, and modifying the ProjectParameter.json files on the fly for rapid simulation condition modifications and testing. I decided to do a Computational Fluid Dynamics simulation to familiarize myself with the multifaceted capabilities of the Kratos framework.

2.1 T-Pipe CFD Simulation

For the CFD simulation I decided to model a simple T-shaped piping system with time staggered input velocities. A progression of the models and solution results can be found below in figures 1-6f, with a quick summary of the results following after.

Figure 1a: T-Pipe to be Modeled

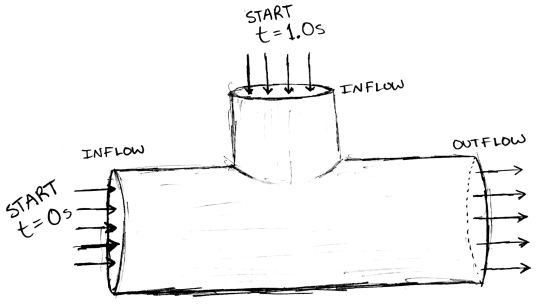


Figure 1b: Initial Idea Sketch

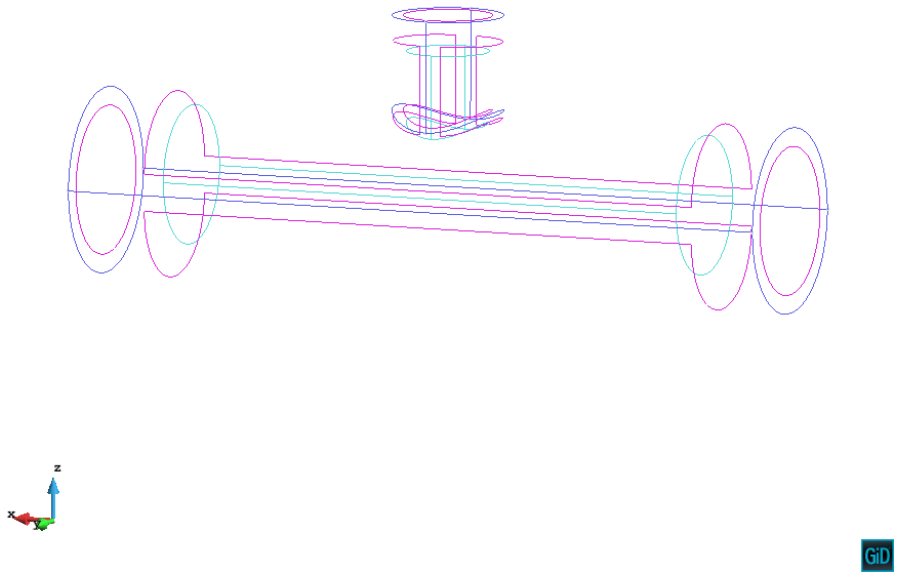


Figure 2: T-pipe GID Geometry

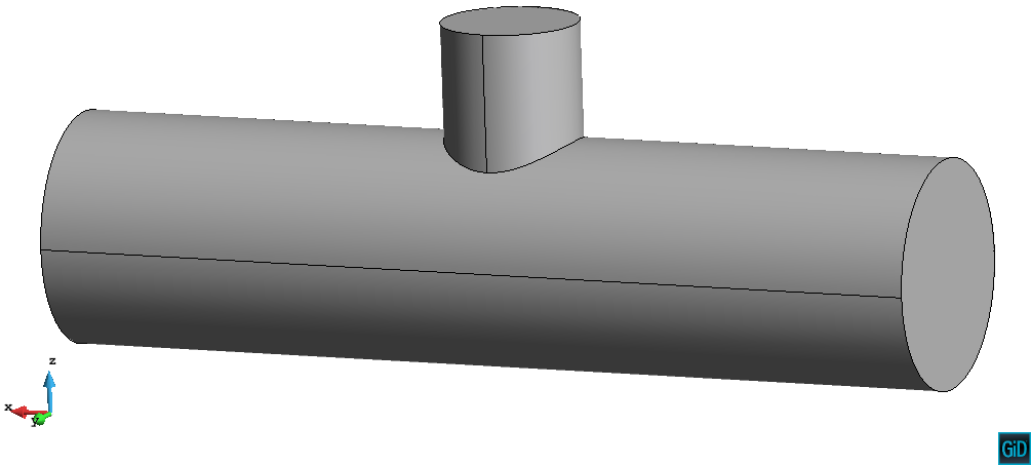


Figure 3: T-pipe GID Model

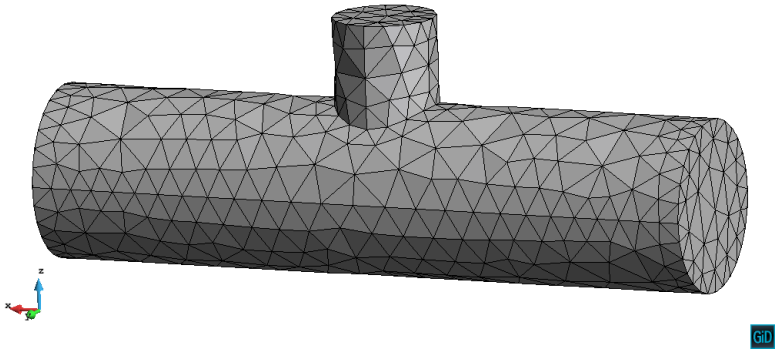


Figure 4: T-pipe GID Model With Mesh

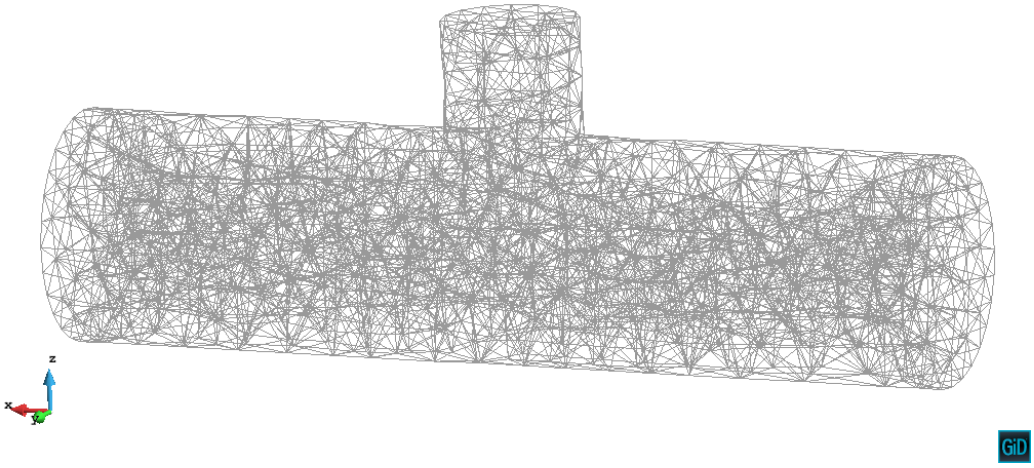


Figure 5: T-pipe Volume Mesh

Figure 6: Preliminary transient solution at various time steps Δt=0.05

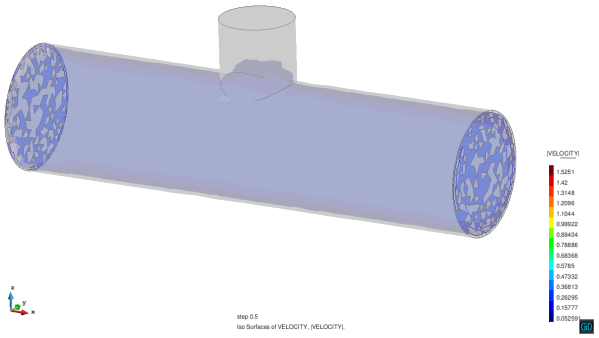


Figure 6a: Time 0.5s

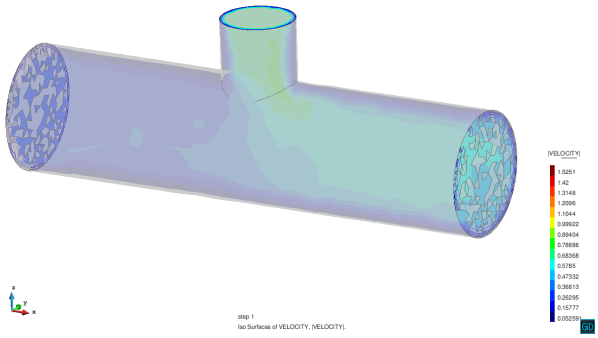


Figure 6b: Time 1.0s

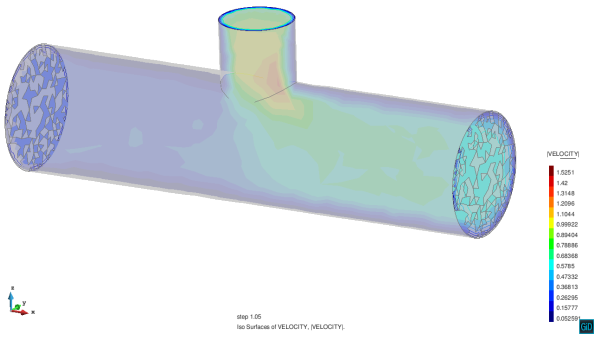


Figure 6c: Time 1.05s

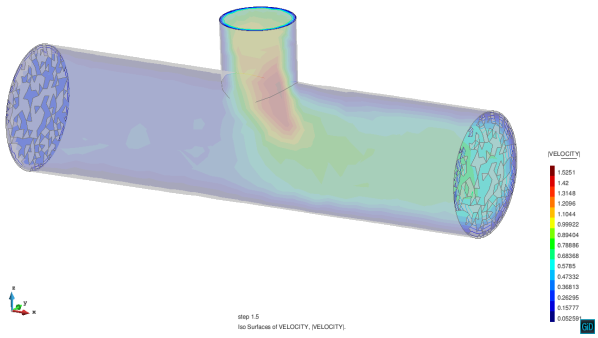


Figure 6d: Time 1.5s

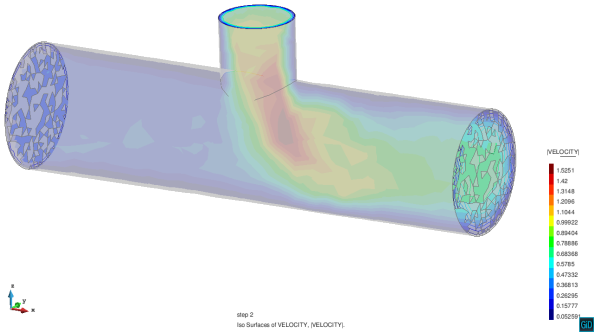


Figure 6e: Time 2.0s

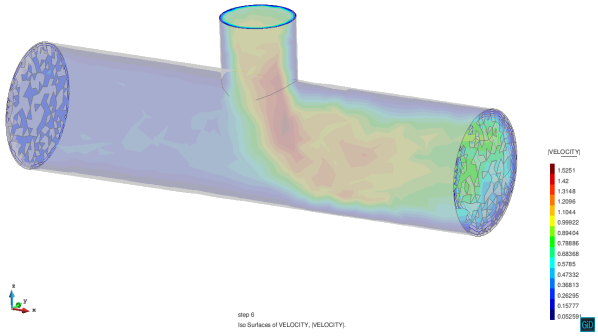


Figure 6f: Time 6.0s

Above we can see the results of the simulation. At time t=1.0s in figure 6b we can see the injection of fluid into the pipe geometry and the resultant increase in outflow velocity compared to that of the velocity profile in figure 6a at time t=0.5s. This increase in velocity is due to the necessary requirement for conservation of mass flow rate. At t=1.0s there is an increase in mass flow rate at the inlets of the system and the outlet is forces into compensating accordingly. This behavior is expected. Additionally, we can notice that the highest magnitude of velocity takes place at, and immediately following, the trailing edge of the injected inlet pipe. It is this region where the horizontal component of the static inlet flow velocity combines with the vertical component of the injected fluid to produce the high velocity magnitude region apparent in the figure.

3 Kratos Chimera Application Test Cases

Now that I was familiar with the GID and Kratos envirenment, The next main task of the internship was the simulation of a wind turbine using the recently developed Chimera domain coupling application within Kratos CFD. This application is capable of coupling surfaces in 2d and volumes in 3d simulations at every time step allowing for movement of smaller meshes and geometry within larger background meshes. This type of application has extensive use cases. However, since since I was still relatively new to the Kratos framework and the Chimera application was still in development, it was decided that the problem should be solved incrementally, meaning that I was to start with the simplest 2d cases of the Chimera application and work towards progressively more complex 3d simulations with every design iteration. The development and progression of this will be shown and discussed below and on the subsequent pages and represents the bulk of the internship work.

3.1 Test Cases 1,2, and 3

The first three test cases would use some of the simplest geometry possible for testing. Below in figures 12, 13 and 14 we can see the mesh and geometry used in these test cases. It was decided to manipulate a square object with no slip conditions applied to its sides within a circular mesh of equal size to the background mesh it was embedded in. This background mesh was rectangular in shape with an inlet velocity on the left side and a gradually increasing mesh size on the right to diffuse the velocity effectively and prevent back flow phenomena from polluting the solution. A detail view of these overlapping meshes can be seen below in figure 13. It is important to specify the terminology that will be used in the subsequent results and discussion. The term "patch" will refer to the small mesh and geometry that is embedded in the larger "background" mesh domain. In figure 14, the "patch" is the meshed area between the circle and the square. The overlaying mesh is part of the background. Additionally, a sample code for the movement of the patch is provided in figure 15. This code corresponds to the sinusoidal movement present in test case 2 in figure 17.

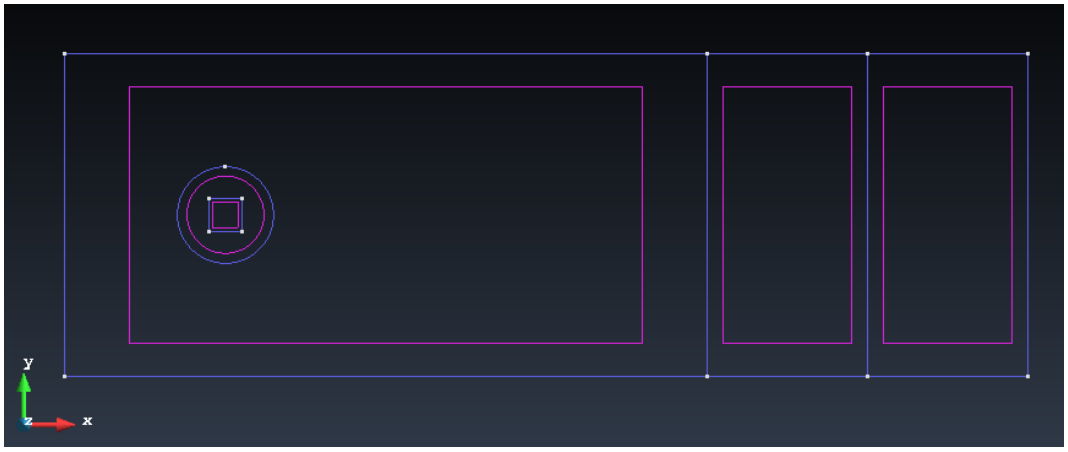


Figure 12: Geometry and Mesh for the first 3 test cases

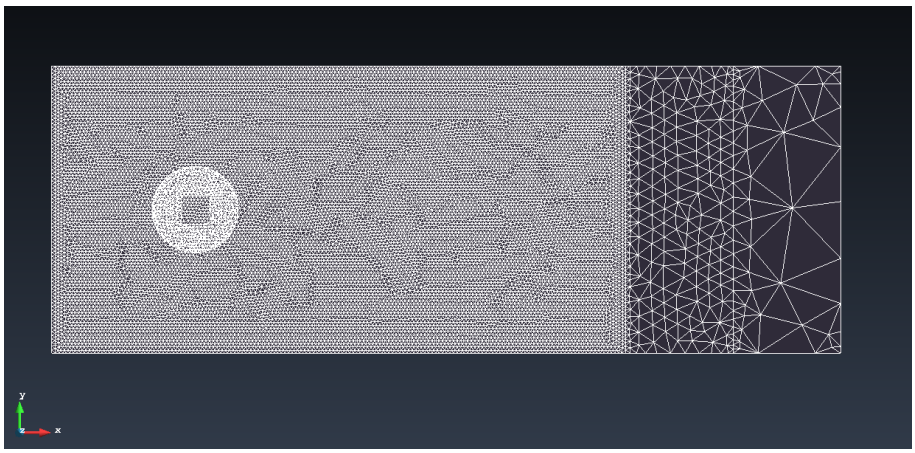


Figure 13: Geometry and Mesh for the first 3 test cases

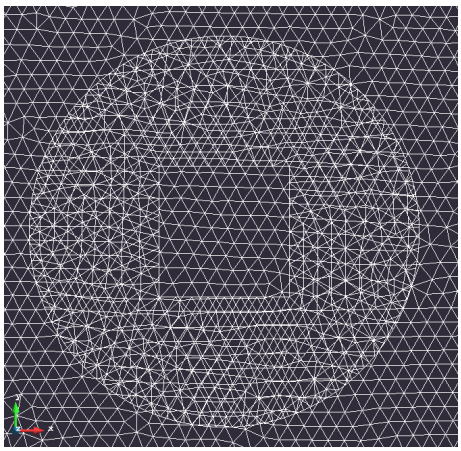


Figure 14: Detail view of overlapping surface meshes for Chimera Domain Coupling application

```
14
15  def Movepatch(patch,time,Dt):
16      vel_x = 1
17      vel_y = 5*cos(time)
18      for node in patch.Nodes:
19          node.X = node.X + 0.025
20          node.Y = node.Y + 5*sin(time) - 5*sin(time - Dt)
21          node.SetSolutionStepValue(KratosMultiphysics.MESH_VELOCITY_X,0,vel_x)
22          node.SetSolutionStepValue(KratosMultiphysics.MESH_VELOCITY_Y,0,vel_y)
23
```

Figure 15: Patch movement code for the second test case

3.1.1 Chimera Test Case 1 Results

The first case had the previously described patch moving in a straight line towards the outlet. Below is a screen shot of the solution at a sample time step with the movement of the patch depicted by an arrow.

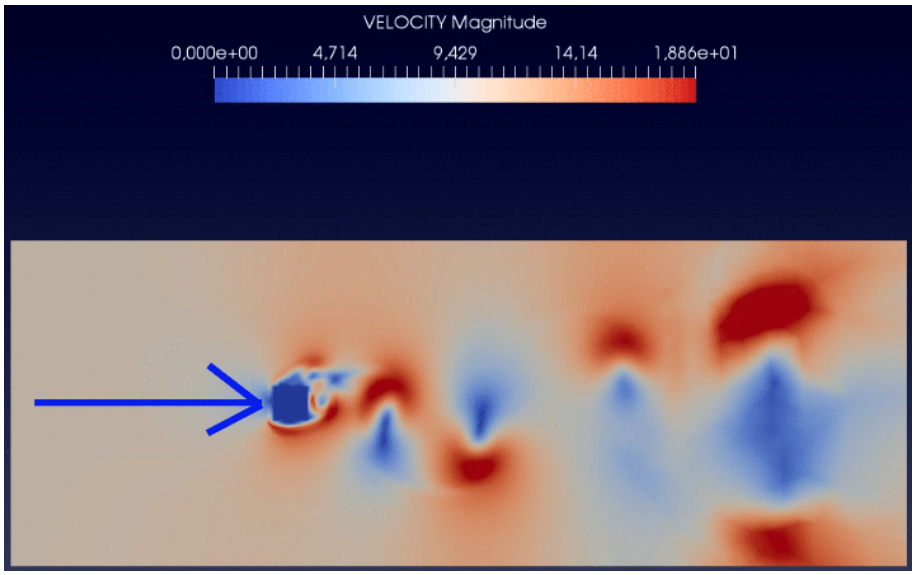


Figure 16: Test Case 1 Screenshot

The test is successful as there is seamless coupling between the patch and the background meshes at every time step.

3.1.2 Chimera Test Case 2 Results

The second case had the patch exhibiting oscillatory behavior towards the outlet. This was done as another preliminary test with oscillatory behavior that built on the previous test adding an extra component of complexity. The code for this behavior can be seen in figure 14, and below is a screen shot of the solution at a sample time step with the movement of the patch also depicted by an arrow.

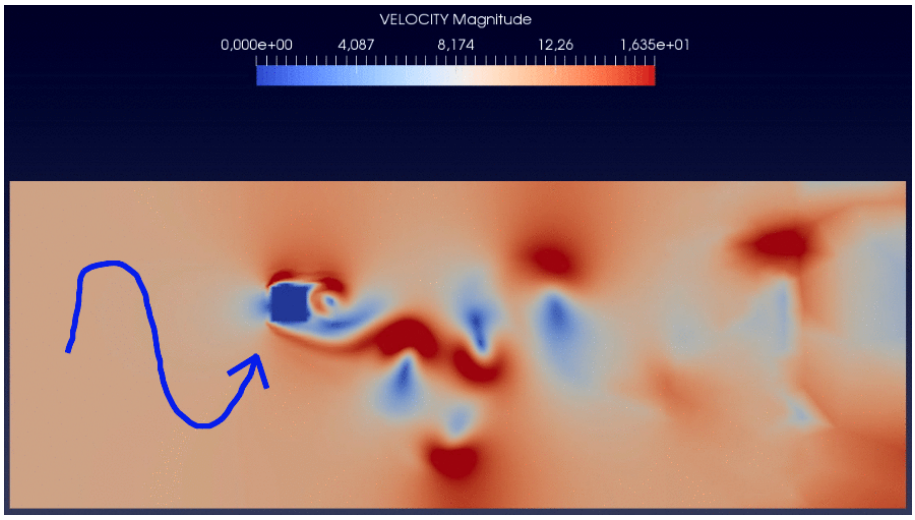


Figure 17: Test Case 2 Screenshot

There was initially some difficulty with the solution as the transition was not seamless and was visibly inaccurate near the boundary of the patch mesh. This was discovered to be due to use of a time step that was too large to successfully accommodate the movement of the patch without transition region errors. Once the time step was lowered the solution became much better.

3.1.3 Chimera Test Case 3 Results

The third case set the patch to spin in place, similarly mimicking the behavior of a wind turbines rotational behavior. While still far away from the goal of a full wind turbine, it was a step in the right direction. This test was the last before scaling up to three dimensions. Below is a screen shot of the solution at a sample time step with the movement of the patch depicted by an arrow.

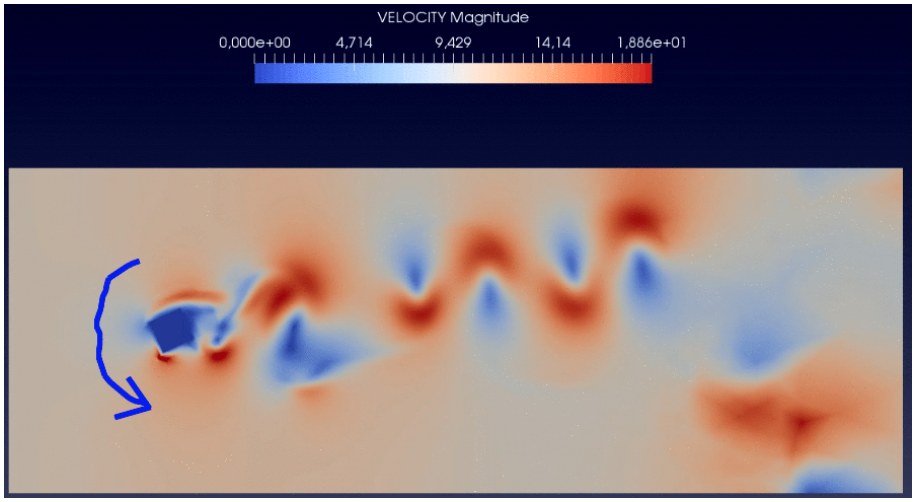


Figure 18: Test Case 3 Screenshot

Similarly to the previous test, there were initial patch boundary errors due to too large of a time step being chosen to successfully accommodate the speed of the rotational behavior of the patch. Once again, after lowering the value time step, the solution became much smoother. The patch successfully couples with the background at each time step. The application is behaving correctly in two dimensions.

3.2 Chimera Test Case 4 (3D Rotating Rectangular Prism)

We will now extend the test cases to the third dimension. In order to keep the problem as simple as possible, it was decided to use a simple rectangular prism as the rotating object inside a larger rectangular prism. The patch in this case is three dimensional and is the volume between the larger rectangular prism and the smaller rectangular prism. This can be clearly seen in figure 20 below. One can also notice a cylinder encompassing these two rectangular prisms. This cylinder will not be moving, but is used as a volume with which to specify a finer mesh size closer to the patch movement. This "refined volume" strategy not only allows us to better capture the behavior of interest, but also saves computational cost. This strategy will also be used in future simulations later on in this report. So in essence, for this simulation we can imagine the the patch formed between these two rectangular prisms simultaneously rotating within this cylindrical region of specified finer mesh size. The geometry, rendered model, mesh, and results can be found below and on the following pages in figures 19-22. And a brief discussion of the results ban be found following figure 22.

Figure 19a: Patch Geometry

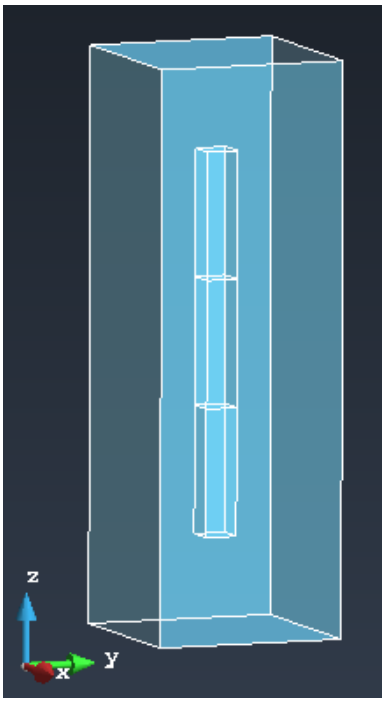


Figure 19b: Rendered Patch

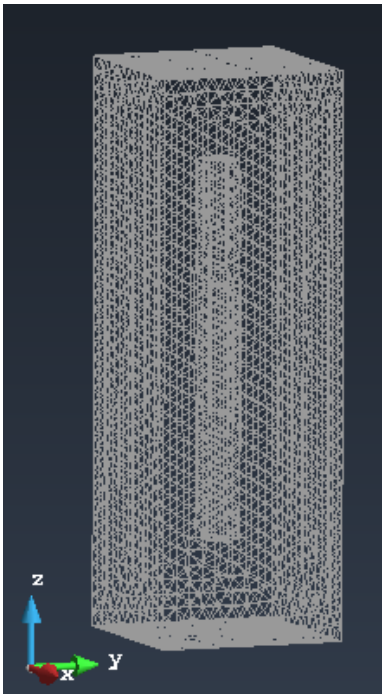


Figure 19c: Patch Boundary Mesh

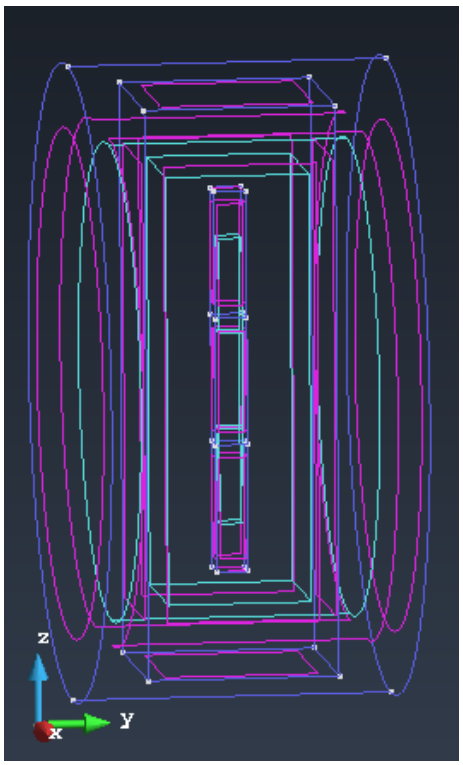


Figure 19d: Patch and Cylinder Geometry

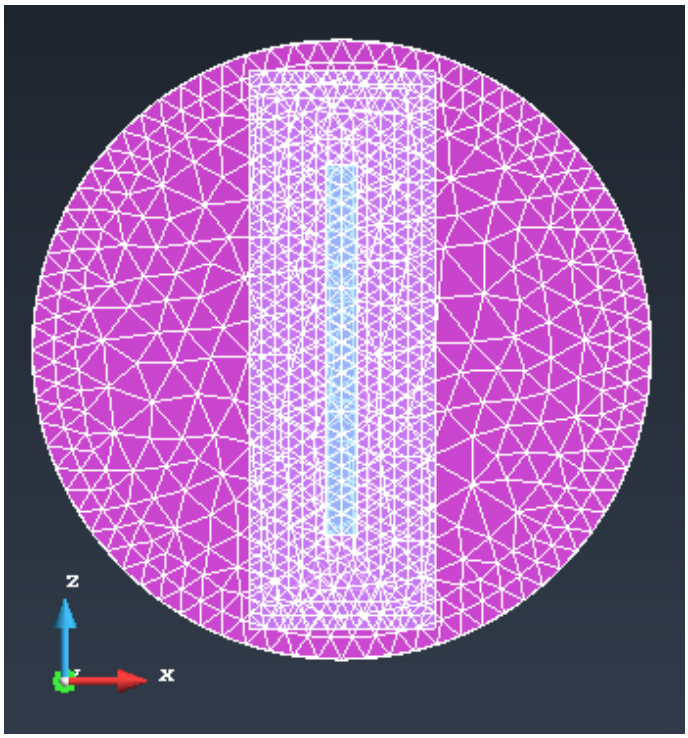


Figure 19e: Patch and Cylinder Surface mesh

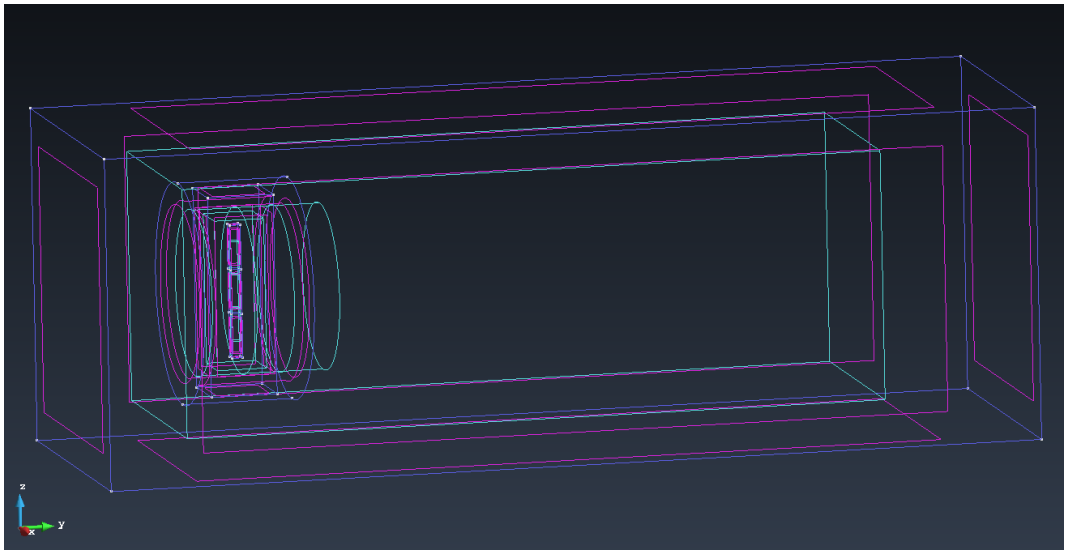


Figure 19f: Test Case 4 Geometry

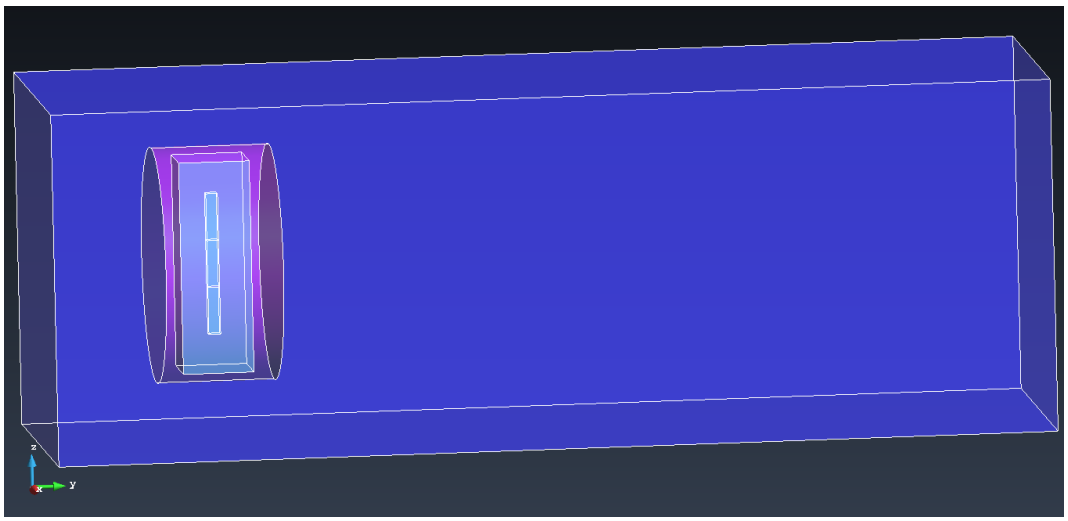


Figure 20: Test Case 4 Rendered Geometry

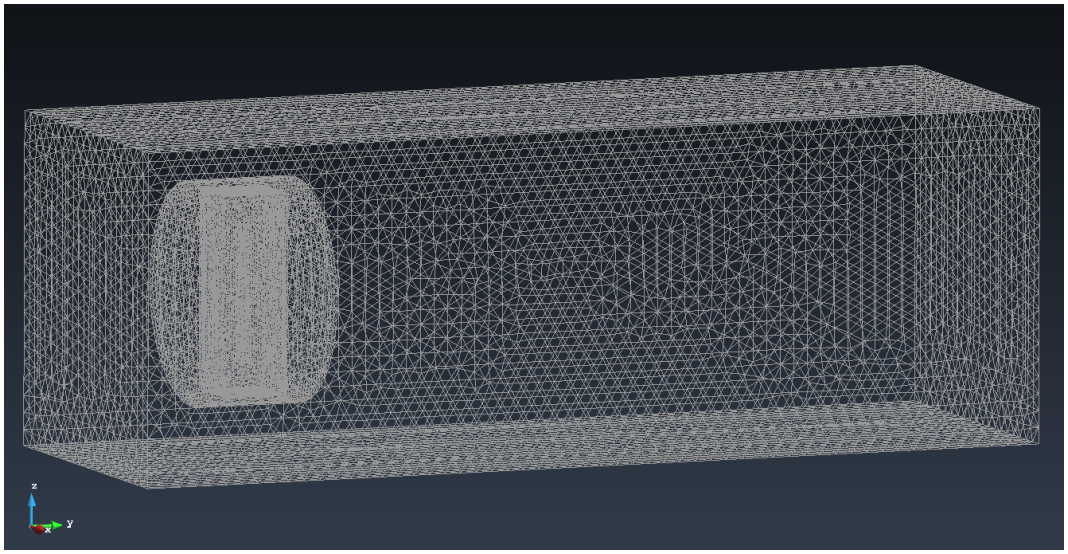


Figure 21: Test Case 4 Mesh Boundary

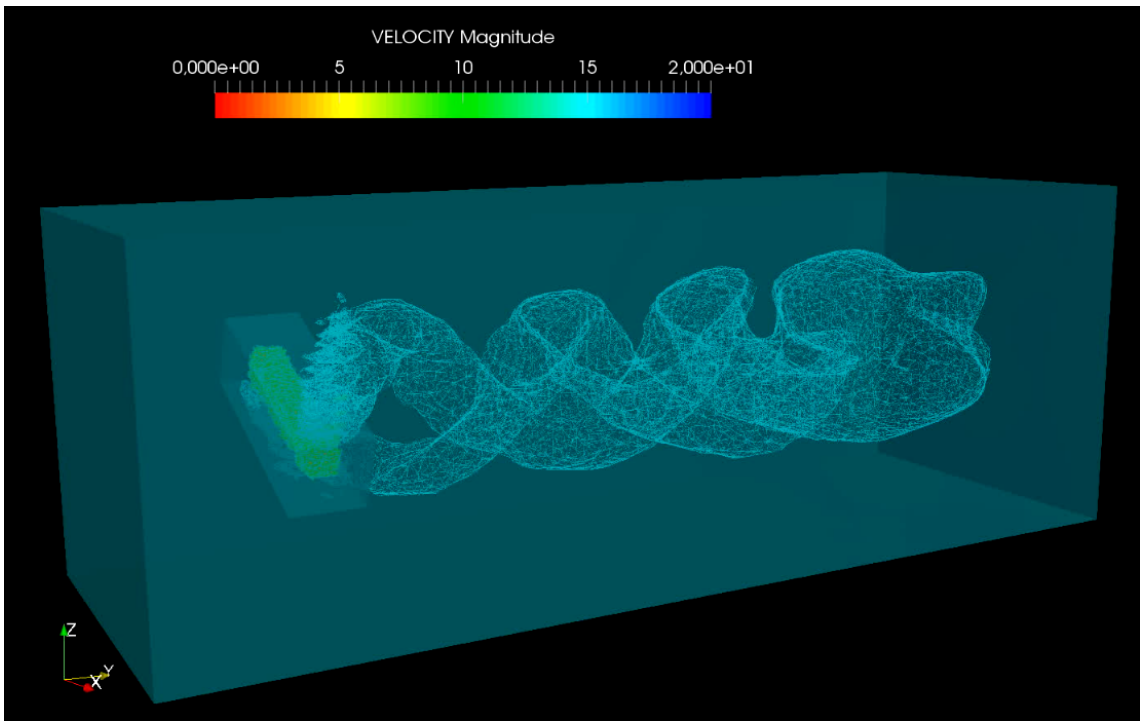


Figure 22: "Rotating Rectangular Prism" Test Case Screenshot

Above we can see a screen shot of the results of the test. Initially, the problem was solved with only the rotating geometry to verify that the patch was moving correctly before turning on the fluid solver. The previous experience experimenting with time step sizes for the sinusoidal 2D cases helped me to correctly choose an appropriate time step size and rotational velocity for this three dimensional case. The results were excellent, the patch was successfully able to couple with the background at every time step. This is confirmed by the continuous behavior at the patch boundary. The isometric velocity magnitude surfaces downstream from the fluid represent the slowed flow resulting from interaction with the rotating rectangular prism. This behavior was expected and was a huge step in the right direction towards the wind turbine goal.

3.3 Chimera Test Case 5 (3D Rotating "Block Blades")

Now that I had a successful basic 3D test completed. It was time to perform a similar test but with more complicated geometry to test the robustness of the coupling mechanism before moving on to turbine geometry that would be even more complicated. I decided to model three-dimensional rectangular blocks consistent with the three pronged shape of the blades that would be used in later test cases. The outer patch boundary was decided to be modeled consistently with this shape as well. And as with the previous case, a cylindrical volume with a finer specified mesh size was employed around the patch to better capture the important behavior. The geometry, rendered model, mesh, and results can be found below and on the following pages in figures 23-28. And a brief discussion of the results ban be found following figure 28.

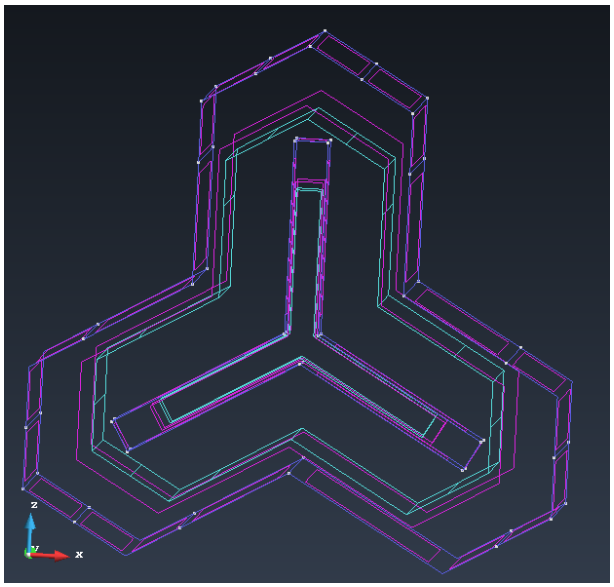


Figure 23a: Patch Geometry

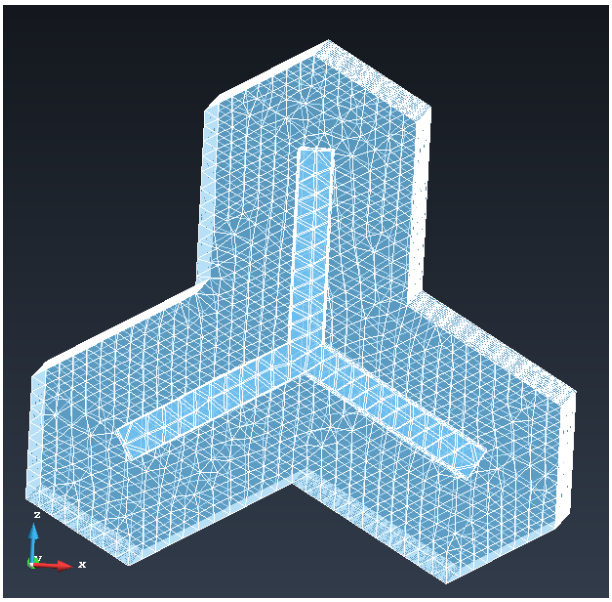
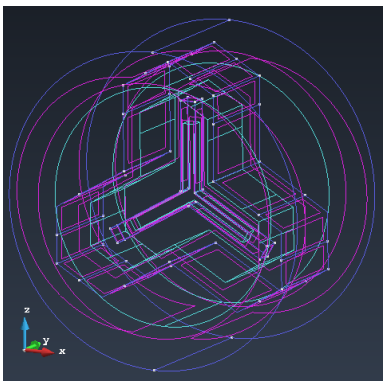
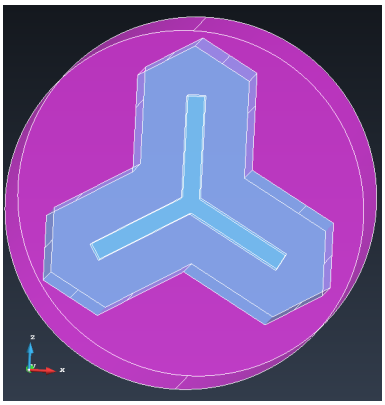


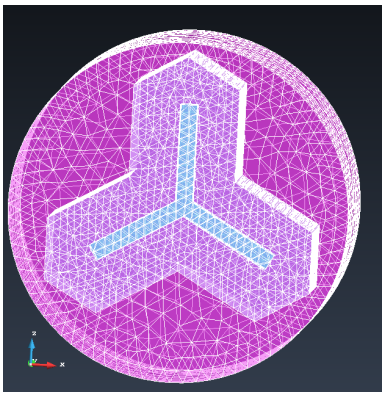
Figure 23b: Patch Surface mesh



Patch and Cylinder Geometry



Rendered Patch and Cylinder



Patch and Cylinder Boundary Mesh

Figure 24: Patch and Cylinder Views

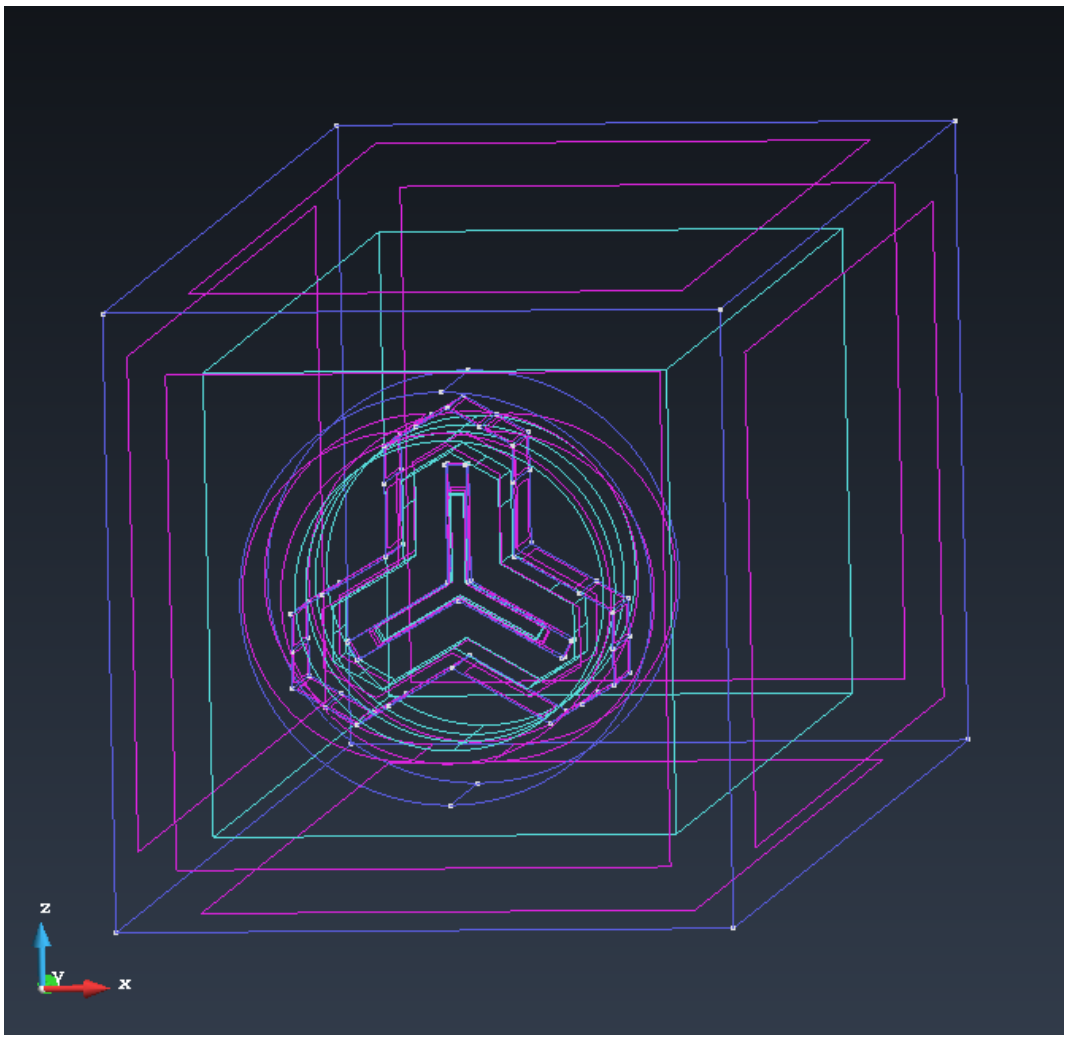


Figure 25: Test Case 5 Geometry

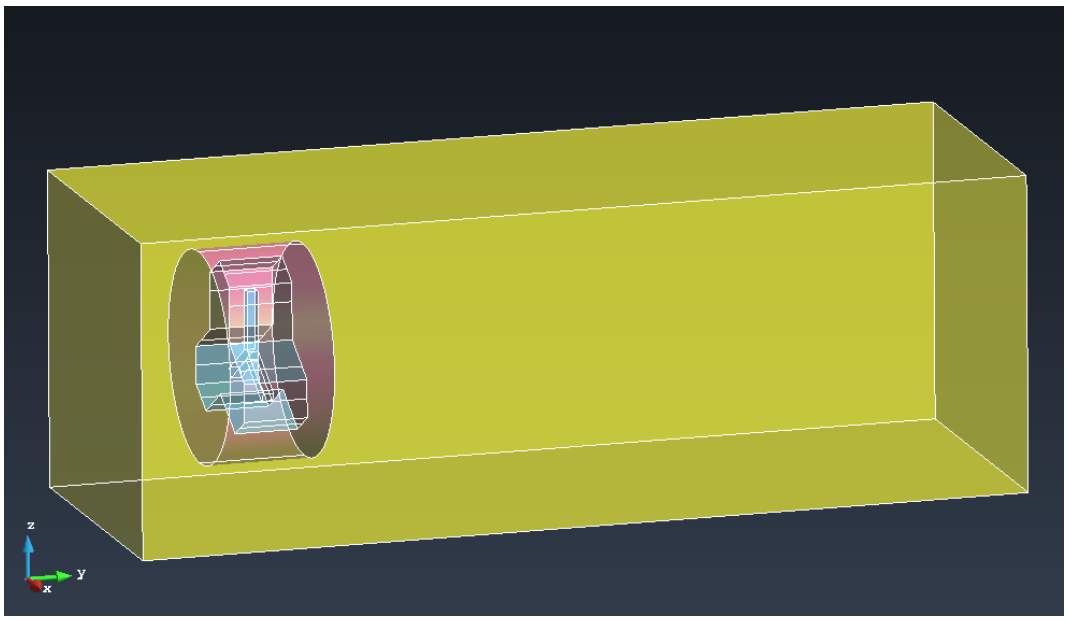


Figure 26: Test Case 5 Rendered Geometry

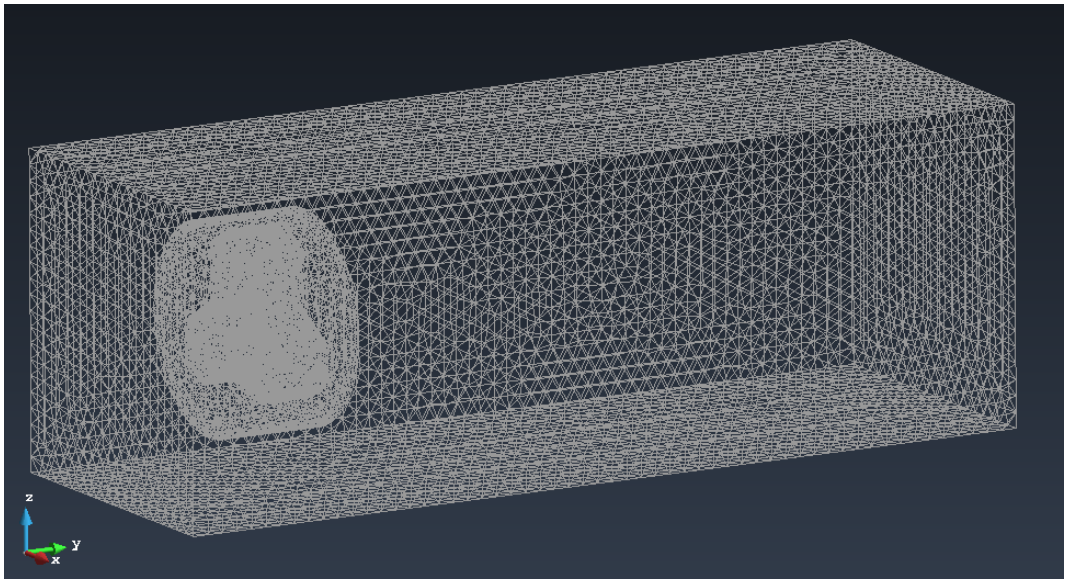


Figure 27: Test Case 5 Mesh Boundary

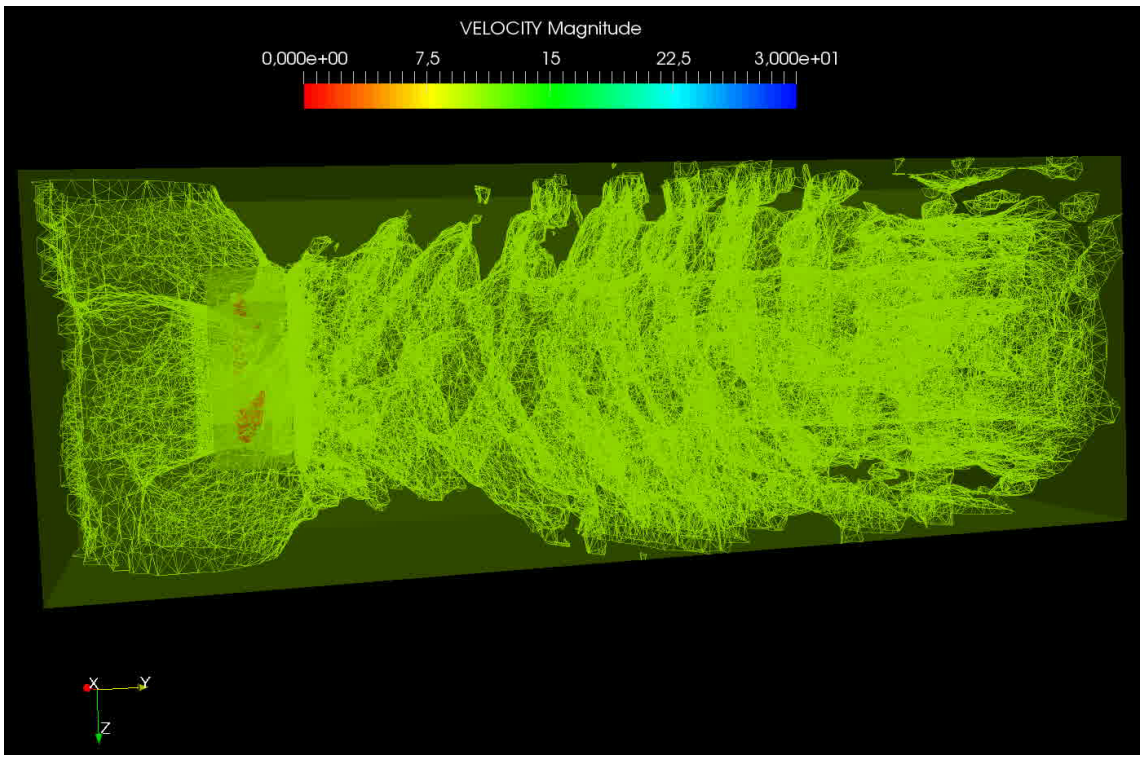


Figure 28: "Block Blades" Test Case Screenshot

Above we can see a screen shot of the results of the test previously described. These results also came out very good. A slower fluid inlet velocity was used here compared to the previous case and as a result we can see many more slowed rotations in the downstream fluid field. There is also continuous behavior at the patch boundary, indicating that the time step chosen was of appropriate size to accommodate the rotation of the blades and the background fluid velocity. Additionally, the chimera coupling application is able to handle the more complex patch geometry. This was another great step in the progression towards the wind turbine model, and was a good indicator that the application should be able to handle the complicated patch geometry of the wind turbine blades

4 Chimera Wind Turbine Test Case

4.1 Wind Turbine Test Case 1 (Simplified Turbine Geometry)

Now that some successful 3D cases had been achieved, it was time to model and extend the tests to a simplified turbine model. An iges file of a simple wind turbine was used here, but a significant amount of effort had to be put into taking this iges file and successfully creating and prepossessing appropriate volumes, surfaces, patch geometry, and mesh details within the GID interface. Below and on the next pages, the products of this development, progression, and details of this case can be seen, along with the test case results. It is worth noting the shape of the patch used here. In figures 31 and 32 the patch for this case can be seen. It is the volume in between the blades and outer boundary. One can notice that the patch width is thicker near the turbine hub and that there is more patch volume near the trailing edge of each blade. These choices were made after extensive trial and error of various patch geometries. It is also worth noting the relative sizes of the meshes being used in these cases. In this case the mesh had around 800,000 nodes, which is 8 times larger than the previous 3D cases, each with around 100,000 nodes. We can notice that the mesh on the blades in figure 30 is coarse, this was acceptable because this was just a preliminary test. More complex and finer meshes would be employed if this was successful. The resulting aerodynamics were not deemed important, all that mattered was that the Chimera application was functioning correctly and that the case was solvable.

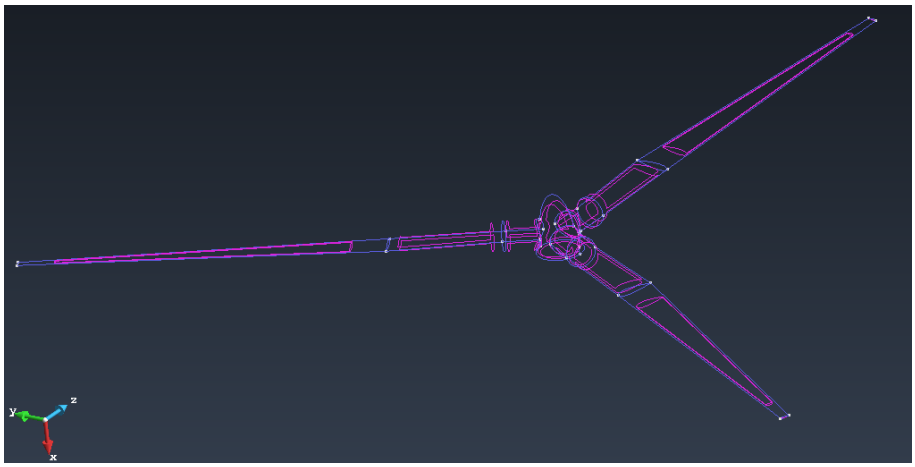


Figure 29: Simplified Turbine Surface Geometry

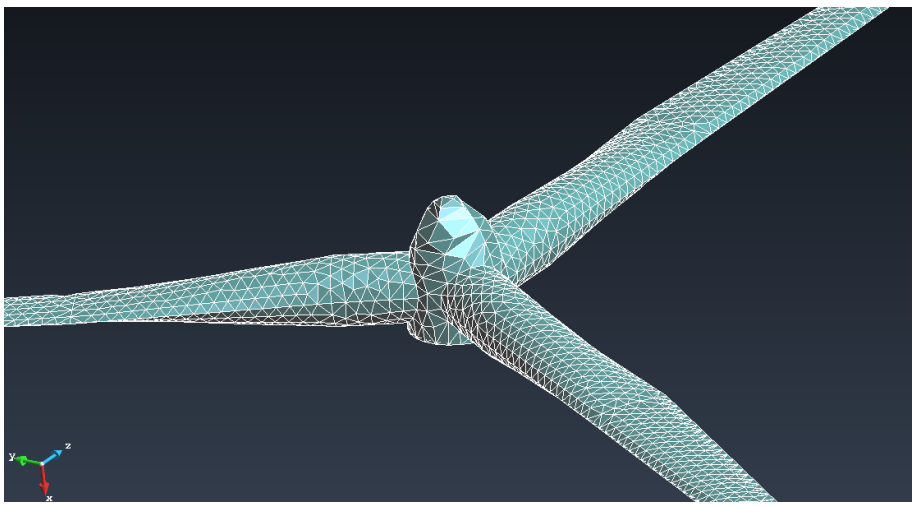


Figure 30: Simplified Turbine Rendered Surface mesh

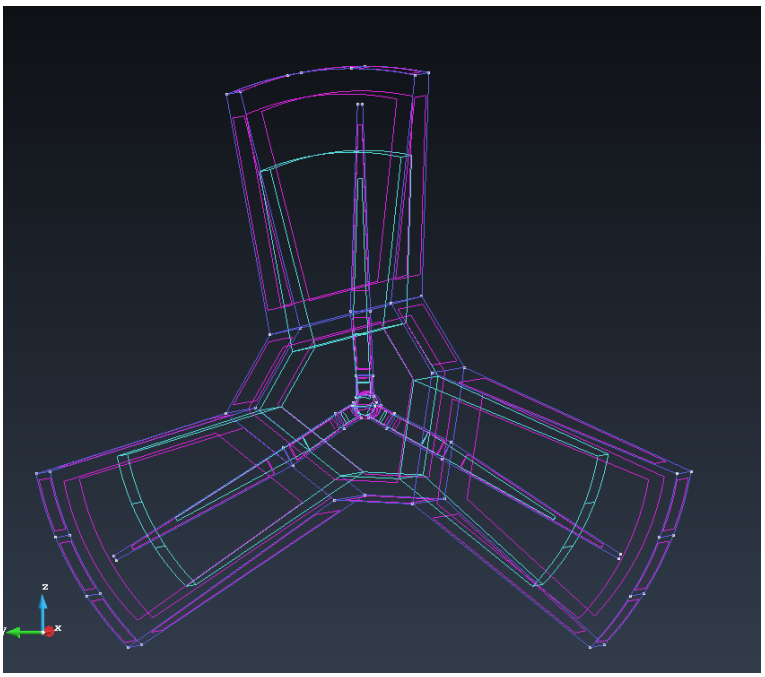


Figure 31: Simplified Turbine Patch Geometry

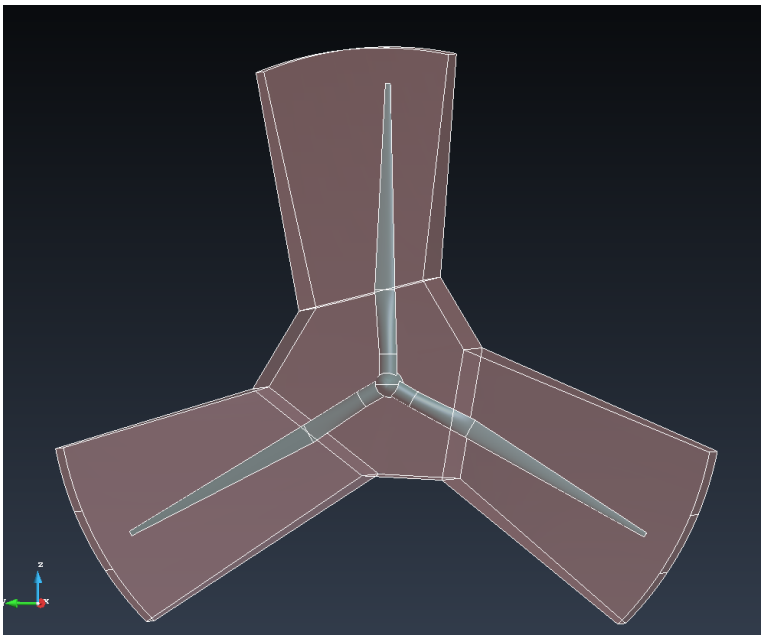


Figure 32: Simplified Turbine Rendered Patch Geometry

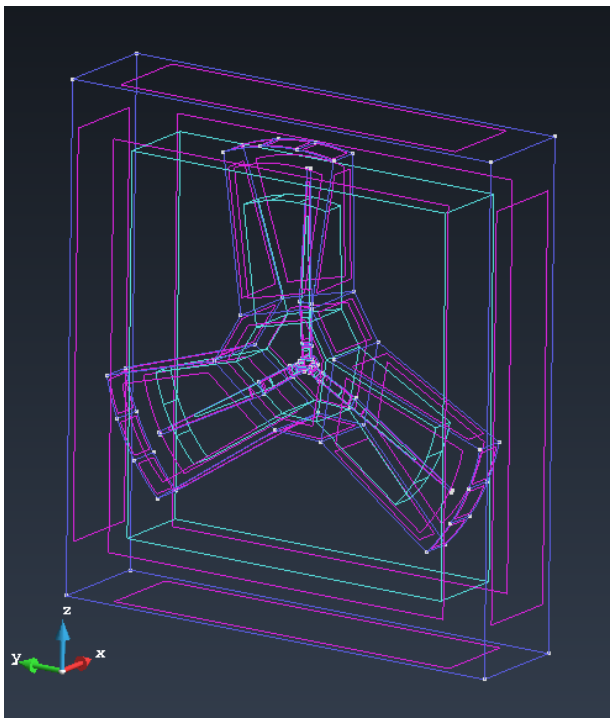


Figure 33: Patch In First Refined Volume

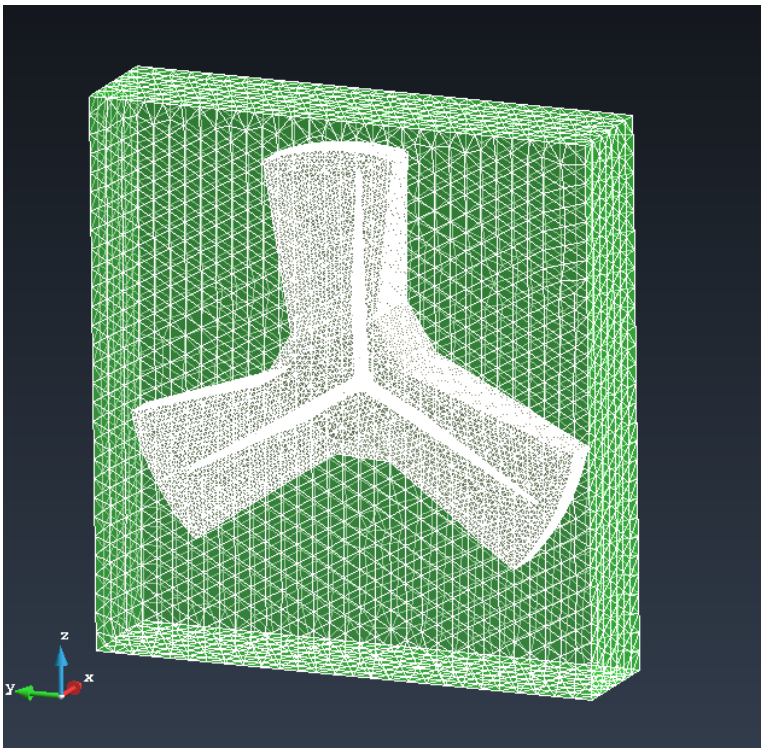


Figure 34: Boundary Mesh Of Patch In First Refined Volume

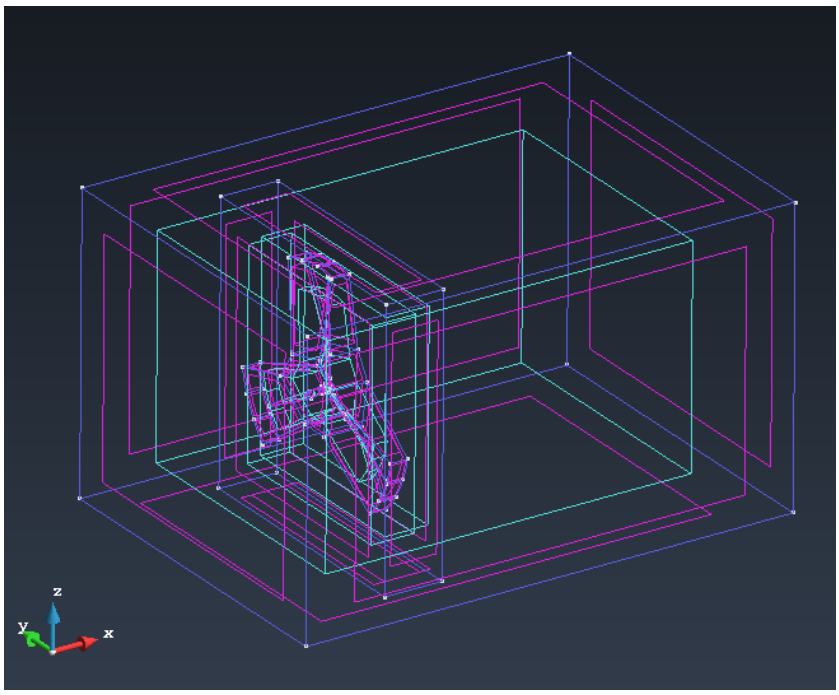


Figure 35: Total Geometry

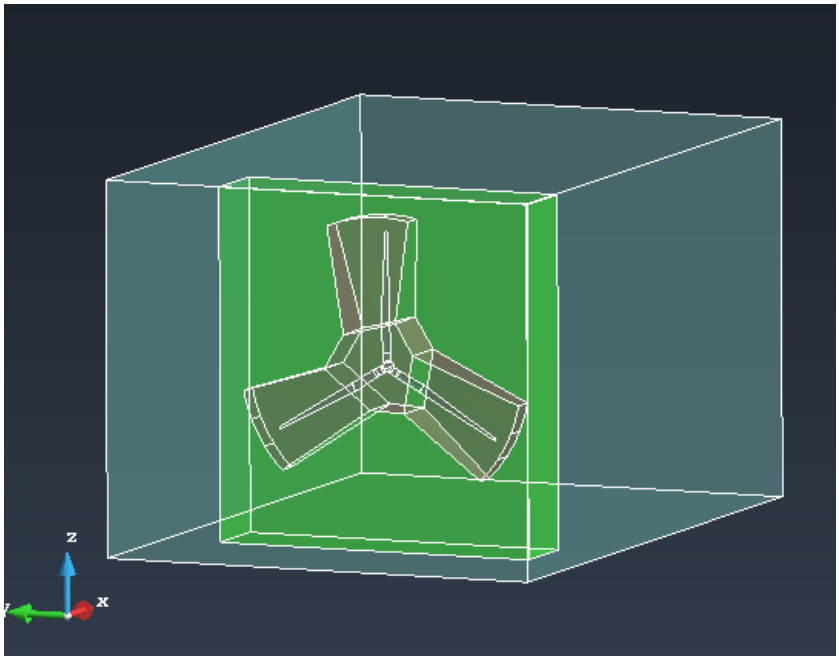


Figure 36: Total Model

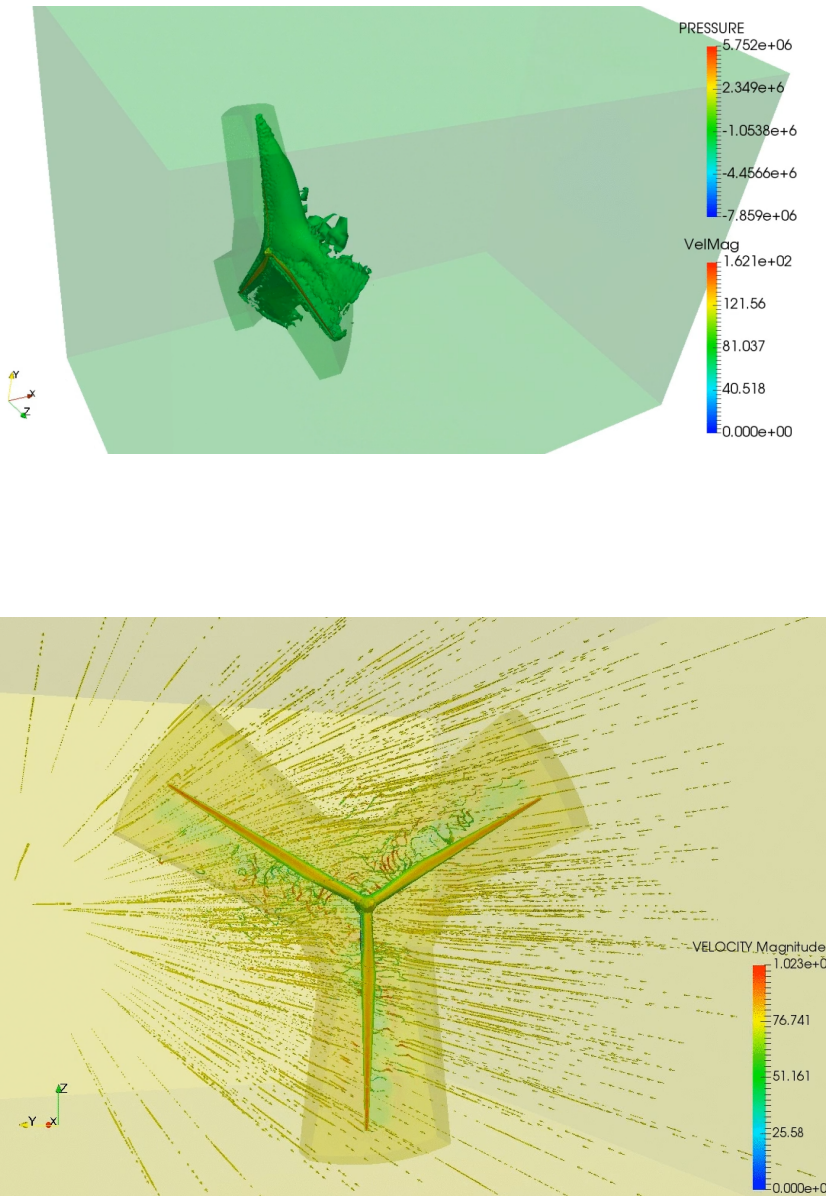


Figure 37: Simple Turbine Results: Contour (top) and Streamline (bottom)

Above in figure 37 we can see some results from this simulation. The test case was a great success and the Chimera application was able to solve the case. Although the specific blade aerodynamics are not accurate, we can comment on the general overall shape of the resultant fluid behavior. In the top part of the figure we can notice pressure successfully being calculated on the blades and the resulting isometric surfaces of the velocity magnitude of the fluid following the blades. The slowed fluid follows the movement of the blades as is expected and the transition of the behavior from the patch to the background is smooth. In the bottom portion of figure 37 we can see a streamline analysis and the resultant turbulence from the fluid's interaction with the blade. This test was another big success and step in the right direction.

5 Future Work

By the time the first wind turbine case was done, around 6 of the 8 weeks of the internship were complete. However, while the previous case was being tested and simulated, additional models for further testing and simulation were developed. These will be included and explained in the following sections for future work and testing.

5.1 Wind Turbine Test Case 2 (Complex Turbine Geometry w/ Torque)

Now that a successful basic wind turbine case had been created, it was time to significantly refine the model and create a case that would approach real world accuracy. A much more complicated iges model was preprocessed for this case and a much finer mesh was employed. Additionally, a extra layer of refinement was included as well as a larger overall domain. It is important to emphasize how fine of a mesh was used here. Over 1.8 million nodes were used and below in figure 38 we can see just how fine this mesh is when compared to the meshes employed in previous cases. Additionally, a new patch shape was developed to better capture the behavior of the fluid immediately surrounding the blades. In addition to this, a small piece of code was written to calculate the torque around the rotation axis, and then applied the resulting angular acceleration to the patch at each time step to simulate a turbine starting to spin due to the torque produced from the wind hitting the blades. A significant amount of effort was put into preparing the volumes, surfaces, and mesh for the geometry used in this case as well. Below and on the next pages we can see the products of the development of this test case.

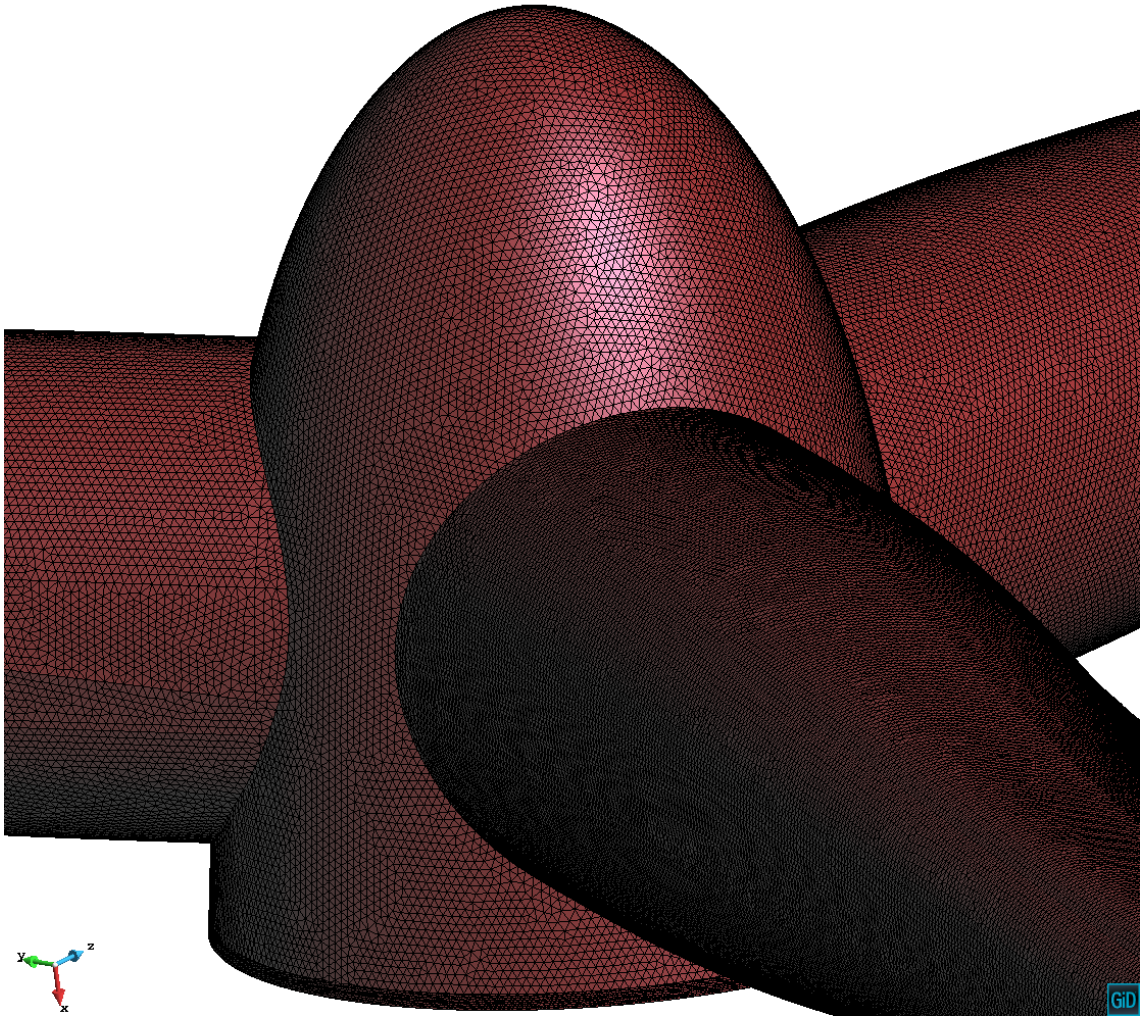


Figure 38: Complex Turbine Geometry Hub With Refined mesh

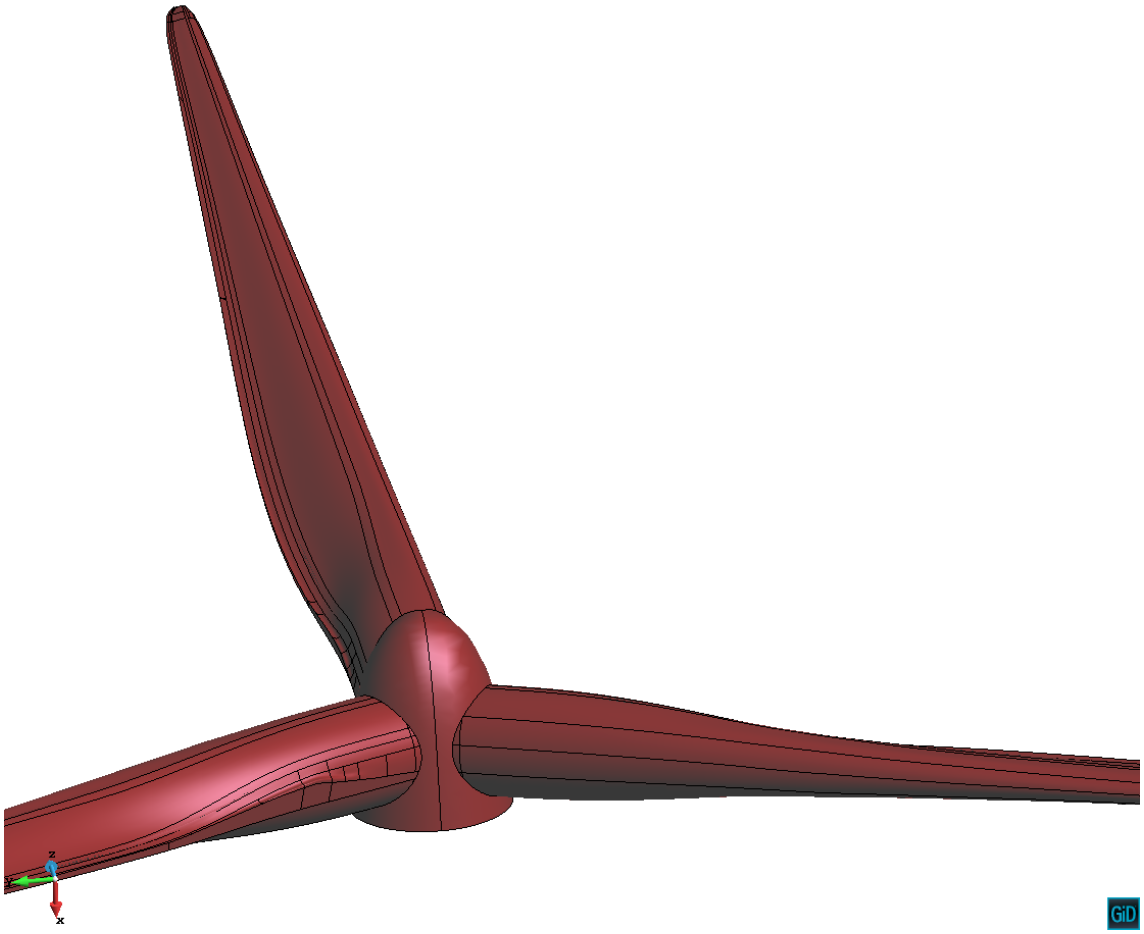


Figure 39: Complex Turbine Geometry Rendered View

```python

def CalculateTorque(model_part, first_point, second_point):
    ##AXIS OF ROTATION INFORMATION
    #FIRST POINT
    X1 = first_point[0]
    Y1 = first_point[1]
    Z1 = first_point[2]
    #SECOND POINT
    X2 = second_point[0]
    Y2 = second_point[1]
    Z2 = second_point[2]
    #LINE VECTOR
    L = X2 - X1
    M = Y2 - Y1
    N = Z2 - Z1

    #DECOMPOSITION OF UNIT VECTOR ALONG LINE
    U1 = L / sqrt( L**2 + M**2 + N**2 )
    U2 = M / sqrt( L**2 + M**2 + N**2 )
    U3 = N / sqrt( L**2 + M**2 + N**2 )

    #INITIALIZE TORQUE SUM
    SIGMA_TORQUE = 0

    for node in model_part.Nodes:
        #COORDINATES OF CURRENT NODE
        PX = node.X
        PY = node.Y
        PZ = node.Z

        #SUBTRACTION OF NODE COORDINATES FROM FIRST POINT
        DPX = PX - X1
        DPY = PY - Y1
        DPZ = PZ - Z1

        #CARTESIAN FORCES OF CURRENT NODE
        FX = -node.GetSolutionStepValue(KratosMultiphysics.REACTION_X,0)
        FY = -node.GetSolutionStepValue(KratosMultiphysics.REACTION_Y,0)
        FZ = -node.GetSolutionStepValue(KratosMultiphysics.REACTION_Z,0)

        #CALCULATION OF TORQUE
        T1 = U1 * ( DPY*FZ - DPZ*FY )
        T2 = U2 * ( DPX*FZ - DPZ*FX )
        T3 = U3 * ( DPX*FY - DPY*FX )

        #SUM OF POINT TORQUES
        T = T1 + T2 + T3

        #CUMULATIVE SUM FOR THIS TIME STEP
        SIGMA_TORQUE = SIGMA_TORQUE + T

    return SIGMA_TORQUE

def CalculateAngularVelocity(Model_part, OmegaPrevious, I, Dt, Torque, First_point, Second_point):
    Torque = CalculateTorque(Model_part, First_point, Second_point)
    OmegaNew = OmegaPrevious + ( (Dt*Torque) / I )

    return OmegaNew
```

Figure 40: Torque and Angular Acceleration Calculator

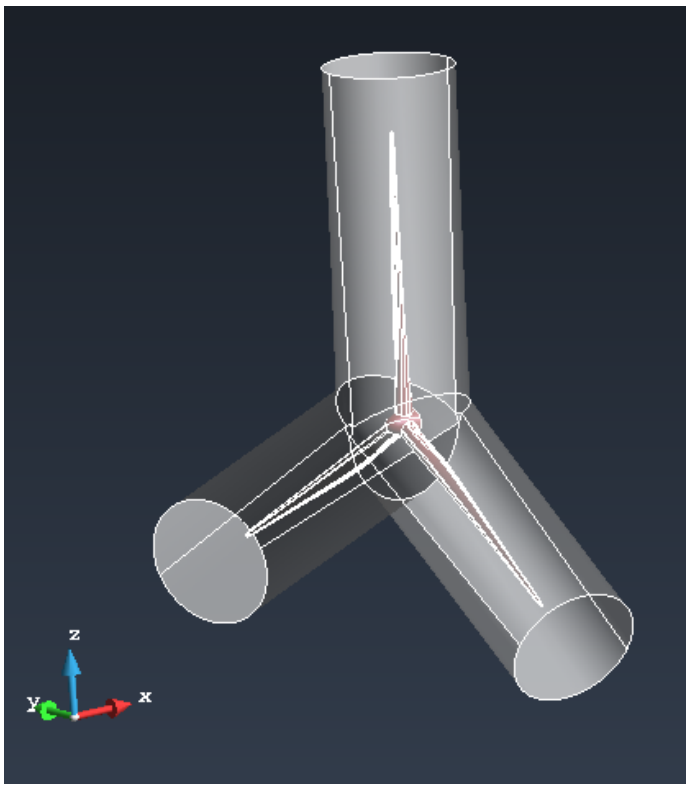


Figure 41: Complex Blades In Updated Cylindrical Patch Geometry

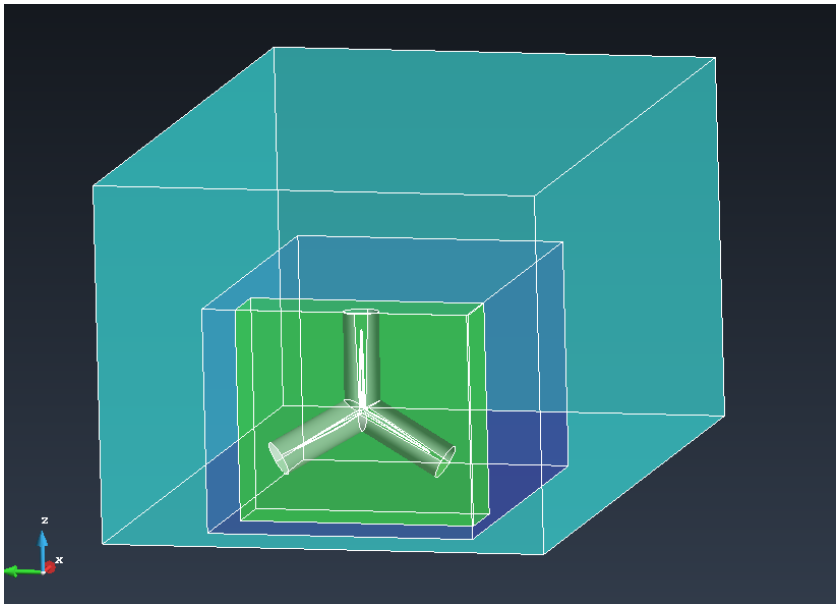


Figure 42: Total Updated Rendered Model

The result from this test is unfortunately not available at this moment as it was blowing up after several time steps. Further testing and refinement must be conducted to get this case working.

5.2 Wind Turbine Test Case 3 (Complex Turbine/Pole Geometry w/ Torque)

The next test case that was developed was essentially the same as the previous test case but with the addition of the pole geometry. This addition of the pole is an important step towards simulating real world turbines as there are a lot of interesting fluid behaviors taking place around the pole and close to the central hub of the turbine. However, one of the biggest challenges with the inclusion of this geometry is that the patch volume overlaps with the pole, and these nodes are now out of the bounds of the simulation and must be deactivated at each time step. The current state of the chimera application was not prepared for this at the time so the simulation was not tested. However, in time and in future development, the application should be able to handle this kind of simulation, so it was developed anyways. Below we can see the simulation geometry developed for this case.

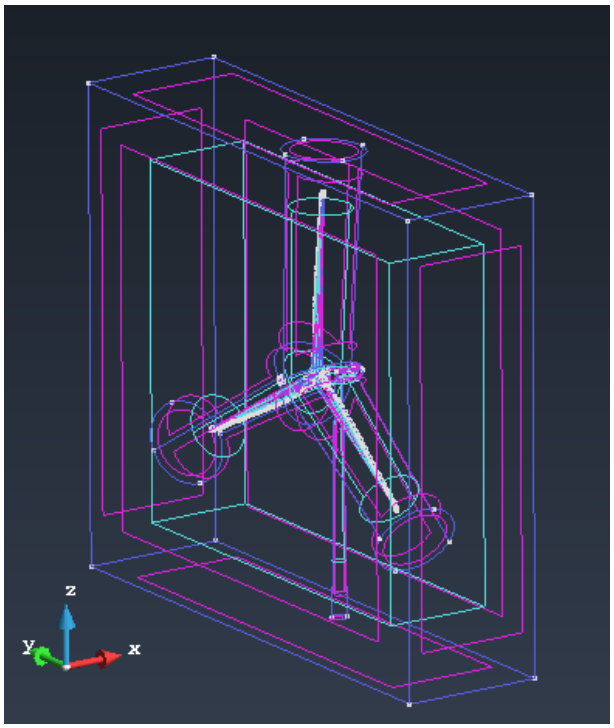


Figure 43a: Patch and Cylinder Geometry

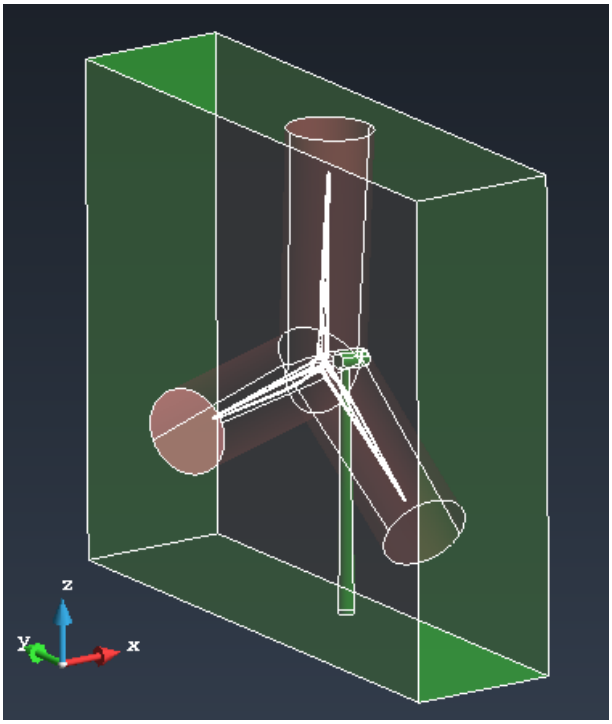


Figure 43b: Patch and Cylinder Surface mesh

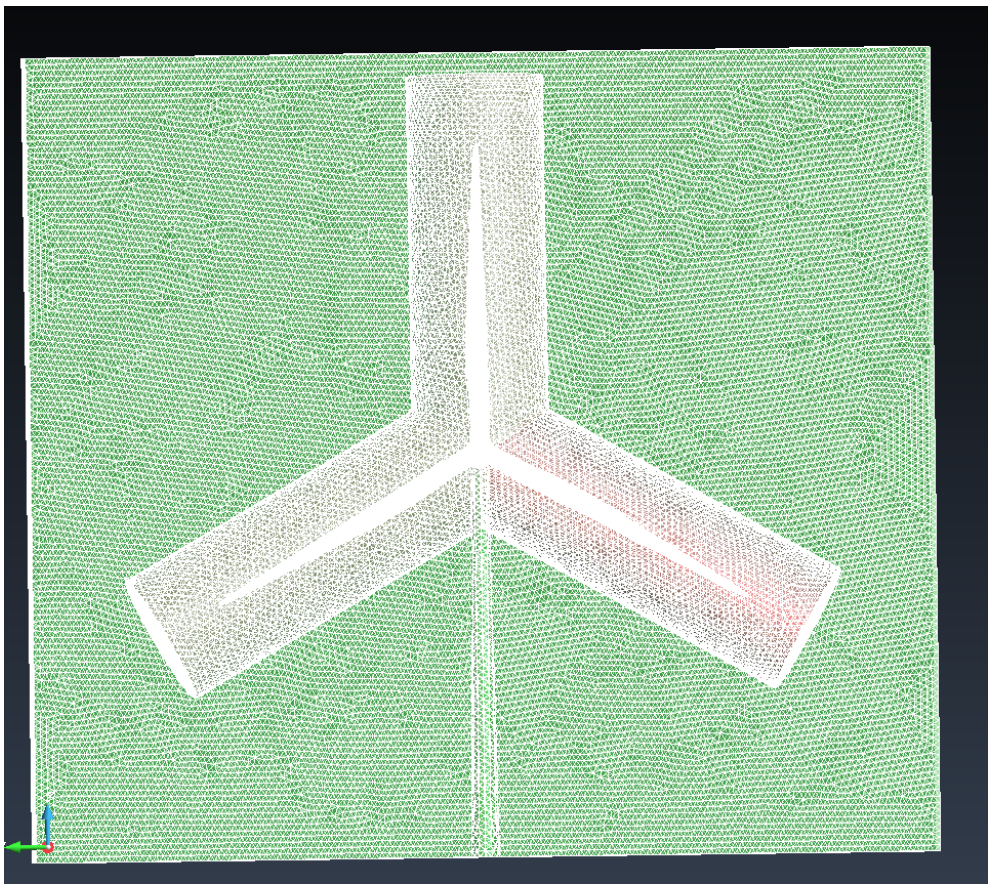


Figure 44: Total Geometry

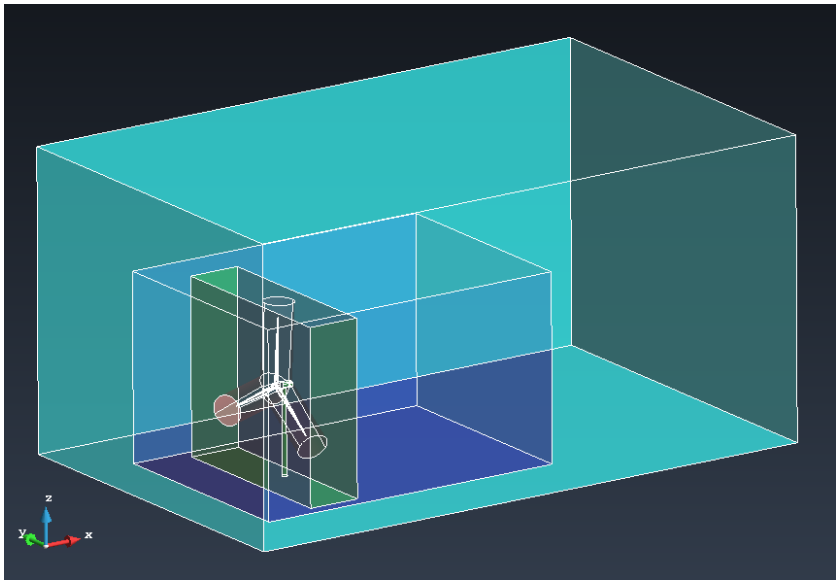


Figure 45: Total Model

As stated previously, there are currently no results for this simulation. With future testing and development of the application. This kind of simulation should be possible.

5.3 Chimera Quadcopter Test Case

Now that the work on the wind turbine models was for the most part complete, it was decided to extend the investigation and test of the chimera application to a case with multiple moving patches. A useful case of this is that of a quadcopter with 4 spinning blades, with each diagonal set moving counter clockwise and clockwise respectively. Below we can see the progression of the test case and model to be developed. The initial geometry was modeled in the modeling software NX. It was saved as a iges file and imported to GID for preprocessing. Quite a bit of work had to be done in the preprocessing stage as the imported iges file was severely broken. What was left of the iges file was turned into a volume tight surface model with 4 blades each serving as an individual patch. Additionally, the volume refinement strategy previously employed was used here as well. This can be seen in figure 47 as 4 cylinders encompassing each of the 4 blades. This model was by far the largest, with over 5 million nodes required for computation. A progression of the model can be found below and on the following pages.

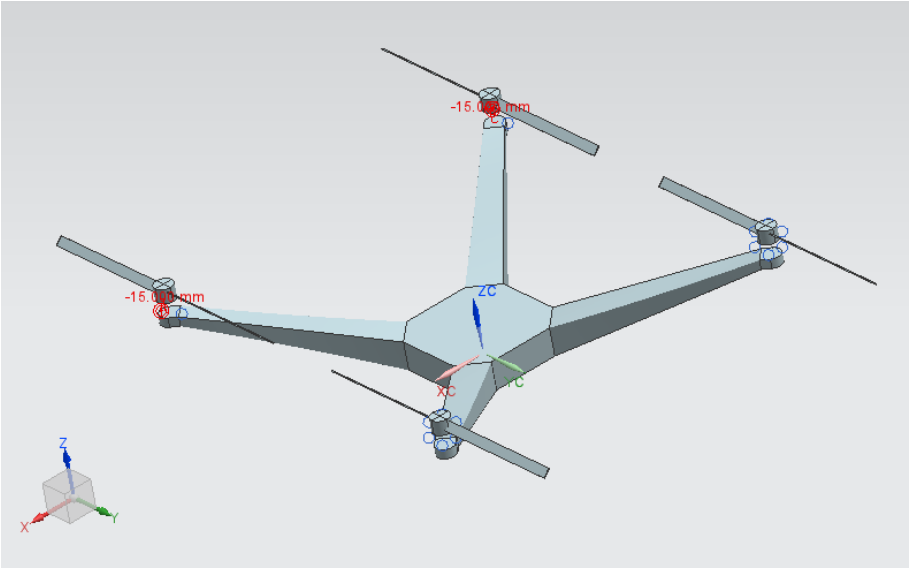


Figure 45: Initial Quadcopter Geometry Developed in NX

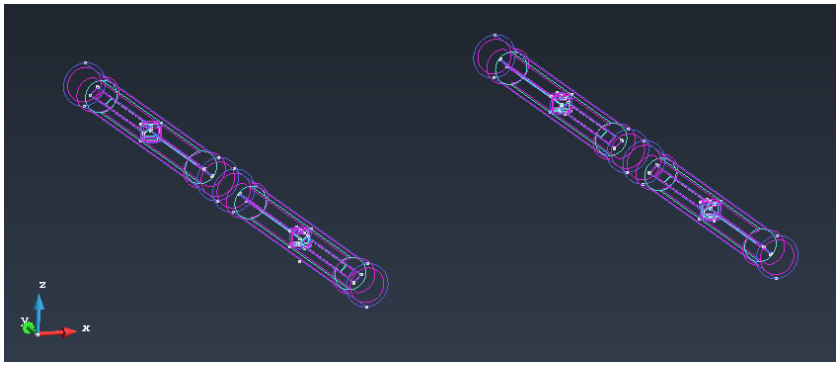


Figure 46: The 4 Blade Cylinder Patches to Be Rotated

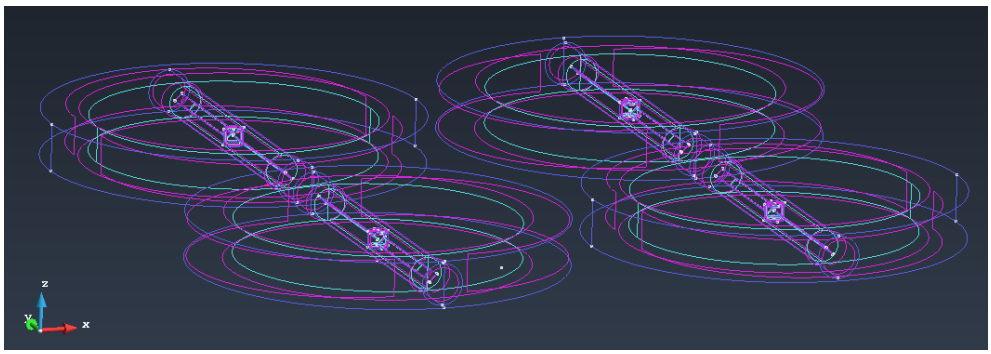


Figure 47: Encompassing Cylinder Mesh Refinement Strategy

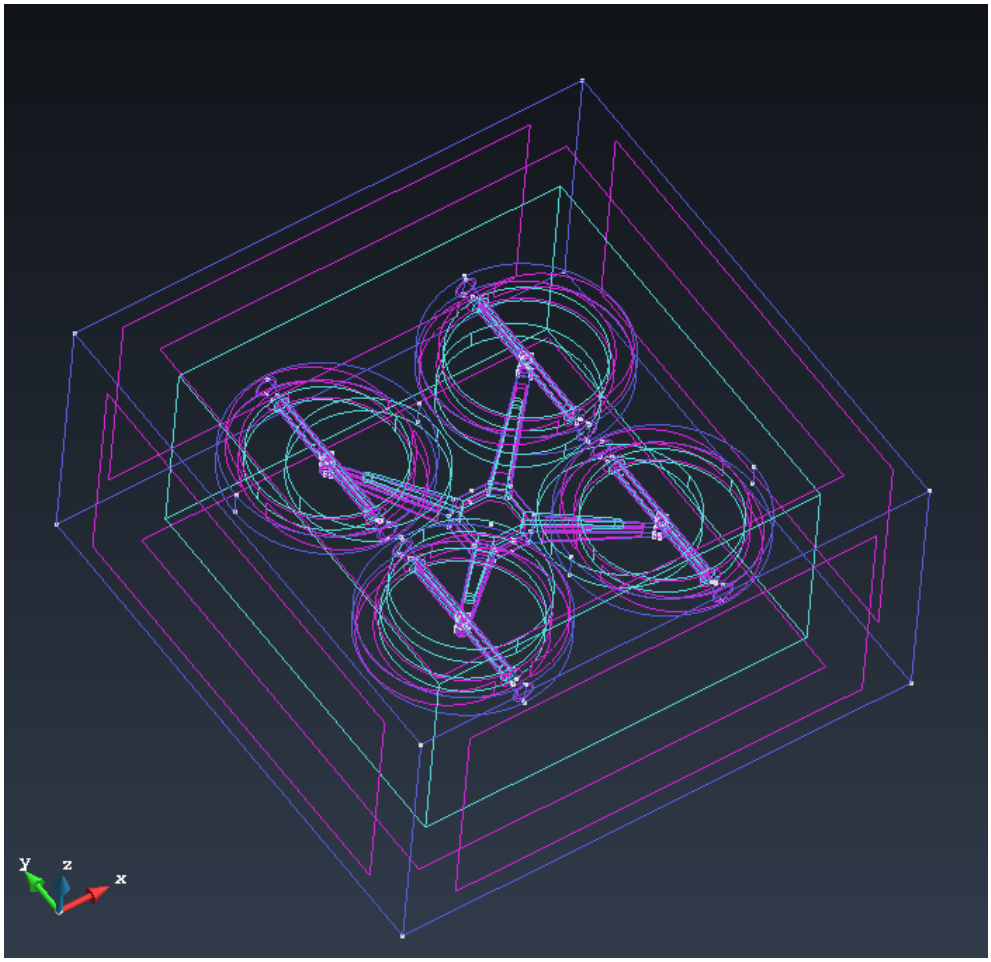


Figure 48: Total Geometry

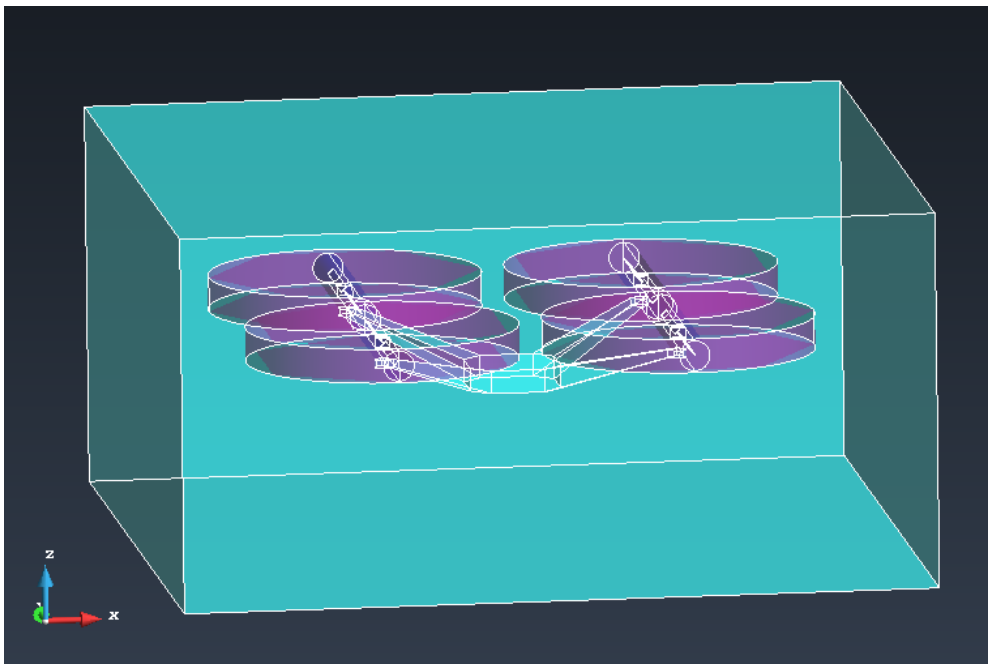


Figure 49: Total Rendered Geometry

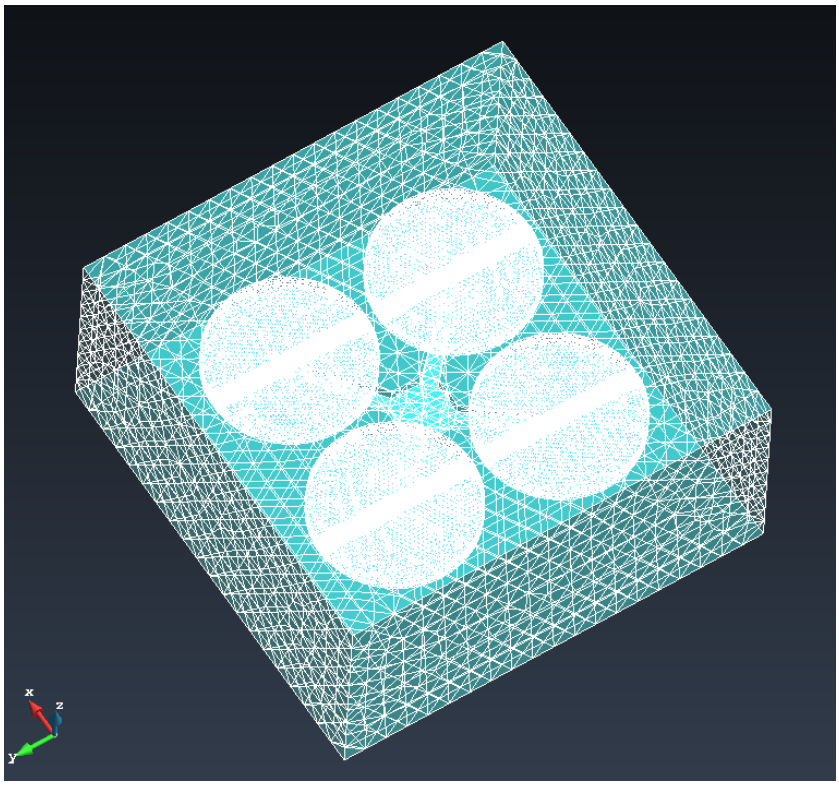


Figure 49: Total Geometry With Mesh Boundary

At the time of writing this the simulation is still underway and will take a couple weeks to finish. It is uncertain the quality of the results of this simulation.

6 Conclusion

In conclusion, over the course of the 8 weeks I have spent here at the Technical University of Munich, I have gained an introductory understanding to object oriented programming within the python coding language and a basic understanding of the Kratos framework built upon it. Additionally, I gained an enormous amount of practical experience modeling, meshing, simulating, and postprocesing both test cases and real world phenomena with a variety of software. This experience and the skills I have learned will make me a much stronger candidate for entering industry. I would like to thank my supervisor Aditya Ghantasala for all his help, and the Technical University of Munich for hosting me and providing me with this opportunity.