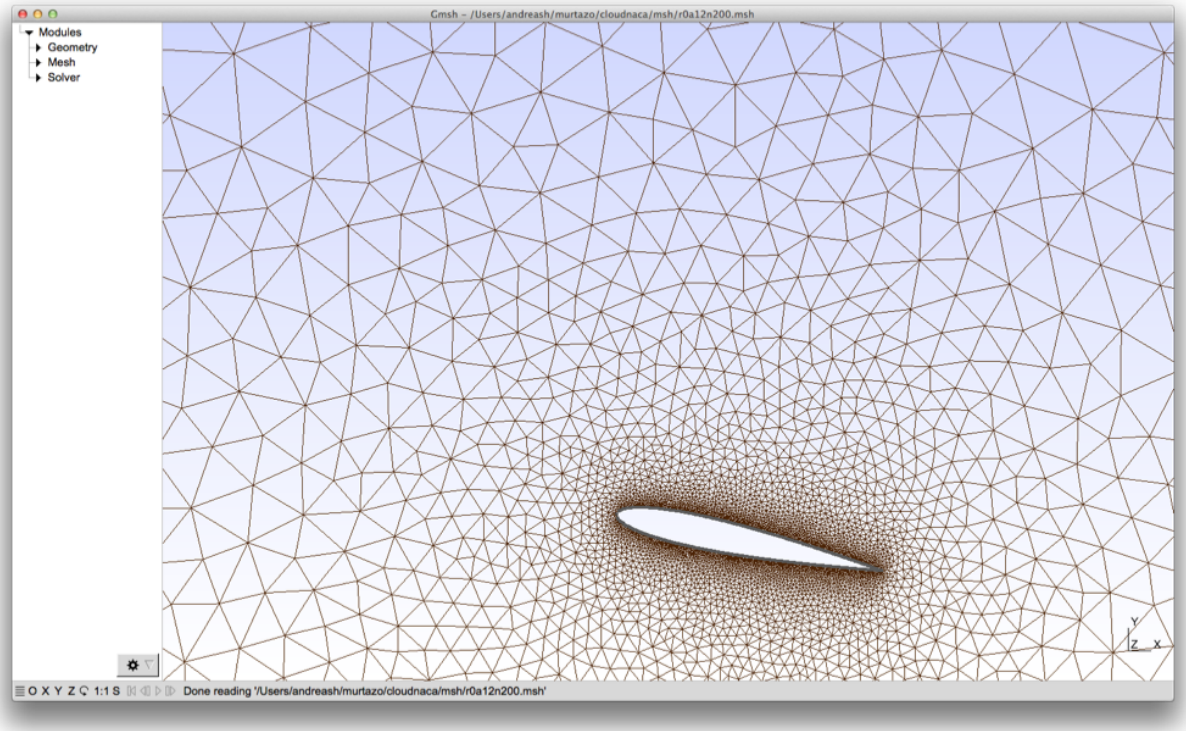
Angle of Attack in Airfoil Simulations

In this project you will build a distributed system and associated cloud service for assessing the optimal angle of attack of an airfoil based on Finite Element (FE) calculations.

To your disposal is a 2D Navier-Stokes solver written by a researcher at TDB, the binary is called “airfoil”. It is based on the Finite Element Method (FEM), and implemented using the open source framework FEniCS/Dolfin (www.fenicsproject.org). The easiest way to run FEniCS is via the Docker containers they distribute. Navier-Stokes Equations describe the motion of a viscous fluid, and is in this context used to calculate the velocity and pressure-fields of the air flowing past the wing. From these solutions, one can calculate for example the lift force and the drag force. But how do these forces depend on the angle of attack of the airfoil? Is there an optimal angle? To find this out (or any other question you may come up with), you can use the solver repeatedly with different geometries as input and for each of them calculate the solutions and forces. Of course, this can be a computationally intensive process, but each calculation is independent of any other, so it is very amenable to parallelization. This type of application is commonly referred to as a *Parameter Sweep Application (PCA)*.

Gmsh (http://geuz.org/gmsh/) is a widely used mesh generation software, and it can be used to create the type of meshes required as input to “airfoil”. Another researcher TDB has provided us with a script to generate input geometries and meshes based on Gmsh. The program can generate meshes with different angle of attack. The meshes will be in Gmsh’s own mesh-format, but for input to the FEM code, you need it in Dolfins XML format. There is a utility “dolfin-convert” (ships with FEniCS) that you can use to convert between the two mesh formats.

Fig 1. Example of a Gmsh mesh around an airfoil.



Your task is to design and implement a cloud-based solution and service for conducting experiments to assess the influence of the angle of attack on the lift force (for different values of the other inputs to the program). As a minimal requirement, the system needs to work for a statically configured input (apart from the meshes) and there needs to be an assessment of the horizontally scalability of your solution. For higher grade, the degree of sophistication of the solution will be assessed, in particular, how easily the system can make use of dynamic provisioning of compute resources if the workload increases. For example, if the number of input meshes are increased, can the system respond by adding more computational resources on-demand or even automatically? Can the system save already computed parameter points in some database or on files to avoid recalculating them at a later stage if the user requests some of the same parameter points? How will the user interact with the service, a simple RESTful API or a WebUI (not a requirement for a passing grade)?

Good luck!