

# A workflow for fluid-structure interaction simulations with preCICE

Introductory course at the preCICE Workshop 2024, Stuttgart, September 24-27, 2024

Claudio Caccia, Politecnico di Milano  
<claudiogiovanni.caccia@polimi.it>

Gerasimos Chourdakis, University of Stuttgart  
<gerasimos.chourdakis@ipvs.uni-stuttgart.de>

This is a community-contributed part of the [preCICE training course](#): You can directly [contribute on GitHub](#). The text and images of this lesson are available under the [Creative Commons Attribution 4.0 International license](#), while the data and source are available under the [MIT license](#).

Find citation information and check for newer versions on Zenodo: [10.5281/zenodo.13961370](https://zenodo.org/record/105281/files/preCICE_training_course.pdf).

## Dependencies

We will use the following software:

- [FreeCAD](#) (at least version 0.21)
- [preCICE](#) (at least version 3.1)
- [CalculiX](#) (tested with v2.20) and the [CalculiX adapter](#) (tested with v2.20.1)
- [OpenFOAM](#) (tested with v2406) and the [OpenFOAM adapter](#) (tested with v1.3.1)

## Task 0: Prepare the geometry



Figure 1: Cross-section of the 3D wing geometry

We will simulate fluid-structure interaction of a very simple airfoil, in 3D. [NACA airfoils](#) are standardized designs that are popular in designing aircraft wings. Our case is slightly derived from the following paper:

*S. Heathcote, Z. Wang, I. Gursul, **Effect of spanwise flexibility on flapping wing propulsion**, Journal of Fluids and Structures, Volume 24, Issue 2, 2008.*

In the paper, they considered a *NACA0012* profile, while our training involves a **NACA2312** profile wing (to have some lift at 0° angle of attack) with:

- *chord*  $c = 100\text{mm}$
- *span*  $b = 300\text{mm}$

We would typically start by designing our geometry in a CAD software (e.g., FreeCAD), but we assume you already know how to do that using your own workflows. To save some time, use the wing model that you can find in this folder.

We have generated the geometry using the software [Salome](#) and exported it to our needs.

You'll notice that two different file formats are provided: a `.stl` file and a `.step` file. Both are widely used for data exchange and nearly all CAD systems allow importing and exporting such formats. In this training, we will need both files: See why in the notes below.

When meshing a flow domain, we need to consider whether we are simulating external or internal flow. In this case, we are considering an external flow: we will use the solid geometry to generate the **solid mesh**, and we will subtract it from a sufficiently large box to generate the **fluid mesh**.

### Notes on file formats

You might wonder: "Why two formats?"

We will prepare the fluid mesh using the **snappyHexMesh** tool of OpenFOAM. This [supports various geometry formats](#), including [STL](#), a quite common format.

STL files describe unstructured triangulated surfaces. However, some tools cannot model solids based on this description. Instead, we will use a [STEP](#) file to describe the solid, which explicitly defines volumes.

Once you generate your model with your favorite CAD tool, you can export it in both formats and use these the way we'll use them in the following tasks.

Notice that, sometimes, some parameters need to be tuned in order to obtain a sufficiently refined STL surface. Use the provided files to avoid any mesh-related issues later on. Overall, meshing is often (very) complicated, and we only want to give you a starting point for a rather simple case.

## Task 1: Mesh of Solid domain

In this section we'll generate a CalculiX mesh for the solid, using the [FEM Workbench of FreeCAD](#). We will only use it to generate the mesh, and we will start the simulation from the terminal later on. See the general overview of this task in [Figure 1](#).

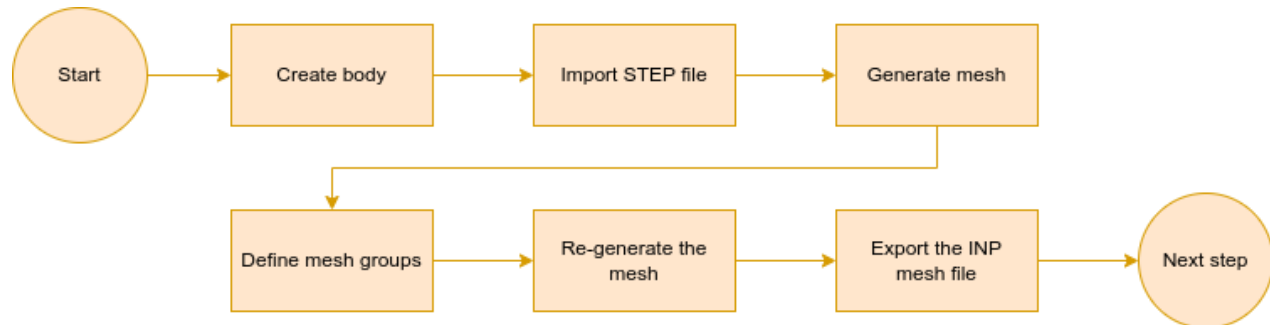


Figure 1: FreeCAD: General overview

## Adjust the FreeCAD settings

### Export settings

Before you start, change the settings of the INP exporter to export groups together with the mesh.

Select the [FEM Workbench](#) ([Figure 2](#)).

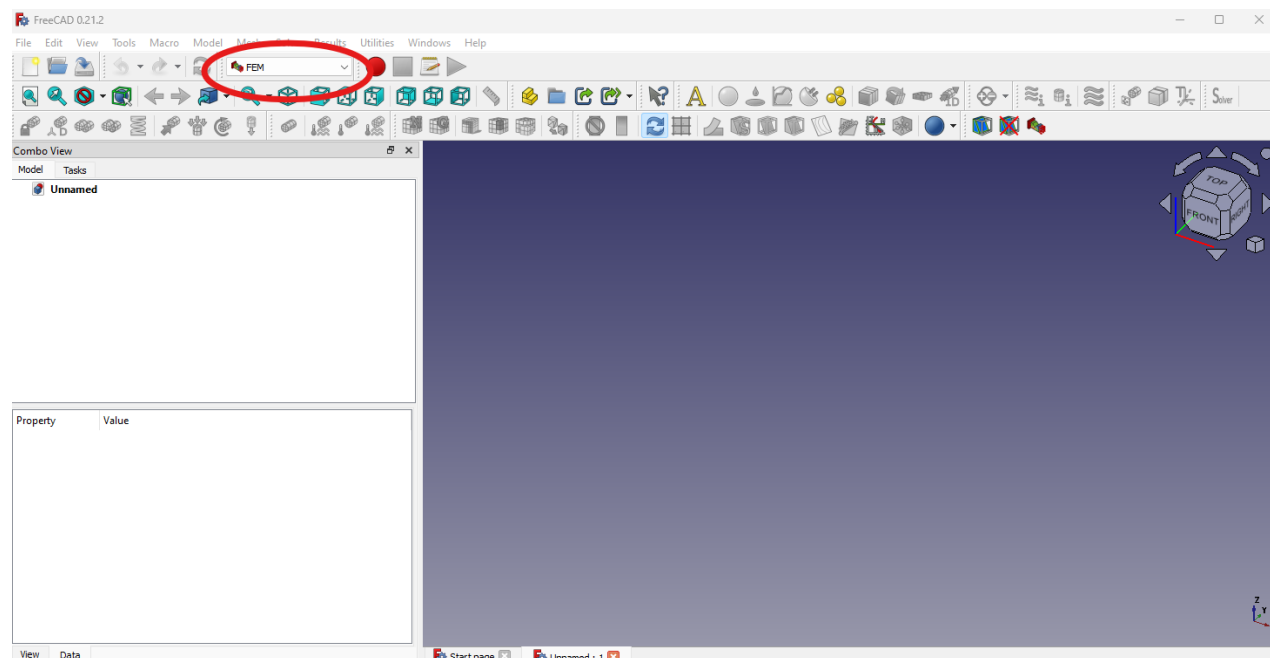


Figure 2: FreeCAD: FEM Workbench

Go to:

- Click Edit -> Preferences....
- Select the Import-Export icon on the left.
- Select the INP tab.

Configure as follows:

- Which elements to export: FEM.
- Export group data: (check).

Then, click **Apply** and **OK**.

## Import the wing

The first task consists in importing the wing (Figure 3):

- Create a new project: Click **File** -> **New**.
- Give the model a name: In the **Combo View/Model** tab on the left, select the unnamed model and change the **Label**: click on the **Unnamed** and rename it to **Wing**.

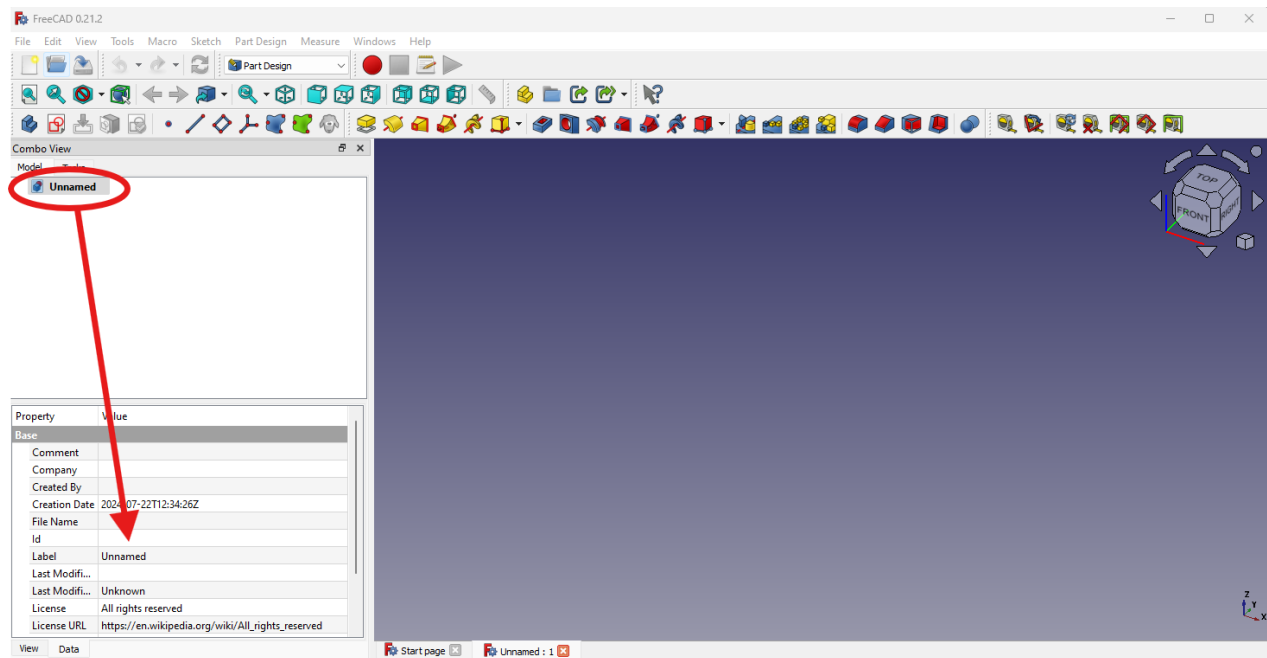


Figure 3: FreeCAD: Rename model label to Wing

- Select the **Part design** workbench from the drop-down menu and click **create body** (Figure 4).
- In the **Model** tab, you can now see a new body as part of the **Wing** model (Figure 5).
- Click **File -> Import...**
- Select the STEP file **naca2312.step**.
- In newer FreeCAD versions, a pop-up dialog will be shown. Click **OK** with the defaults (Figure 6).

An object named **Open CASCADE STEP Translator 7.5 1** should appear on the left, a wing-shaped object should appear rendered. Drag and drop this object into **Body**. This should appear as a **BaseFeature** entry under the **Body** (Figure 7).

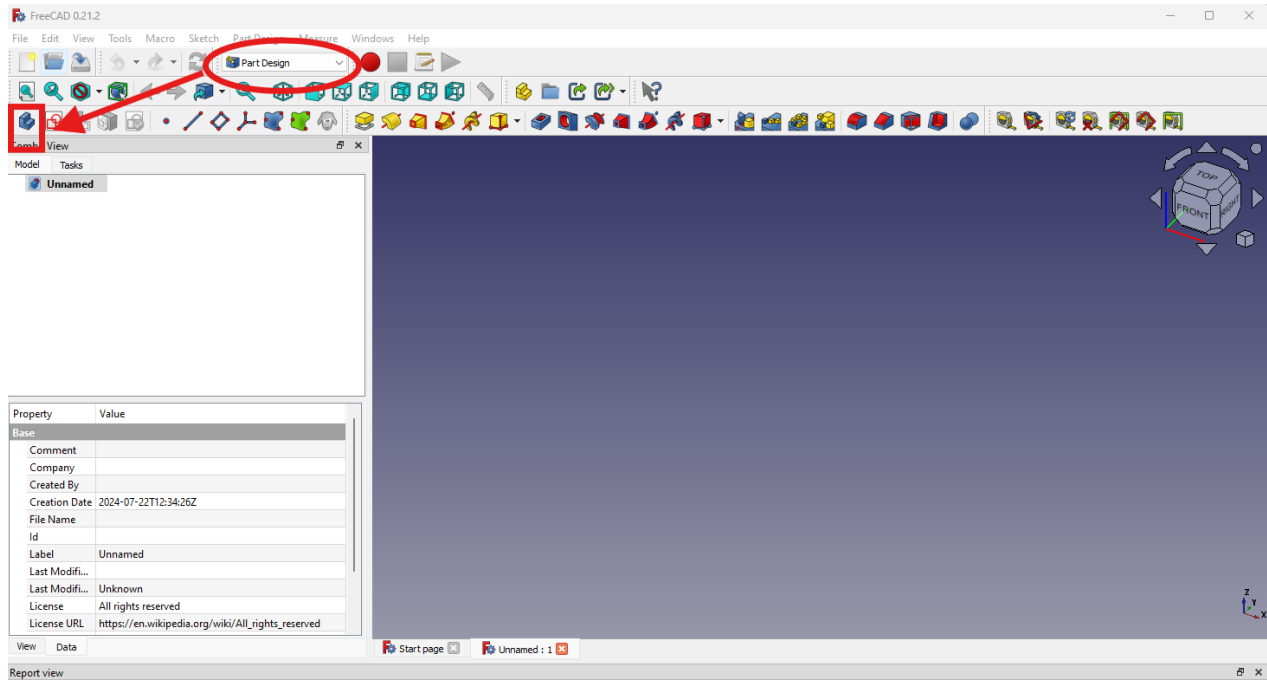


Figure 4: FreeCAD: Create a new body from the Part design workbench

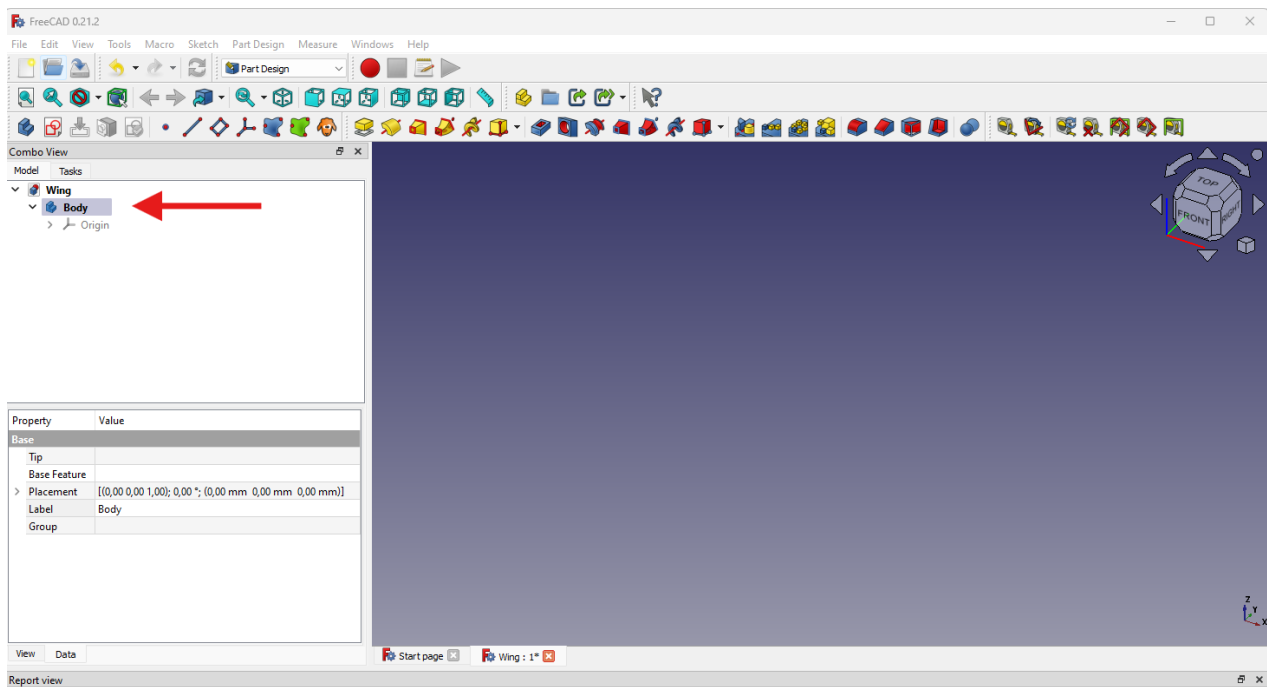


Figure 5: FreeCAD: Create a body

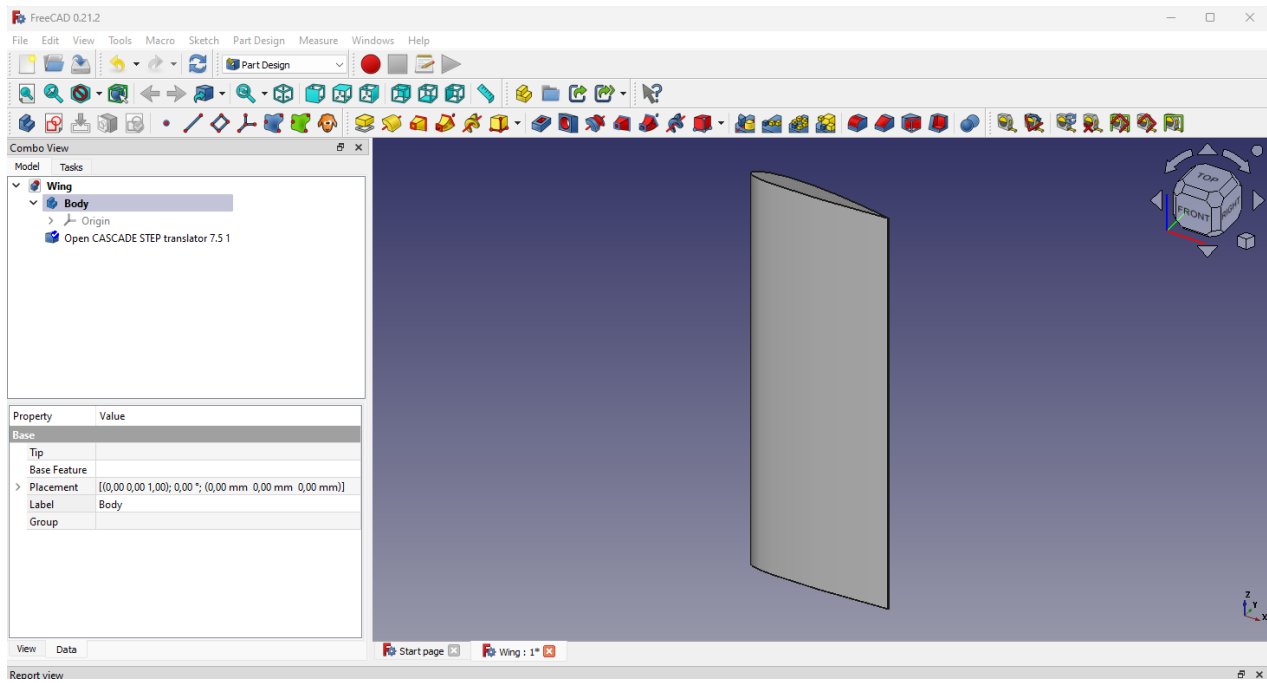


Figure 6: FreeCAD: Import geometry

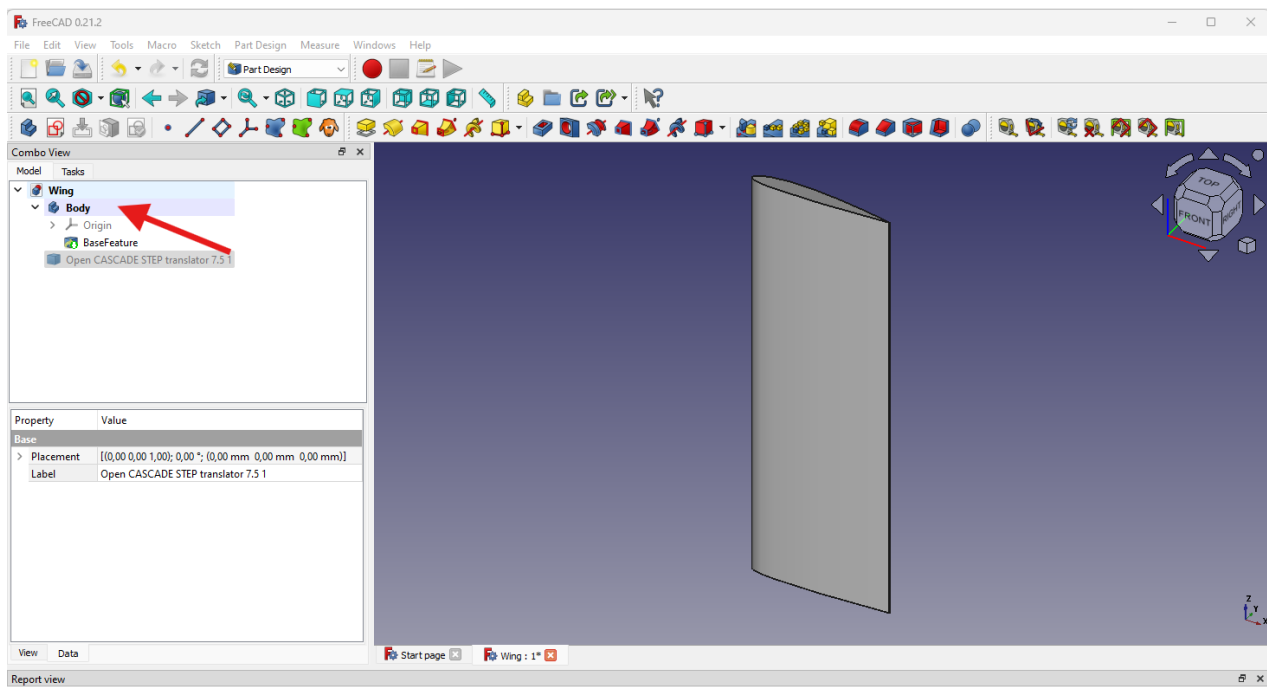


Figure 7: FreeCAD: BaseFeature

## Generate the mesh

Now we can generate a mesh for the **Wing**.

- Switch again to the FEM Workbench from the drop-down menu.
- From the menu bar, click **Model -> Analysis container** (or select the **A** symbol from the toolbar, Figure 8).

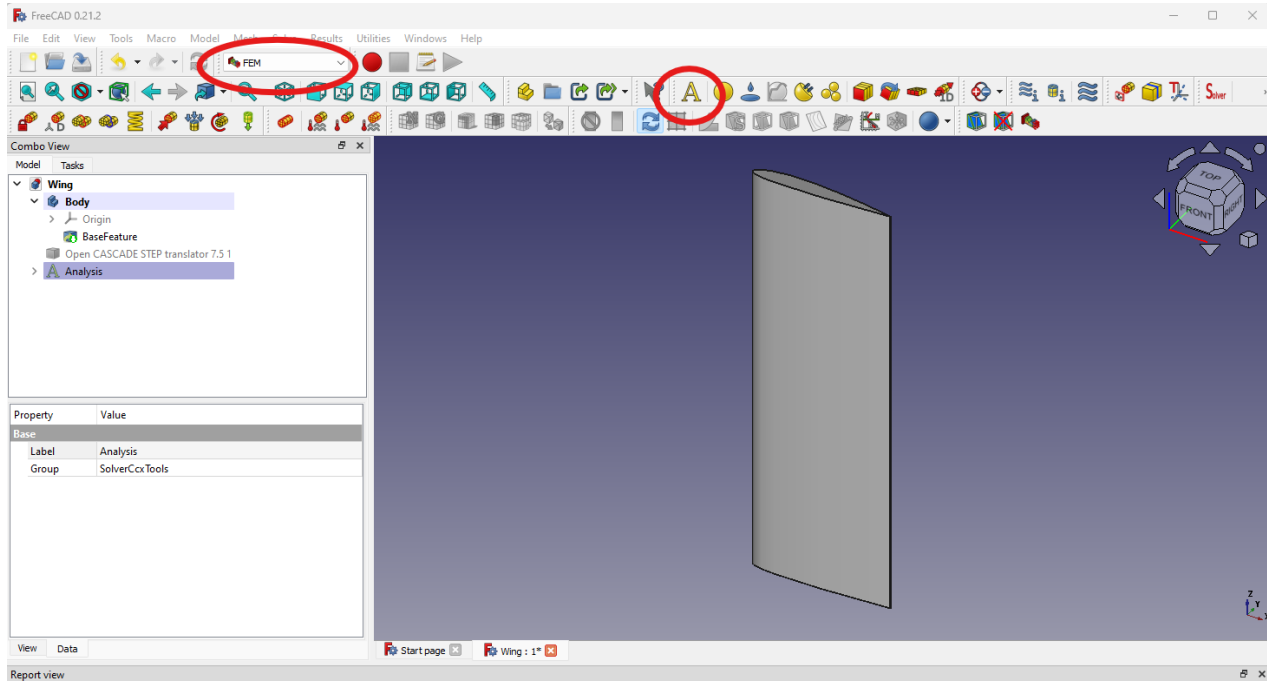


Figure 8: FreeCAD: Add an Analysis container

- We want to mesh the **BaseFeature**: Select it from the sidebar.
- From the menu bar, click **Mesh -> FEM Mesh from shape by GMSH** (Figure 9). The FEM Workbench can generate meshes using different backends; we use **GMSH** here.
- In the **Tasks** tab, use the following parameters (Figure 10):
  - Element dimension: **3D**
  - Element order: **2nd**
  - Max element size: **20mm**
  - Min element size: **10mm**
- Click **Apply** and **OK** to generate the mesh.

If everything went as expected, a mesh should appear (Figure 11). Otherwise:

- In case you get any error related to creating temporary files, see the troubleshooting section below.
- In case you get a **File to load not existing or not readable** error, try again after a couple of seconds.

## Create mesh groups

Now that we have a mesh, we also need to create the boundaries. We will need to identify the **root surface**, which will be clamped, and the **wet surface**, which will be in contact with the fluid. These surfaces are

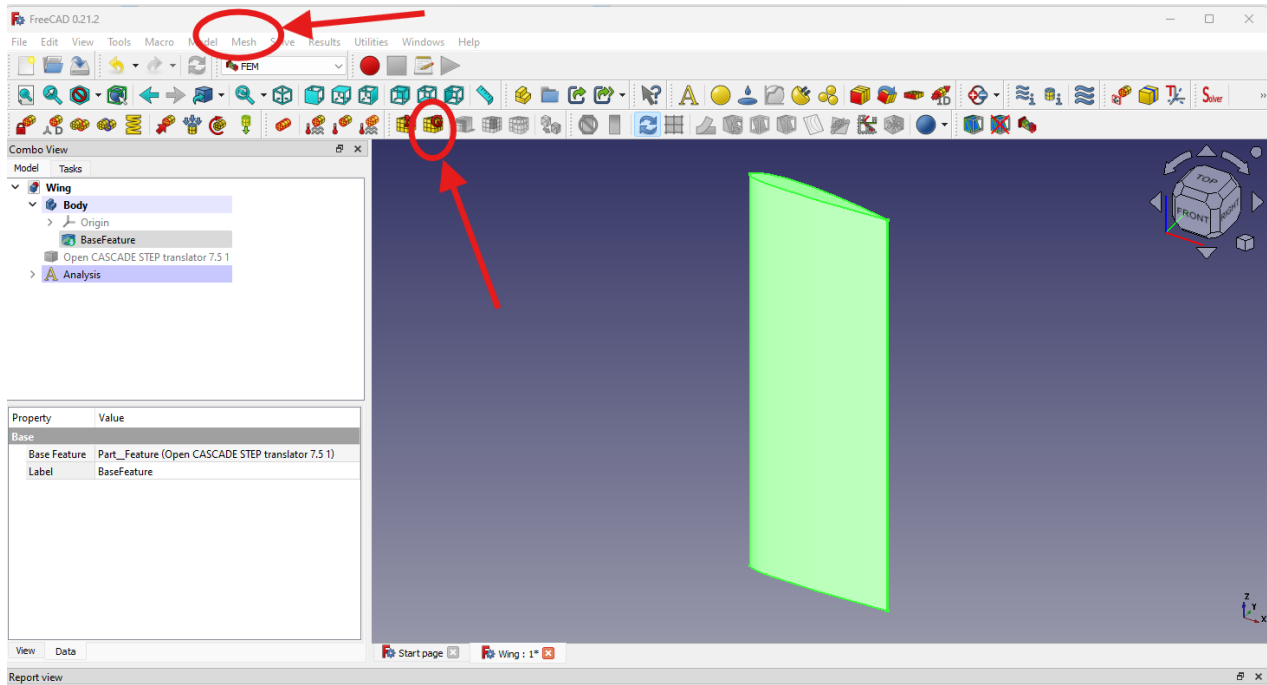


Figure 9: FreeCAD: Add a mesh

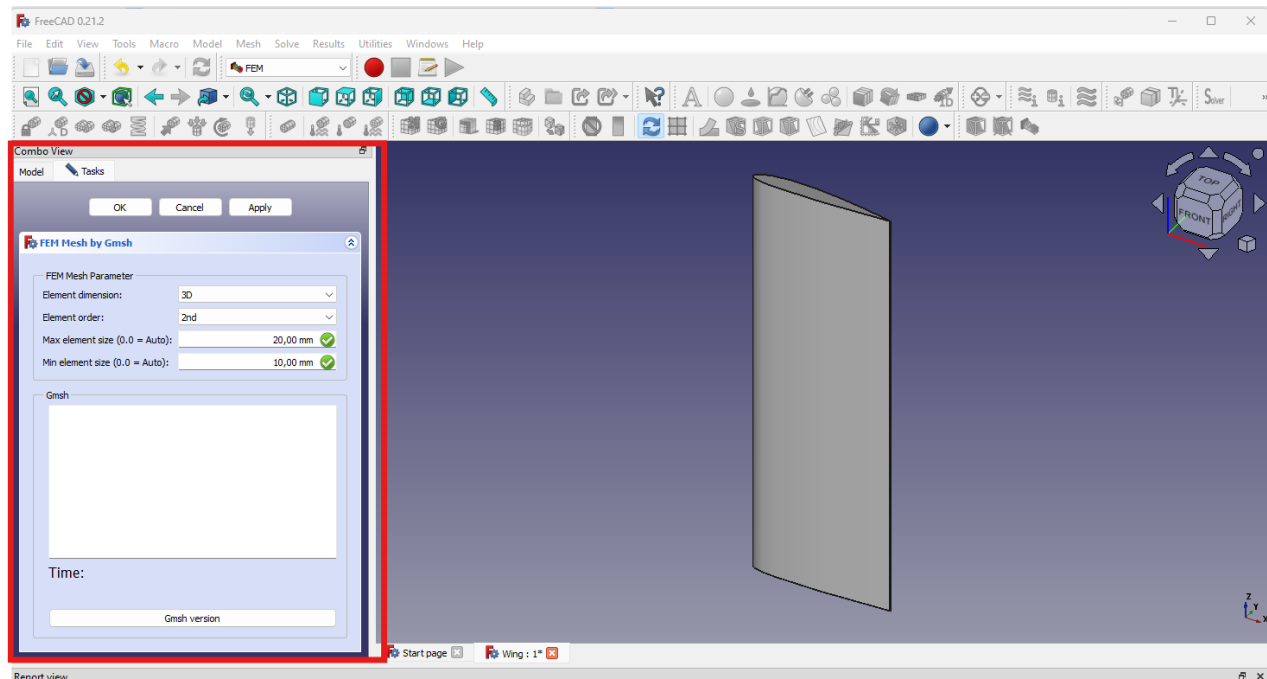


Figure 10: FreeCAD: GMSH parameters



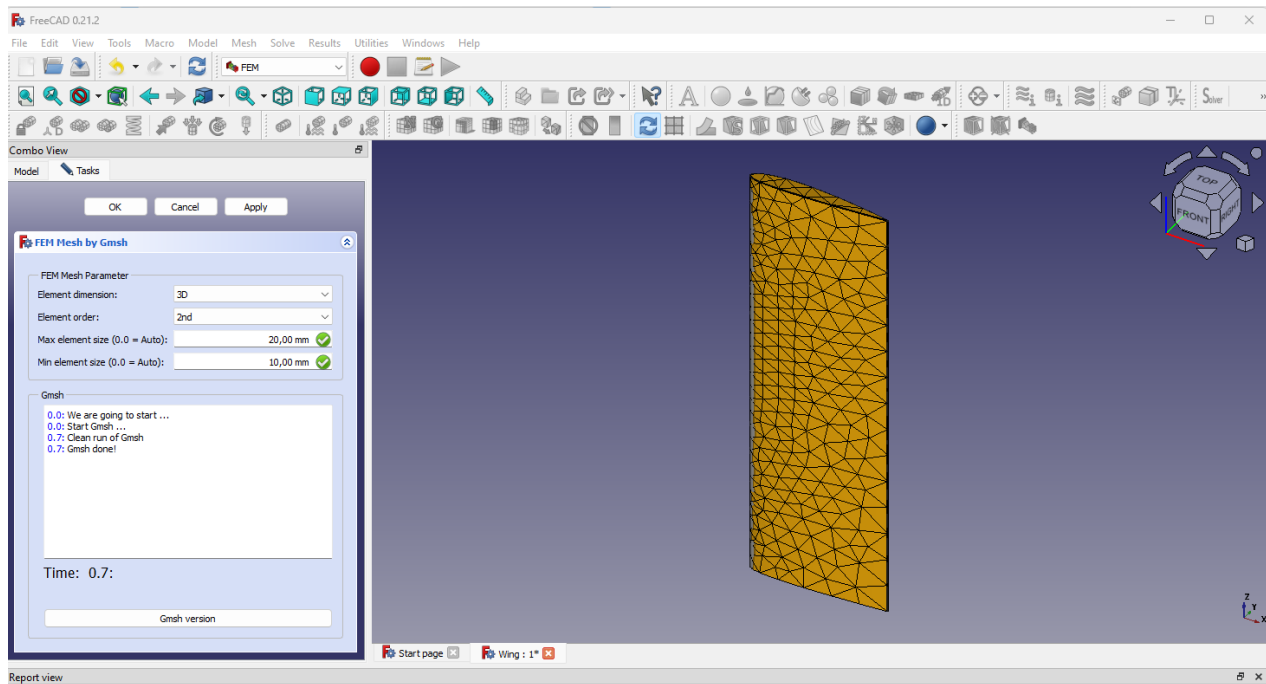


Figure 11: FreeCAD: Mesh generated

defined as [mesh groups](#).

- From the **Model** tab, expand the **Analysis** branch and select **FEMMeshGmsh** (Figure 12).
- Change the property **Groups of Nodes** to **true** (Figure 13).
- Click **Mesh -> FEM Mesh group** (Figure 14).
- In the **Model** tab, there should now also be a **MeshGroup** under the **FEMMeshGmsh** (Figure 15).
- In the **Tasks** tab, select **Label** as **Identifier used for mesh export** and **Face, Edge, Vertex** as **Selection mode** (Figure 16).
- Click **Add**, then click on the rendering to select the profile of the wing (pay attention to reference frame to identify it), and **OK** to add the surface to the mesh group (Figure 17). You can rotate the view using the cube in the upper right corner, or [using your mouse](#) (e.g., by **Shift + right click**).
- In the **Model** tab, select the **MeshGroup** and rename its **Label** to **Nroot** (Figure 18). This will help us define the boundary condition in the Solid domain.

**NOTE:** Node group names need to start with **N**.

- Select again the mesh (**FEMMeshGmsh**) and define a new group comprising all components of the **wetSurface** (they are 4: as shown below. Pay attention to the trailing edge surface, you need to zoom-in to see it). See Figures 19 and 20.
  1. Click **Add**
  2. select a patch
  3. repeat steps 1. and 2. for each of the four elements
  4. Click **OK**
- As for the **root Group**. Change the **Label** in the **Properties** to **NwetSurface** (Figure 21).





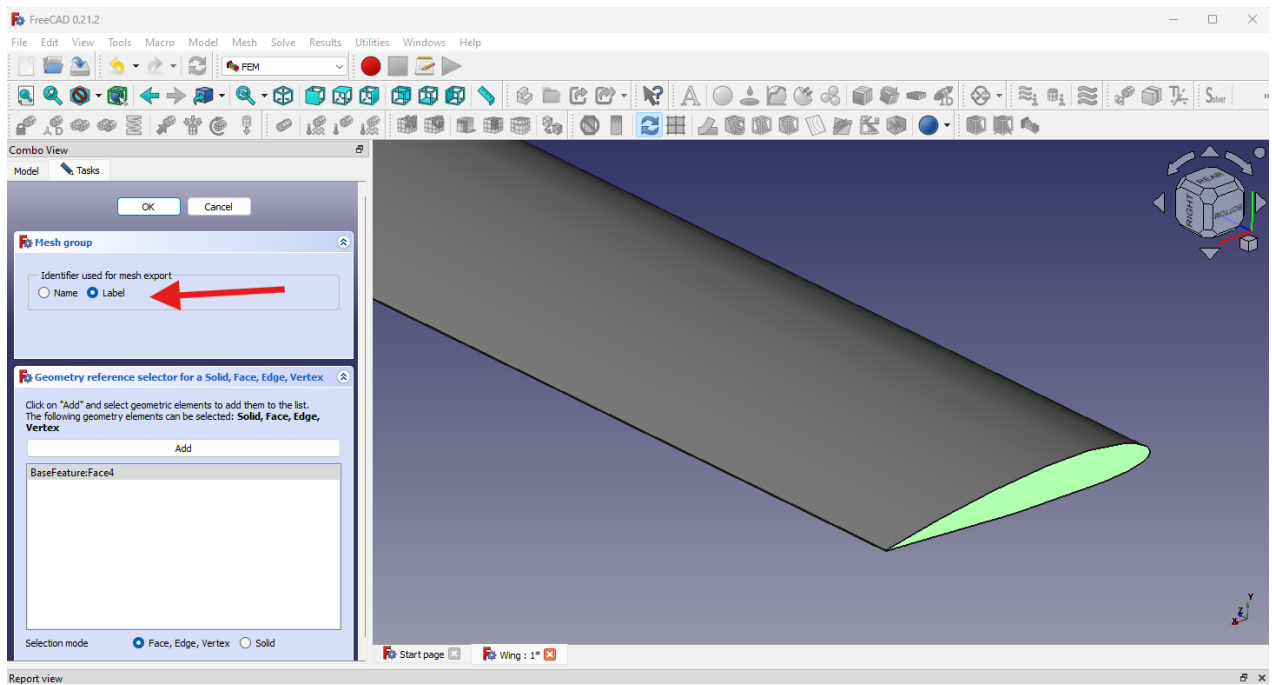


Figure 16: FreeCAD: Add surface to mesh group

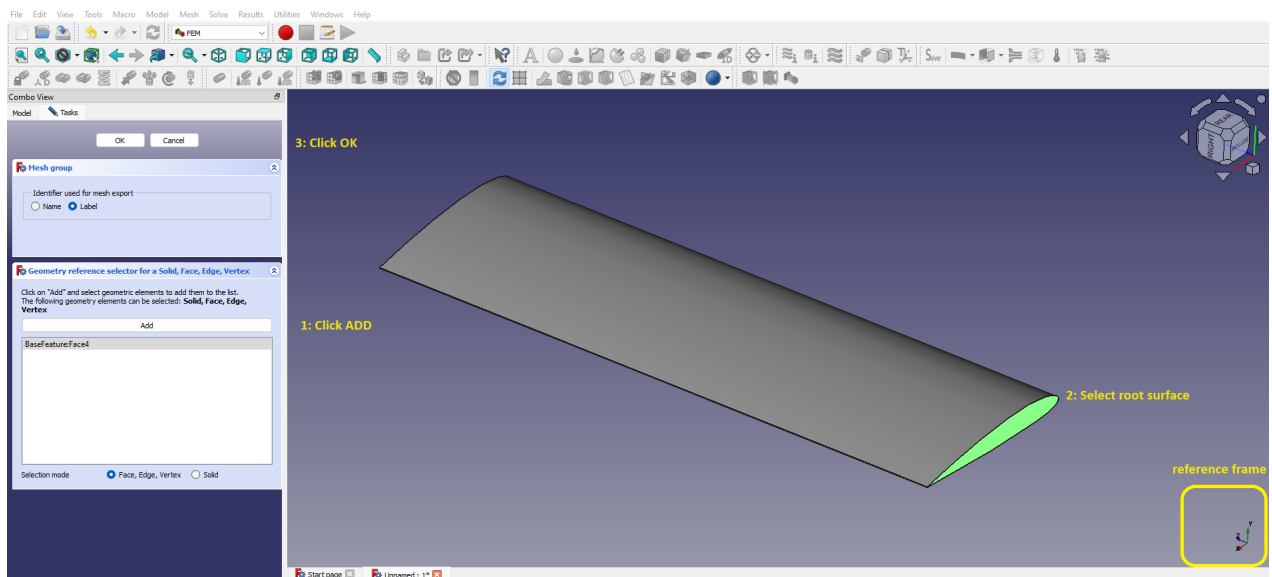


Figure 17: FreeCAD: surface group for root

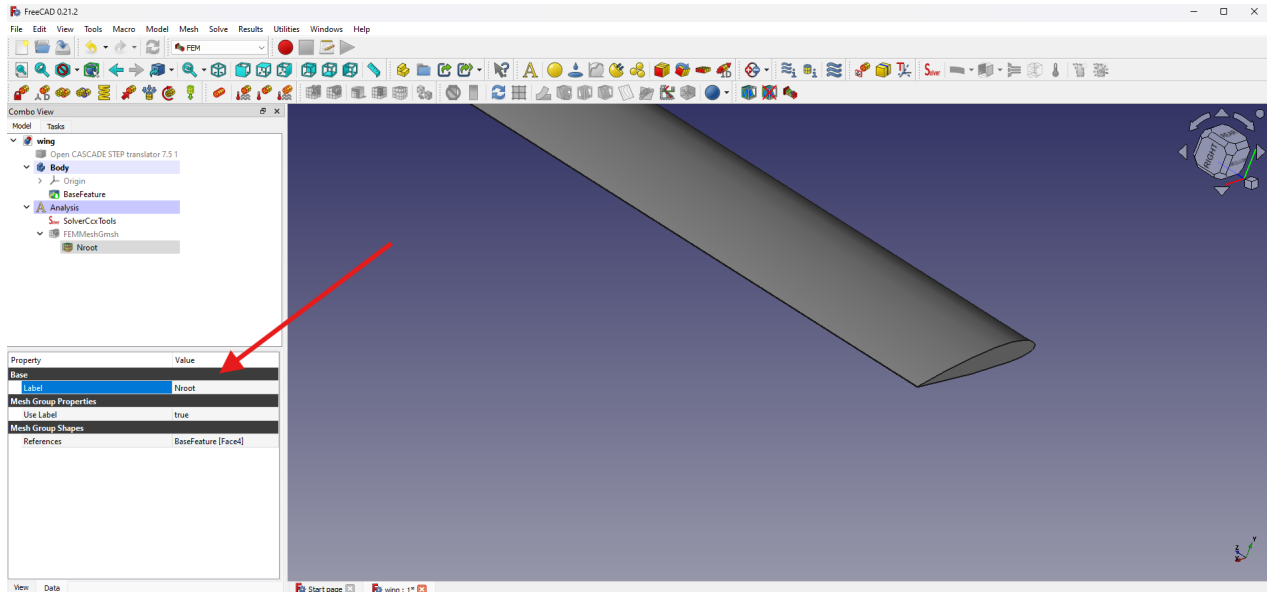


Figure 18: FreeCAD: Rename mesh group

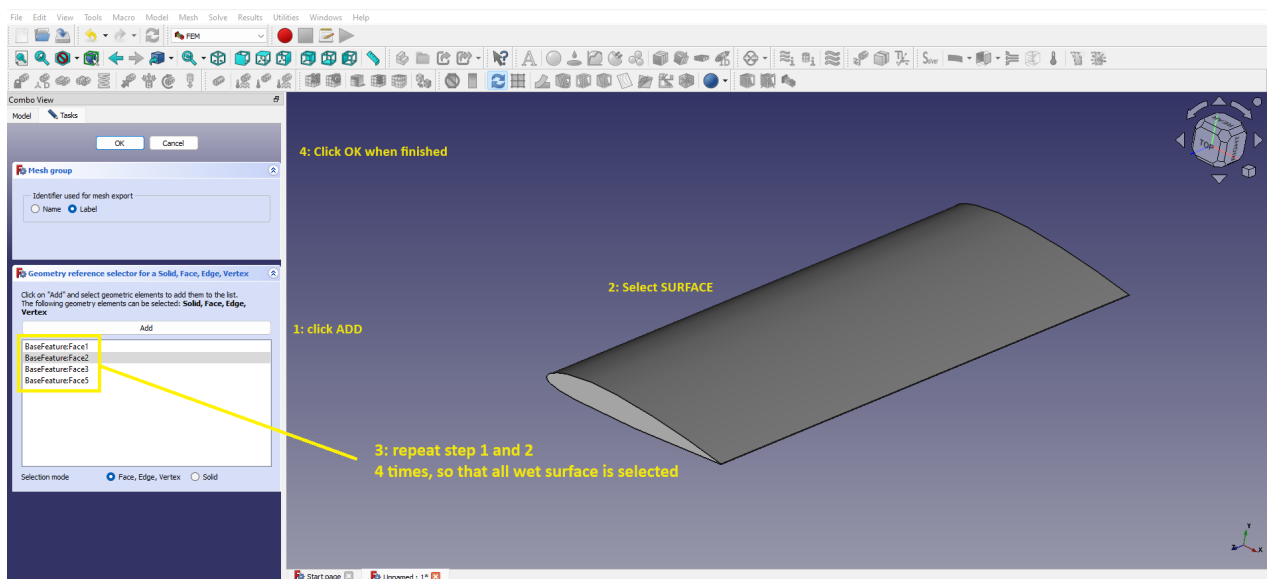


Figure 19: FreeCAD: wet surface steps

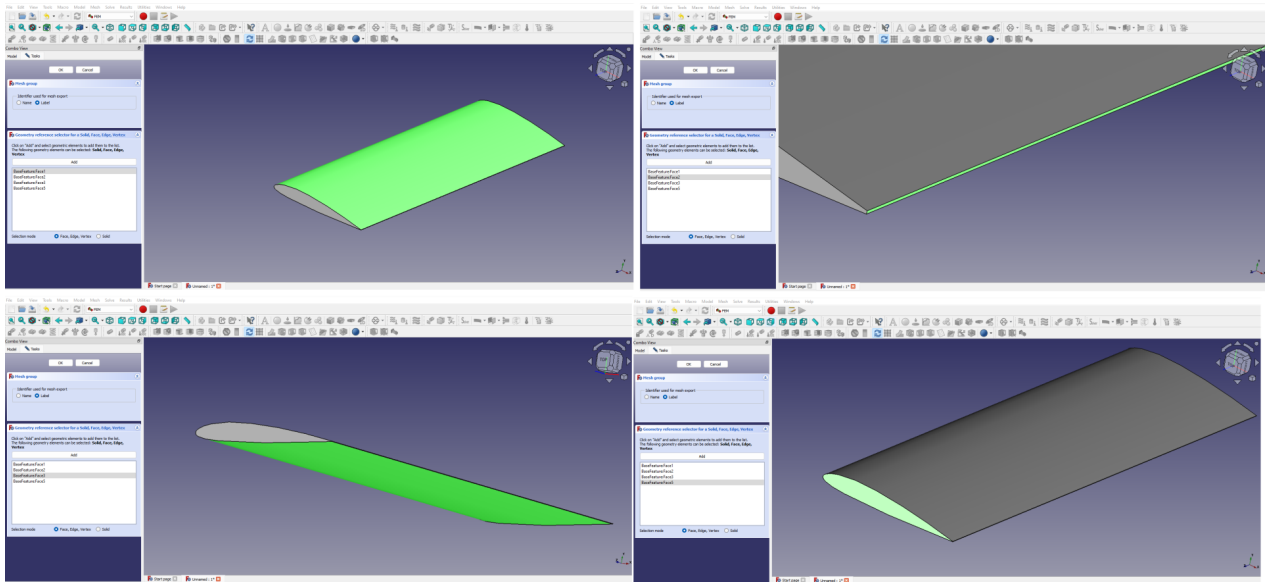


Figure 20: FreeCAD: wet surface group

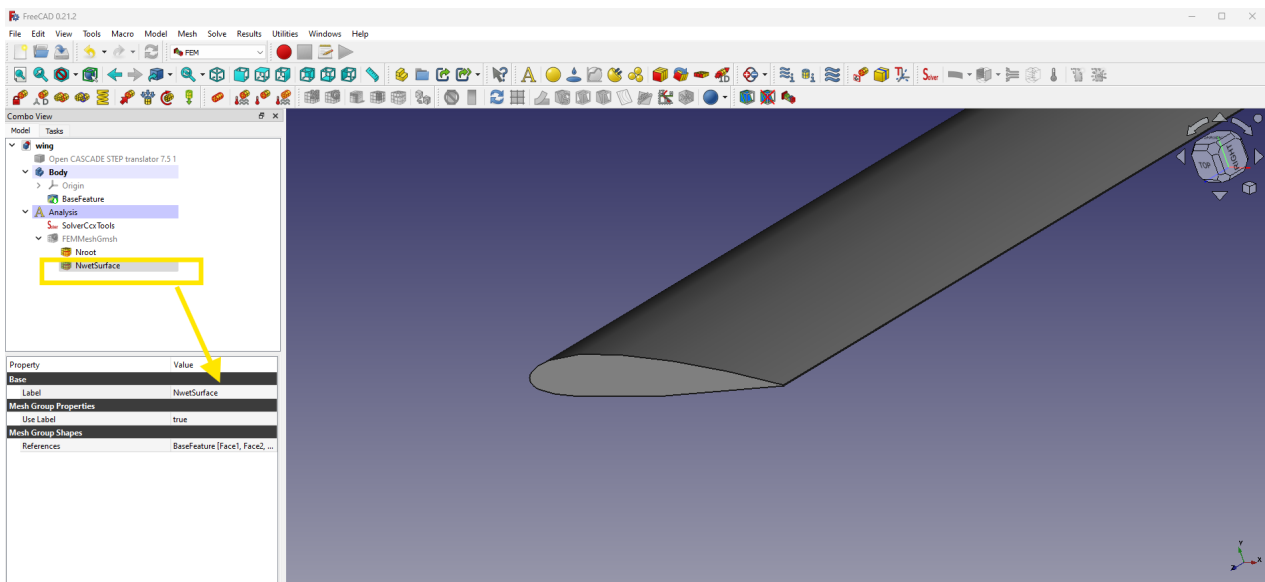


Figure 21: FreeCAD: Rename wet surface group

You should now see a list of two groups under the current mesh (Figure 22).

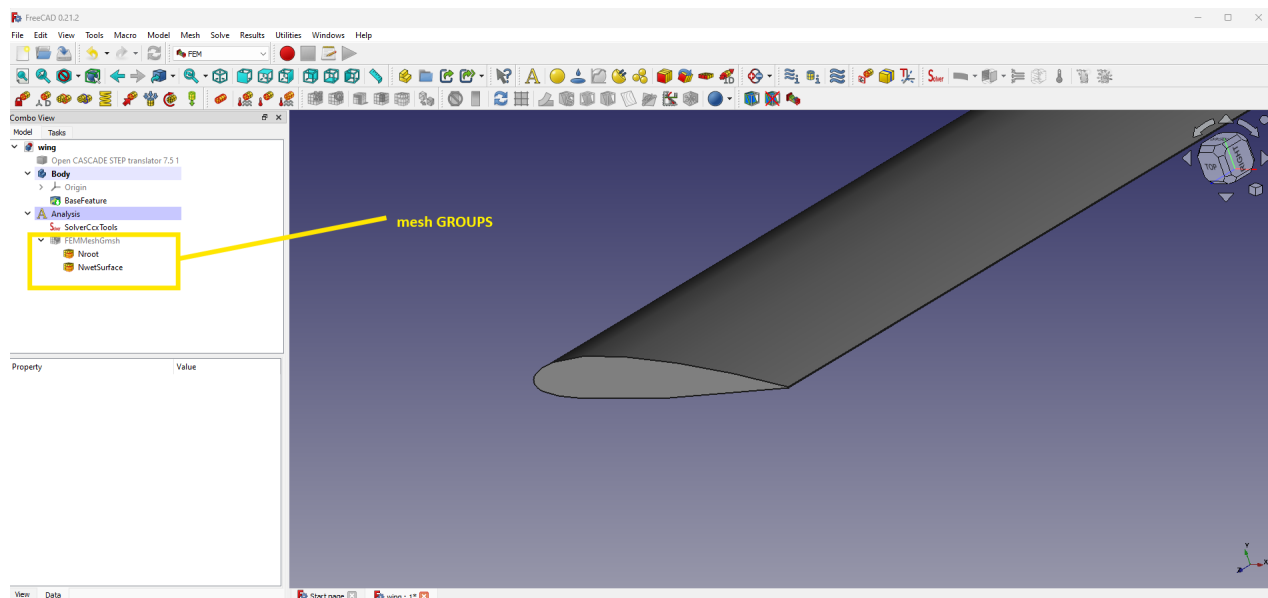


Figure 22: FreeCAD: final list of groups

**IMPORTANT:** You now need to re-mesh to generate the groups. Select the mesh (**FEMMeshGmsh**) -> double click -> click **Apply** to re-mesh and create groups -> click **OK**.

## Export the mesh file and verify

We are now ready to export the mesh into a `.inp` file (Abaqus input file format, compatible with CalculiX):

- Keep the mesh (**FEMMeshGmsh**) selected.
- Click **File** and **Export...**
- Name the file `wing2312.inp` and place it in the `01_solidMesh` directory.
- Select **FEM mesh formats (\*.dat, \*.inp, ...)** and click **Save**.

Save also the FreeCAD project with **File** and **Save**.

To verify, open the `wing2312.inp` file you just generated with a text editor:

- Look for **NSET**: you should find
  - **NSET=Na11**: this keyword defines the beginning of the list of coordinates of all the mesh nodes
  - **\*NSET, NSET=Nroot\_Nodes**: this keyword defines the beginning of the list of node IDs belonging to the mesh group **Nroot**
  - **\*NSET, NSET=NwetSurface\_Nodes**: this keyword defines the beginning of the list of node IDs belonging to the mesh group **NwetSurface**
- Take note of the exact names of all the sets of nodes for each of the groups, because we'll use them in the following steps.
- **\*ELSET, ELSET=GROUPNAME\_Faces**: where **GROUPNAME** is one of the groups that you defined (**Nroot**, **NwetSurface**). These sets define the groups of surface elements. We don't need them for the FSI simulation

## Scale the mesh file

Unfortunately, FreeCAD exports all node coordinates in millimeters and, at least in version 0.21, there is no option to change it. We prefer to have everything in SI units so, in the `01_solidMesh` folder you can find a `inp_convert.py` file. After checking that the name on line 43 matches your mesh file, run it:

```
python3 inp_convert.py
```

You should find a `wing2312_m.inp` file in your folder.

## Troubleshooting

### File permission issues

Check this section if you face any file permission issues while running GMSH.

Depending on the installation method (e.g., when using an AppImage file on Linux), some tools (e.g., GMSH) might not be able to write some necessary temporary files in the default directories. You can change this working directory:

- Click **Edit -> Preferences**.
- Select the **FEM** icon from on the left.
- In the **General** tab, switch from **Temporary directories** to **Use custom directory**.
- Select a path where you know your user can write files (e.g., your **Desktop**).

### Options unavailable

In case you cannot select anything, and you get an error “Active Task Dialog found!”, you might need to switch to the **Tasks** tab and click **OK** or **Cancel** to exit from the previous task.

Several options only appear if you have selected an object they can be applied on, and every workbench comes with completely different buttons and options.



## Task 2: Simulation of the Solid domain

In this section we'll simulate the Solid domain alone, to gain confidence with the CalculiX syntax and to check that our solid mesh and model work. CalculiX allows us to perform different kinds of simulations (e.g., static, dynamic, frequency...). The coupled simulation we want to end up with will be a dynamic one, so we already start with setting up a dynamic single-physics simulation.

This model represents a cantilevered wing subject to its own weight. The load is applied progressively, with a ramp law. Several simplifications are made here in the sake of time restrictions.

See a general overview of this task in Figure 1.

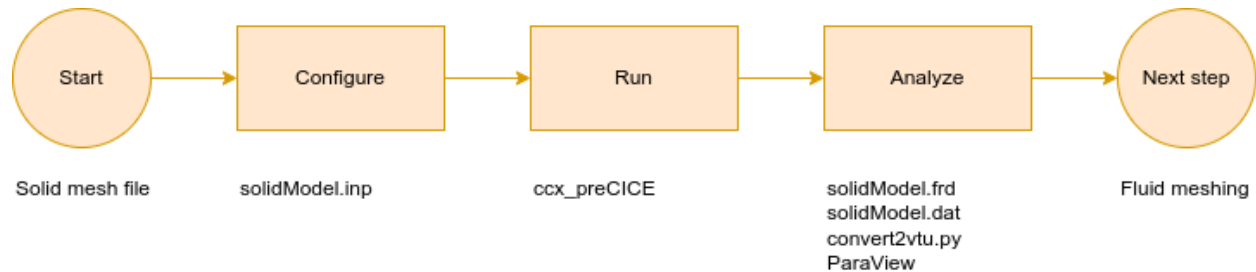


Figure 1: Solid simulation: General overview

### Configuration

In the `skeleton` folder:

- Copy your generated solid mesh into the current folder

Open the `solidModel.inp` file and:

- Replace `YOURMESH.inp` (line 4) with the name of the mesh (we previously named this `wing2312_m.inp`)
  - Note that CalculiX expects distance units in meters, while FreeCAD generates meshes with distances in millimeters. We need to adapt the values (see the end of the solid meshing task).
- Replace the material properties with the following, roughly corresponding to Polystyrene or ABS:
  - Replace `E` with `1.0E9` (Young modulus:  $E = 1\text{GPa}$ )
  - Replace `NU` with `0.35` (Poisson ratio:  $\nu = 0.3$ )
  - Replace `RHO` with `1060` (density:  $\rho = 1060 \frac{\text{kg}}{\text{m}^3}$ )
- Replace the numerical properties `DAMP`, `DT`, `TFINAL` with:
  - Replace `DAMP` with `0.0025` (structural damping, see notes below)
  - Replace `DT` with `5.0E-2` ( $\Delta t = 5 \cdot 10^{-2}\text{s}$ )
  - Replace `TFINAL` with `4.0` ( $t_{\text{final}} = 4\text{s}$ )
- Replace `NODESET` with the name of the set of root nodes (`Nroot_Nodes`)
- Replace `RAMPSEQUENCE` with the sequence `0.0, 0.05, 0.5, 1.0, 4.0, 1.0`. This is a sequence of value pairs `{time, amplitude}` as in Figure 2.

Notice the structure of the file:

- Input a geometry file
- Define the material properties
- CalculiX allows you to add **Rayleigh damping** to dynamic simulation, using the keyword `DAMPING`, which takes 2 arguments: `ALPHA` and `BETA`. They define the damping matrix as  $C = \alpha \cdot M + \beta \cdot K$
- Define a computation step
- Define a dynamic simulation

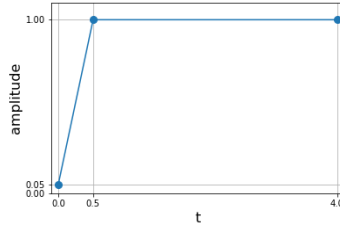


Figure 2: Solid simulation: Amplitude over time for the applied load

- **DIRECT** specifies that the user-defined initial time increment should not be changed
- **ALPHA** takes an argument in the range  $[-\frac{1}{3}, 0]$ . It controls the dissipation of the high frequency response: lower numbers lead to increased numerical damping
- Define constraints: Here, we define a constraint in which the nodes belonging to the set are fixed. Numbers 1, 3 indicate that node coordinates from 1 ( $x$  direction) to 3 ( $z$  direction) are fixed.
- Define loads. We define here a distributed load (body force) **GRAV**  $\vec{g} = 9.81$  with direction  $(0, -1, 0)$ , applied gradually over time.
- Define the simulation output:
  - **U**: displacements
  - **S**: stresses
  - **E**: strains
  - **RF**: resultants (i.e., combinations) of the reaction forces on the root nodes. These are additionally computed values that will be printed to a log file.

## Run the simulation

In order to run the simulation, open a terminal in the current folder and type:

```
ccx_preCICE -i solidModel
```

Notes:

- Remember to type the input file without the extension
- If you need to clean your simulation, you can use `clean.sh`
- Even though we are using the executable `ccx_preCICE` (modified CalculiX which includes calls to preCICE), we have not defined any coupling interface yet. This is only a single-physics simulation for now.

## Analyze the results

The main result files are:

- `solidModel.frd`: CalculiX result format, which contains all the **U**, **S** and **E** information.
- `solidModel.dat`: log file containing the reaction forces.

We can convert `frd` files to other formats supported by ParaView using various converters. Look for the `convert2vtu.py` file in the current folder and type:

```
python3 convert2vtu.py
```

This script calls `ccx2paraview` with the appropriate settings and generates one `vtu` file per time step and a `pvd` file pointing to these.

Alternatively, we could directly open `.frd` in FreeCAD, in the CalculiX tool CGX, or in other tools. However, we will later want to open the results of both the Solid and Fluid participants in the same tool, and ParaView fits this purpose.

## Deformation of the wing

Open the `solidModel.pvd` file in ParaView. You can then look at the deformed shape of the wing by applying a `WarpByVector` filter based on the `U` vector (and a small scale factor). See Figure 3.

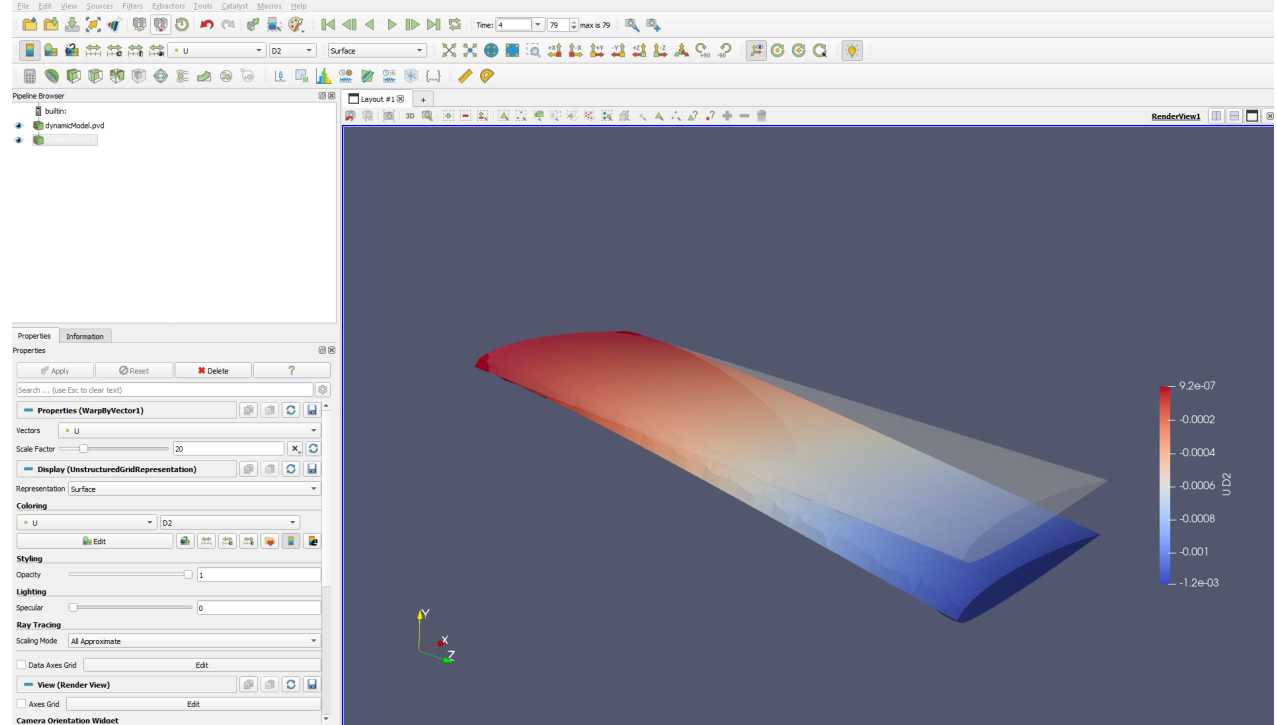


Figure 3: Solid simulation: Wing deformation, scaled (`WarpByVector` filter)

## Reaction forces

Open `solidModel.dat` with a text editor. This file contains a vector for each time step of the simulation. If you go towards the end, you'll notice that the `y` component of the reaction force converges (in magnitude, opposite in sign) to the weight of the wing.

## Theoretical data of the wing

The good thing with using standard geometry designs (in this case, NACA airfoils), is that we get data to compare our simulation results to, or we can easily compute derived quantities. For this NACA2312 and these material parameters:

- Area section of the wing:  $A = 8.0958 \cdot 10^{-4} m^2$
- Inertia moment  $J_x = 6.9464 \cdot 10^{-9} m^4$
- Length of the wing:  $l = 0.3m$
- Total weight of the wing is  $\rho \cdot g \cdot A \cdot l = 2.526kg$
- Distributed load along the span (beam approximation, see picture below)  $w = \rho \cdot g \cdot A$

- Expected tip displacement (Figure 4):  $y_B = -\frac{wl^4}{8EJ_x} = -1.227 \cdot 10^{-3}m$

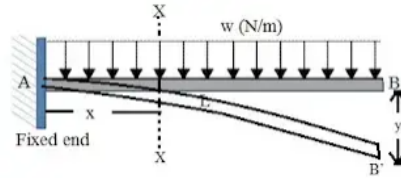


Figure 4: Solid simulation: Wing tip displacement analysis

Note that we are a bit lazy here, as the  $x, y$  axes are not *principal axes*. Nevertheless, we are very close and the approximation holds.

## References

- CalculiX 2.20 user manual: [https://www.dhondt.de/ccx\\_2.20.pdf](https://www.dhondt.de/ccx_2.20.pdf)

## Task 3: Mesh of the Fluid domain

In this section we'll generate the Fluid Mesh for OpenFOAM. As we are doing external aerodynamics, we will first generate a background mesh and then embed the geometry in that mesh, removing the respective region. We will generate the mesh directly with the OpenFOAM tools `blockMesh` and `snappyHexMesh`.

See a general overview of this task in Figure 1.

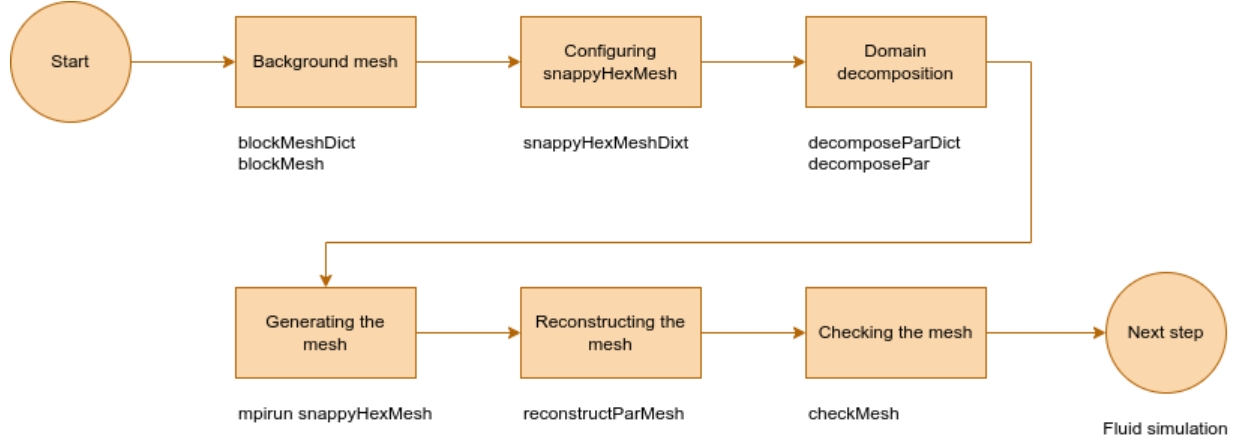


Figure 1: Fluid mesh: General overview

In `skeleton/`, you can find:

- The `constant` and `system` directories expected in an OpenFOAM case
- `clean_mesh.sh`: Removes all the intermediate and solution files
- `run_mesh.sh`: Runs all the required steps in a batch

In order to generate a mesh, we need the following:

- A background hexahedral mesh which defines the extent of the computational domain and a base level mesh density. Generated using `blockMesh`
- An STL file with the respective object geometry
- A `snappyHexMeshDict` dictionary

### Background mesh

To generate the background mesh, we need to configure the `system/blockMeshDict` file. Inspect this file and follow the comments (everything is pre-filled).

### Domain dimensions

We create a bounding box with dimensions:

- 1.6m long in  $x$  direction (the direction of the freestream)
- with a section of  $0.48 \times 0.48$  m in  $y$  (*lift* direction) and  $z$  (*span* direction)

The reference frame of the wing assumes point (0,0,0) to be in the middle of the chord:

- We place the `inlet` face at  $x_1 = -0.24$  m and the `outlet` face at  $x_2 = 1.36$  m.
- We place the wing in the middle of the box in  $y$  direction, so we place the  $y$  limits at  $y_1 = -0.24$  m and  $y_2 = 0.24$  m.

- Finally, we place the root section at  $z_1 = 0$  m and the final face at  $z_2 = 0.48$  m.

All these parameters are set in the beginning of `blockMeshDict`. For the moment, leave them like this.

### Mesh resolution

Once we have defined the limits, we need to define the number of cells that we want in each direction. Look for the dictionary entry `blocks` in `blockMeshDict`, which defines a block of a  $20 \times 8 \times 8$  cells per  $x, y, z$ . This means that we divide the domain into cells of  $0.08 \times 0.06 \times 0.06$  m. Since time in this course is short, let's stick with such a coarse background mesh.

### Boundaries

The dictionary entry `boundary` defines the following boundary patches of the domain:

- `inlet`
- `outlet`
- `slip`: far away faces (considered as frictionless walls)
- `symmetryPlane`: the root of the wing

While we specify some patches here as `slip` or `symmetryPlane`, we will define the concrete boundary conditions in the next task.

### Generating the background mesh

In the `Fluid` folder, run:

```
blockMesh
```

If you only want to look at the topology of the domain, without yet meshing it, you can run `blockMesh -write-vtk`. You can then visualize the `blockTopology.vtu` file in ParaView (Figure 2). You can enable the `Axis Grid` in the Properties tab.

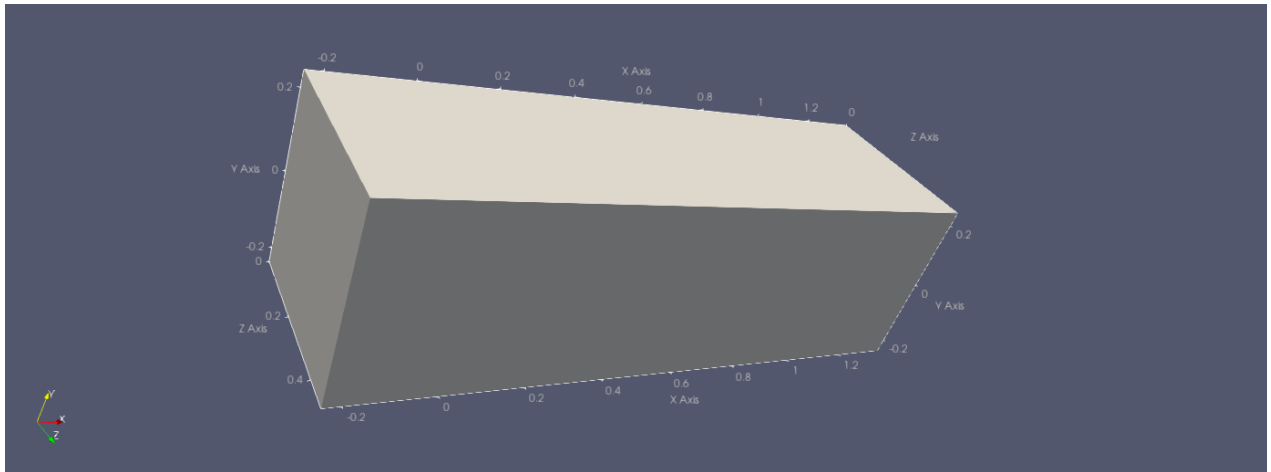


Figure 2: Topology of the domain in ParaView

### Configuring snappyHexMesh

Once we have generated the background mesh, we need to refine it and subtract the wing. The geometry of the wing must be a surface data file in STL format, either binary or ASCII, located in the `constant/triSurface`

subdirectory of the case directory. Copy the `naca2312.stl` there.

The mesh generation process in `snappyHexMesh` comprises three stages:

- **castellatedMesh**: performs cell splitting and removal
- **snap**: performs cell vertex points motion onto surface geometry
- **addlayers**: introduces additional layers of hexahedral cells aligned to the boundary surface

Each stage can be activated in the beginning of the `system/snappyHexMeshDict` file. We have enabled all of them.

There are a lot of parameters in the `snappyHexMeshDict` dictionary; we invite you to look at the comments in the file, at the references below, and at the documentation for further details. Let's focus on the most relevant for this exercise.

## Geometry

This is defined in the `geometry` subdictionary.

The geometry of the main elements of the mesh can be specified through an STL surface or geometry entities. Here we define our wing and two refinement regions:

- substitute `yourSTLfile.stl` with `naca2312.stl`
- notice that two refinement regions are defined:
  - `refineBox` around the wing
  - `wake` behind the wing

## Castellation

The castellation stage removes the cells inside (or outside) the specified geometry, resulting into a castellated (staircase-shaped) mesh. This is defined in the `castellatedMeshControls` subdictionary.

In the `refinementSurfaces` entry, substitute `DEFINETYPE` with `wall` under `naca2312/patchInfo`: We are telling snappy that our STL file is a boundary.

Notice the `locationInMesh` entry: This is an arbitrary point outside the wing and inside the initial mesh (any location in this region will do).

## Adding layers

Close to the boundaries, it is good practice to have additional layers of refinement. We define this in the `addLayersControls` subdictionary:

- under `layers`, substitute `yourSurface` with `naca2312` (i.e., the name assigned to your STL file in the `geometry` subdictionary)
- under `layers`, substitute the value `NL` with `3` at the `nSurfaceLayers` entry
- under `expansionRatio`, substitute `ER` with `1.0`: we want the layers to be of the same height.

Now your `snappyHexMeshDict` is complete. As we want to perform this expensive operation in parallel, we also need to define how to decompose the domain.

## Domain decomposition

Open the `decomposeParDict` file in the `system` directory and substitute `ND` with `8` in `numberOfSubdomains`: this is the number of subdomains in which your case will be decomposed, and it should typically not exceed the number of cores of the system. This number needs to agree with the number of subdomains per direction, defined in `hierarchicalCoeffs` (in this case, `4x1x2`).

Decompose the domain by running (in the case directory):

```
decomposePar
```

Eight directories with names `processor[0-7]` will be generated, including configuration files similar to the ones defined for a single-process case.

## Generating the mesh

Now we can create the mesh for the eight subdomains in parallel:

```
mpirun -np 8 snappyHexMesh -parallel
```

This will take a few minutes to complete.

In case you get an error that there are not enough slots in your system to run eight processes, reduce the `numberOfSubdomains` in `decomposeParDict` (adjusting the subdomains per direction) and the number of processes in `mpirun` accordingly. If you still want to execute eight processes, you can pass the `--oversubscribe` option to `mpirun`. This is then expected to take significantly longer.

## Reconstructing the mesh

Once `snappyHexMesh` has finished, you can reconstruct your domain from the decomposed ones by running `reconstructParMesh`.

When finished, you will see three time folders (0.001, 0.002, 0.003) in the root directory of the case. Each one corresponds to a stage of `snappyHexMesh` (Figures 3, 4, 5). The time step size depends on the `deltaT` parameter in the `controlDict` file, but it is not relevant. You can obtain the final mesh in the `constant` directory, without the intermediate steps, by adding the `-overwrite` option to `snappyHexMesh`. In the next step (fluid simulation), we will use and rename the 0.003 directory.

## Checking the mesh

`checkMesh` can give us some mesh quality metrics (in particular, whether there are any distorted cells):

```
checkMesh -latestTime
```

If everything goes well, you should see a `Mesh OK.` at the end.

## References

Most of the information is taken from [this training presentation of Wolf Dynamics](#) (with permission).

You can also consult the [official documentation](#)



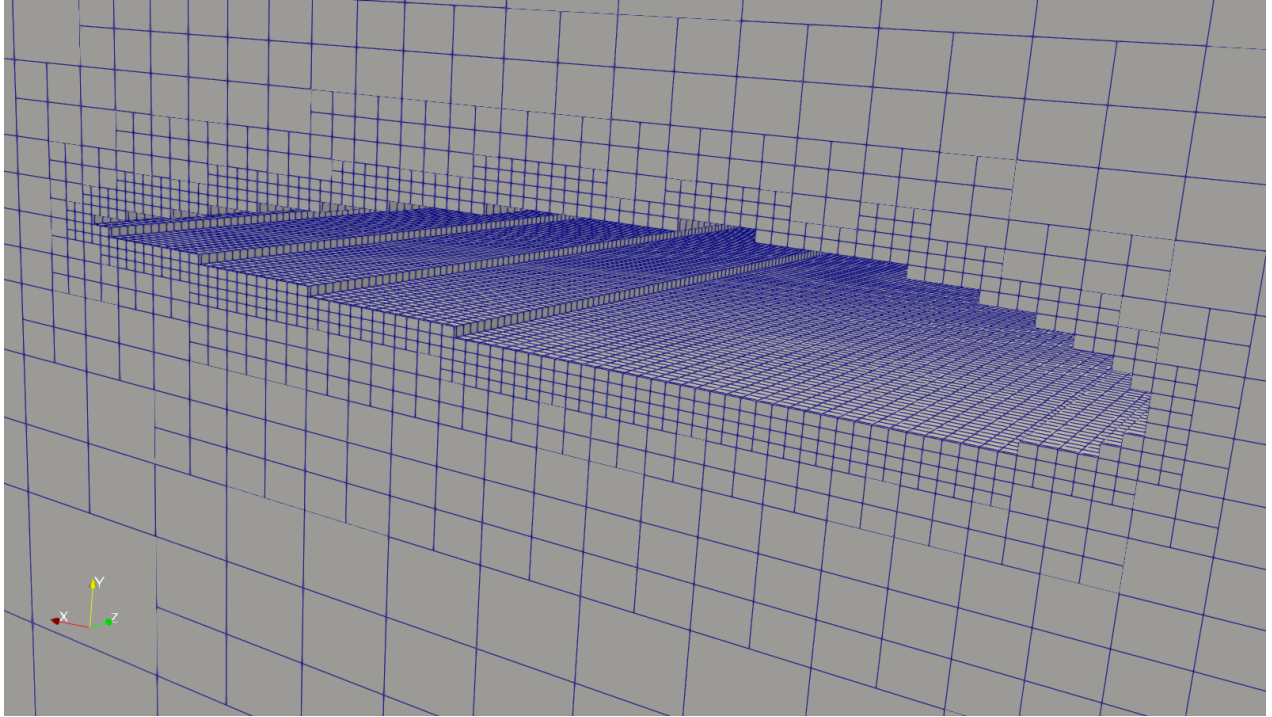


Figure 3: snappyHexMesh output: Castellation stage (0.001 directory)

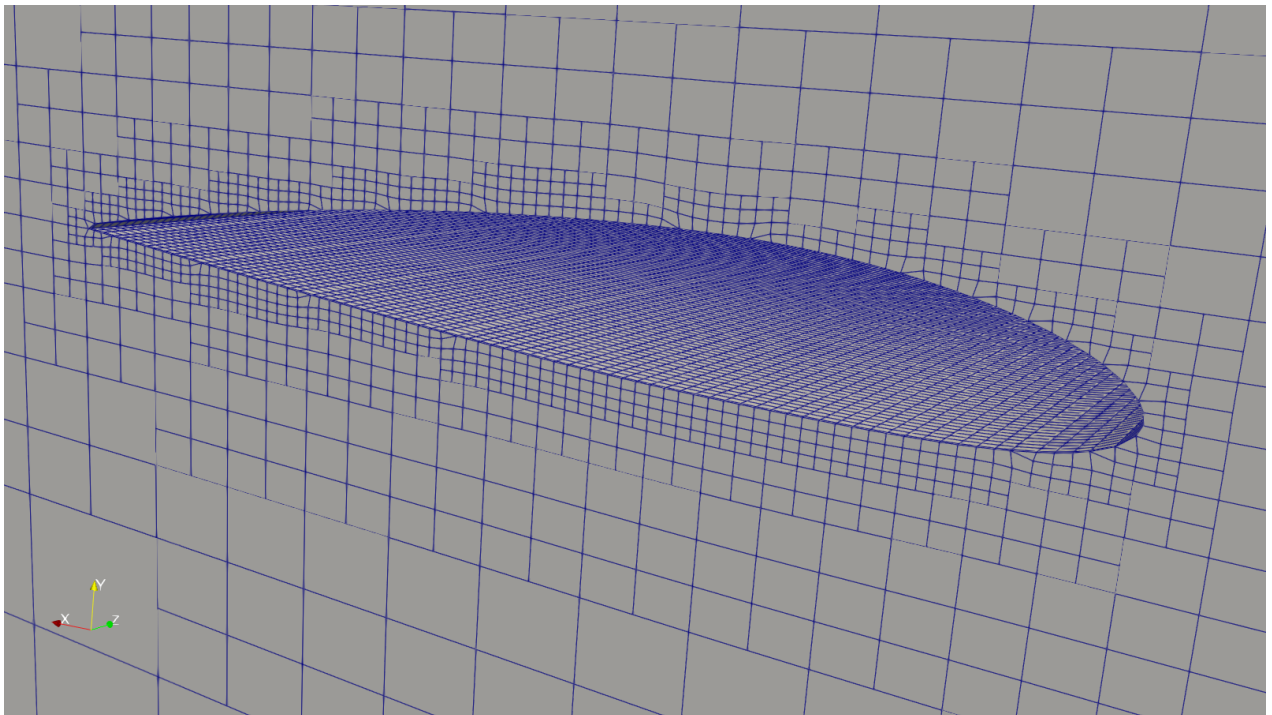


Figure 4: snappyHexMesh output: Snap stage (0.002 directory)

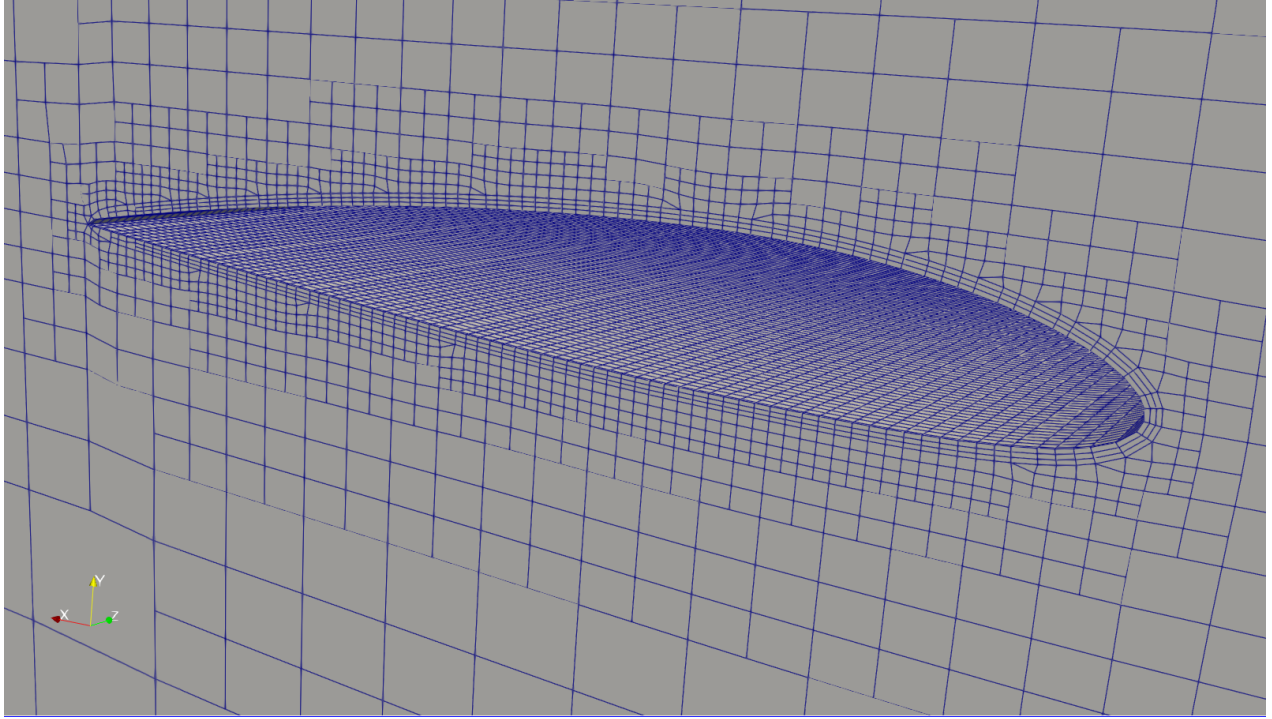


Figure 5: snappyHexMesh output: Boundary layer stage (0.003 directory)

## Task 4: Simulation of the Fluid domain

In this section we'll use the previously generated Fluid Mesh to perform a single physics flow simulation. Besides checking the validity of the model, this allows us to obtain an initialized fluid domain for the FSI simulation.

We start here from a steady-state simulation. Later, in the FSI part, we will switch to a transient simulation. See a general overview of this task in Figure 1.

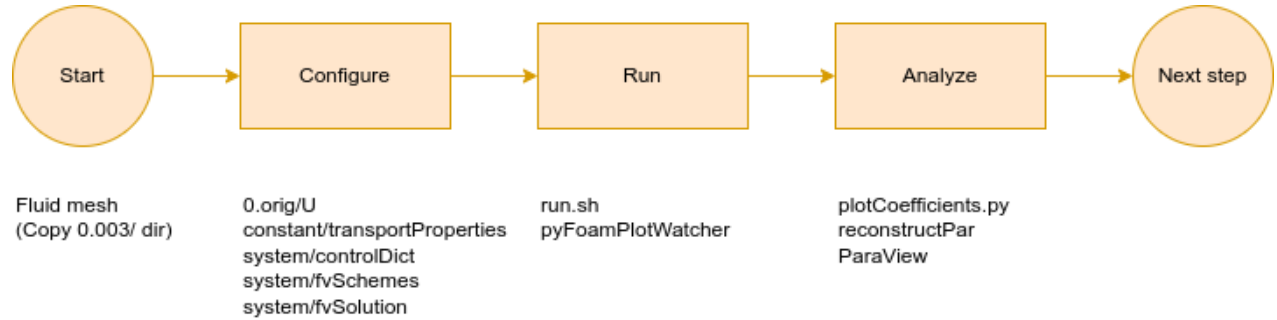


Figure 1: Fluid simulation: General overview

### Configuration

In the following, we will simulate a scenario with incompressible, laminar flow of water with:

- $U_{\infty} = 0.5 \text{ m/s}$
- $\rho = 1000 \text{ kg/m}^3$
- $\nu = 1 \cdot 10^{-6} \text{ m}^2/\text{s}$
- $Re = \frac{U_{\infty} c}{\nu} = 5 \cdot 10^4$

The solution also includes a scenario with air.

### Boundary and initial conditions

The new folder `0.orig/` contains the boundary and the initial conditions for each of the simulation variables: files `U` and `p`.

Open the file `U` and:

- Substitute `UINF` in the `internalField` dictionary entry with the value `0.5`. This initializes the whole domain to  $U_{\infty}$
- Substitute the boundary condition `BOUNDARY` for the `naca2312` patch in the `boundaryField` entry with `noSlip`.

Note: we use the folder `0.orig` instead of the usual folder `0` just in case the simulation overwrites the initial conditions (e.g., you execute `potentialFoam` to initialize the fluid domain). The run script copies `0.orig` to `0`.

### Mesh and model properties

In the `constant/` directory:

- Copy here the `polyMesh` folder, from the `0.003` folder or the previous task.
- In the `transportProperties` file, replace the `NU` (kinematic viscosity) with `1e-06`.
- In the `turbulenceProperties`, we already define a `laminar` simulation.

## Simulation control and numerics

In the `system/` directory, we define numerical properties and other options regarding the simulation execution. The files one typically needs to configure are `controlDict`, `fvSchemes`, and `fvSolution`.

- Have a look into the `controlDict` file and:
  1. Substitute `END` with `250` at `endTime` entry: we will perform 250 steady-state iterations at most.
  2. Substitute `RHO` with `1000.0` in the `forces_object` and in the `forceCoeffs_object`. These function objects compute some forces we will later analyze. We have already pre-filled further parameters (`magUInf`, `lRef`, `Aref`).
  3. Notice that we define `simpleFoam` as `application`. This is a steady-state solver implementing the [SIMPLE algorithm](#).
- Open `fvSchemes` and substitute `SIMULATIONTYPE` with `steadyState`.
- Open `fvSolution` and in the `residualControl` entry set the thresholds for earlier exiting the steady-state simulation:
  1. Substitute `P_RES` with `1e-4`
  2. Substitute `U_RES` with `1e-4`

## Running the case

In order to run simulation, open a terminal from the `Fluid` folder and type:

```
./run.sh`
```

This script:

- Copies `0.orig` into `0`
- Decomposes the case
- Runs `simpleFoam` in parallel and logs the output in `log.solver`
- Reconstructs the latest time step

By default, the script runs the case with 8 processes, using over-subscription. You can change the partitioning by changing the `system/decomposeParDict` and you then change the number of processes in `run.sh`.

The simulation will probably take around 5 min to complete all 250 iterations. We still get a pretty much converged state, even if the residuals in this case don't reach the limits which would automatically stop the simulation.

Use the script `clean.sh` if you need to start from scratch.

## Monitoring

To check the simulation progress and plot the residuals over time, you can use [PyFoam](#):

```
pyFoamPlotWatcher log.solver
```

A pop-up window with residual graphs should appear, as in [Figure 2](#). Note that there are incompatibility issues on some systems. If this does not work out for you, just move on for now. For roughly monitoring the progress of the simulation, you can also just look at the names of the result directories generated.

## Analyzing the results

In order to understand if your simulation has converged and if you have obtained reasonable results, you can look at the output of the `functions` that we enabled in the `controlDict` dictionary. You can plot force coefficients over time with:

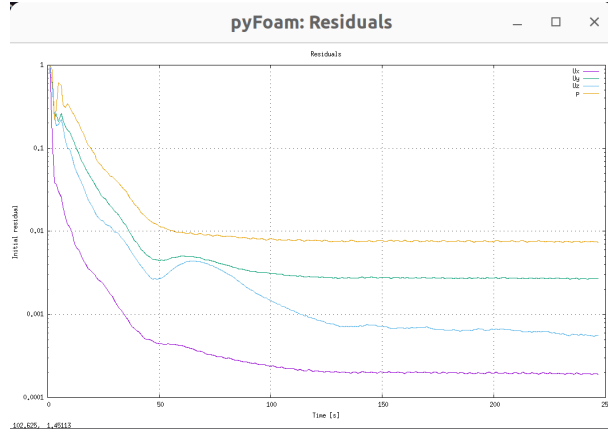


Figure 2: Fluid simulation: Monitoring residuals with PyFoam

```
python3 plotCoefficients.py
```

A pop-up window should appear, as in Figure 3.

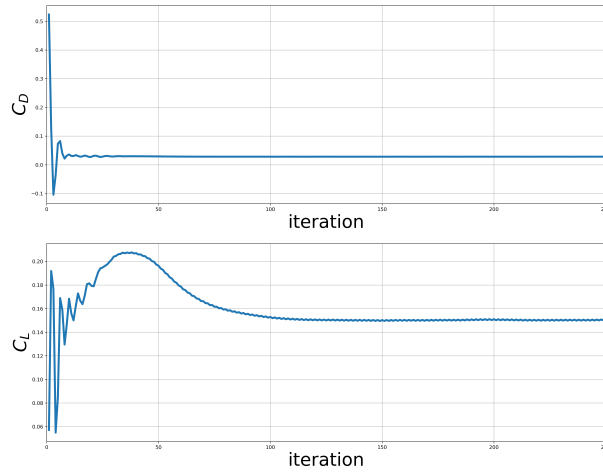


Figure 3: Fluid simulation: Monitoring the Cd/Cl coefficients

You could compare these values with theoretical data (not available), or you could perform some mesh-convergence study to check the convergence of your setup.

### Reconstructing the case

Your case is decomposed into 8 subdomains. You can still view the results in ParaView by selecting **Decomposed Case** once you opened **Fluid.foam** (Figure 4).

In our case, we only need the latest time step (250), which will be the initial state of our coupled simulation. We can reconstruct the decomposed case for this time step:

```
reconstructPar -latestTime
```

There should now be a **250/** directory containing the files **U**, **p**, and **phi**. Move these files into **results/water/**, overwriting the currently empty files. We will use these results as initial state for the FSI simulation.

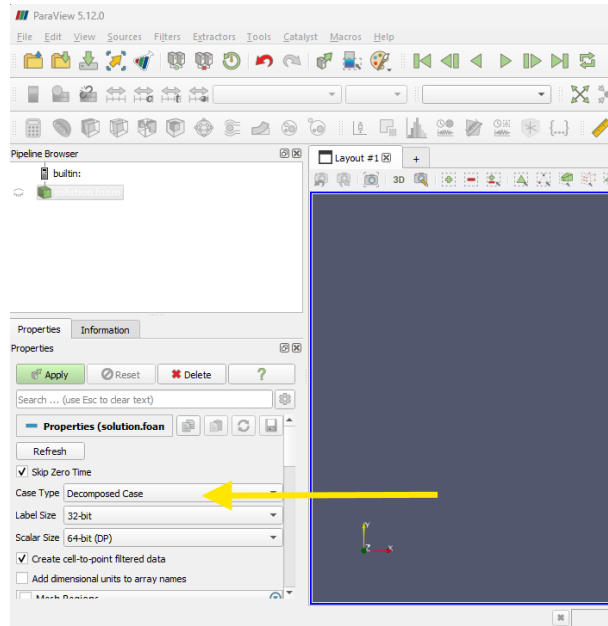


Figure 4: Fluid simulation: Select decomposed case in ParaView

### Alternative setup: air (optional)

Now we consider a laminar incompressible simulation in air, with the same Reynolds number. The main parameters are:

- $U_{\infty} = 7.5 \text{ m/s}$
- $\rho = 1.225 \text{ kg/m}^3$
- $\nu = 1.5 \cdot 10^{-5} \text{ m}^2/\text{s}$
- $Re = \frac{U_{\infty} c}{\nu} = 5 \cdot 10^4$

For this:

- Use `clean.sh`. It removes the following:
  - `0` folder
  - `processor*` folder
  - `postProcessing` folder
- Remove the `250` directory
- Update the simulation values
- Rerun the simulation
- Reconstruct the case and move the `250` directory in the `results/air/250` folder

## Task 5: FSI simulation

Finally, we are now able to put everything together and start our fluid-structure interaction simulation. We'll use all the work done until now to configure the Fluid and Solid participants. The starting point of our case is in the `skeleton` folder, which is the root of our FSI case. This includes each participant to a separate directory, and the `precice-config.xml` file in their common parent directory.

See a general overview in Figure 1.

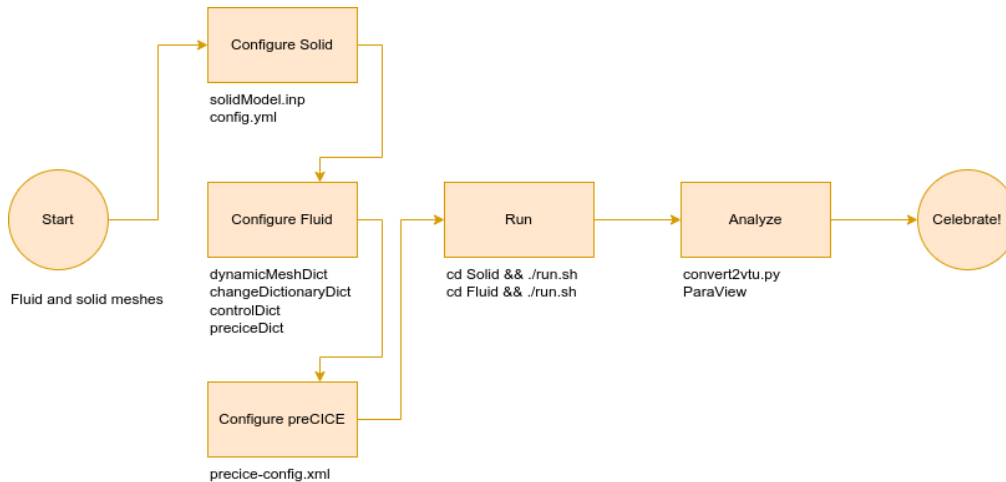


Figure 1: FSI: General overview

### Solid configuration

Copy the `wing2312_m.inp` mesh from `01_solidMesh` into the `Solid` directory. Remember to use the one converted to *meters*. Then, we can adjust the model and configure the CalculiX-preCICE adapter.

### CalculiX configuration

Check the following changes in `solidModel.inp`:

- Section `*DYNAMIC`: we perform a simulation 0.2 seconds long with a time-step of `1ms`
- Section `*AMPLITUDE`: we ramp the loading of the wing, starting with the 5% of the total load, arriving at 100% after `0.1s`
- There is no section `*DLOAD` anymore, but there is now a `*CLOAD` section.

Complete the file with the following information:

- section `*CLOAD`: replace each of the `WETSURF` entries with the name of the group given to the **wet surface** (`NwetSurface_Nodes`)

### CalculiX adapter configuration

The file `config.yml` is specific to the [CalculiX adapter](#). The information here must match the information in `precice-config.xml`.

The entries of this file specify the path to the preCICE configuration file, the coupling mesh (`Solid-Mesh`, a nodes-based mesh defined in the preCICE configuration file), the read data (forces), and the write data (absolute displacements). In this file:



- Replace WETSURF with the name of the group given to the *wet surface* (see the mesh .inp file), **WITHOUT the N at the beginning** (i.e., wetSurface\_Nodes).

The Solid participant is now ready, and we can move to the Fluid participant.

## Fluid configuration

For the Fluid participant, we will use the [OpenFOAM-preCICE adapter](#).

Copy the previously generated mesh: copy the polyMesh folder from your 0.003 folder in 03\_fluidMesh/skeleton, or from the constant folder in 04\_fluidSimulation/skeleton/Fluid. We can then adjust the setting that are specific the coupling.

## Mesh motion

We are using the ALE approach to FSI. This means that OpenFOAM applies a displacement vector (mesh motion) on the otherwise static mesh (the number of cells and connection between points remains the same).

In the constant/dynamicMeshDict, replace WETSURF with the name given to the wing patch (naca2312, specified in constant/polyMesh/boundary).

In the 0.orig/ folder, you can also find a new dictionary file pointDisplacement, required by the mesh motion solver.

## Initial state

In task 4, we obtained an steady-state solution, which we wanted to use as initial state here. Copy the files U, p, and phi from 04\_fluidSimulation/skeleton/results/water/250 into the 0.orig/ directory.

## Boundary conditions

When we performed the fluid simulation, we defined the surface of the wing as noSlip in the 0.orig/U file, which has also been applied to the results we just copied. In FSI simulations, we need that the velocity is overwritten by the OpenFOAM-preCICE adapter, for which we need to use the movingWallVelocity boundary condition. To avoid opening such a large file, we can use the utility changeDictionary:

- Open the file system/changeDictionaryDict.
- In the boundaryField dictionary entry, replace PATCH with naca2312 and TYPE with movingWallVelocity.

We will update the 0.orig/U file automatically before running the coupled simulation (see how in prepare.sh).

## Simulation control

Open the controlDict file and:

- Notice that, compared to the previous, steady-state flow simulation, we are now using the transient pimpleFoam solver.
- Replace the entry DT for the entry deltaT with 1e-3 (fixed, as adjustTimeStep is disabled)
- Notice how we are enabling the adapter as a function object at the end of the file.

## OpenFOAM adapter configuration

The OpenFOAM adapter configuration file is system/preciceDict. In this file:

- Replace the entry PATCH in the Interface1 dictionary entry with the name given to the wing boundary patch (naca2312)



- Replace the entry `RHO` in the `FSI` dictionary entry with the water density ( $\rho_{water} = 1000.0 \frac{kg}{m^3}$ ). We are using an incompressible solver, so the adapter needs a density value to compute the forces.

## preCICE setup

Once we have prepared the two participants, we can now also configure preCICE, i.e., the coupling itself. See a visual overview of the preCICE configuration in Figure 2.

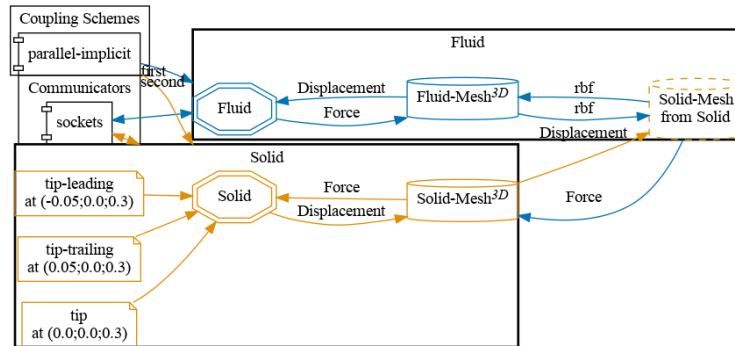


Figure 2: FSI: Visualization of the `precice-config.xml`

In the `precice-config.xml` file:

- Replace DT in the `<time-window>` tag with 0.001
- Replace TFINAL in the `<max-time>` tag with 0.2
- Replace the two occurrences of REL\_CONV in the `<relative-convergence-measure>` tag with 1e-3

NOTES:

- We are considering three watch-points at the tip of the wing, so that we can look at the displacement and at the pitching angle of the final section of the wing:
  - *tip mid-chord* at coordinates 0.0; 0.0; 0.3
  - *tip leading edge* at coordinates -0.05; 0.0; 0.3
  - *tip trailing edge* at coordinates 0.05; 0.0; 0.3
- All the simulation components share the same  $\Delta t$  and  $t_{final}$ .
- The convergence measure that we chose is a good compromise between accuracy and execution time.
- We are using the same  $\Delta t$  for the Fluid and the Solid part, which is also the same as the coupling time window here. This means that the two participants are *not subcycling*.

## Running the coupled simulation

Now we are ready to perform the coupled simulation.

As expected, FSI simulations take a long time. If you are short in time, just read through the instructions and continue with analyzing the provided results in the `solution/` directory.

## Solid participant

Open a terminal and enter the **Solid** folder. Here you simply run the **run.sh** script:

```
./run.sh
```

This starts CalculiX as the **Solid** preCICE participant. The Solid participant should now start and wait for the Fluid participant to appear as well.

## Fluid participant

Open another terminal and enter the **Fluid** folder. Here you have to:

- Run `./prepare.sh`, which:
  - copies `0.orig` into `0`
  - uses `changeDictionary` to switch the boundary condition from `noSlip` to `movingWallVelocity`
  - decomposes the case into **8** subdomains
- Run `./run.sh` to start the parallel simulation (this will take several minutes)
- After the simulation completes, run `./post-run.sh` to remove some empty result directories, which are created for technical reasons but are making further analysis trickier.

In case the 8 subdomains are too many for your system, see the related notes in Task 4.

## Monitoring

You can monitor the ongoing simulation by running the following scripts:

- `./plotDisplacement.sh`: Plots the displacements over time (exported as watch-points).
- `python3 ./plotConvergence.py`: Plots the number of iterations and the relative error for each time step.

Two windows with the following graphs should appear (Figure 3 and 4). The simulation ends after 200 time steps (at  $t=0.2s$ ).

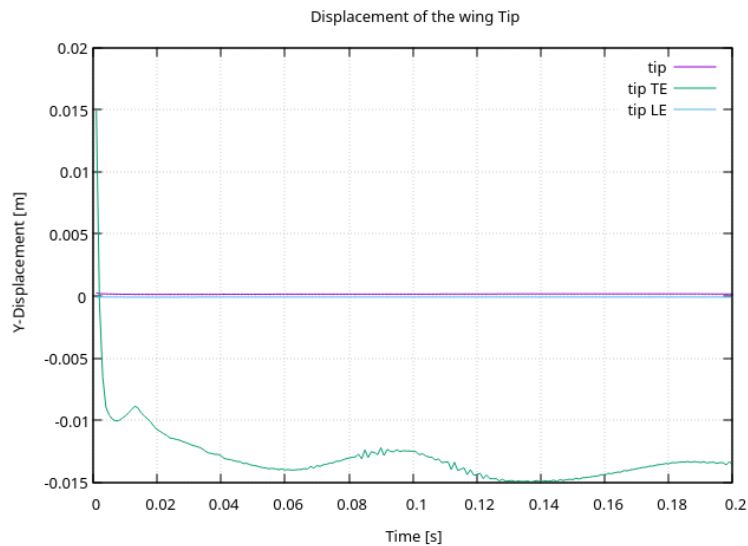


Figure 3: Monitoring: Tip displacement (`plotDisplacement.sh`)

## Cleaning

In case you need to remove the results and log files before starting your simulation again, use the `clean.sh` script of each participant case.

## Results

In order to open the results of both participants in the same ParaView window, we first need to convert the CalculiX results file `solidModel.frd` to a format compatible with ParaView:

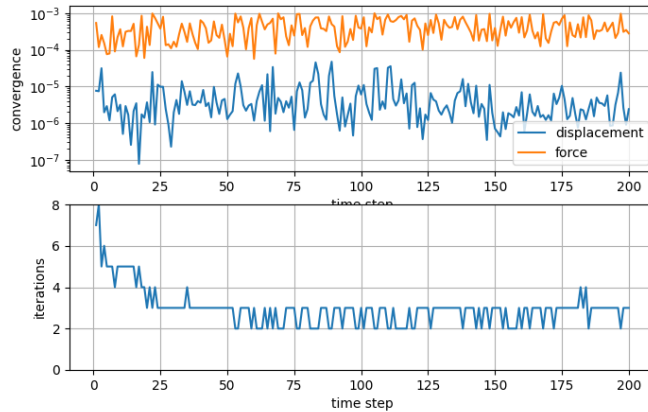


Figure 4: Monitoring: Convergence (`plotConvergence.py`)

```
python3 convert2vtu.py
```

This will create a `convert` folder where you will find one `.vtu` file for each exported time step.

In case you need to synchronize the results of the two participants, you can apply a `TemporalShiftScale` filter in one of the two data sets in ParaView. See also the [visualization documentation page](#).

### Fluid results

Open a terminal in the `Fluid` folder and type `paraFoam`. This will create an (empty) `Fluid.foam` file and open a ParaView window (Figure 5). In `Case` type select `Decomposed Case` and press `Apply`. (Figure 5)

### Solid results

In the same ParaView window, select `File->Open...` and point to the `Solid/convert/solidModel.pvd` file (Figure 6). To see the displacement more easily, you can apply a `WarpByVector` filter, using the displacement (`U`) as vector, and a scale factor of your choice.

### Alternative setup: air (optional)

In the folder:

```
./05_FSI/solution/FSI_air
```

you can find another case with other physical properties, both for the fluid (air) and for the solid (close to propylene). The Reynold number is still  $5 \cdot 10^4$ . We leave it to you as a starting point to change some physical properties or experiment.

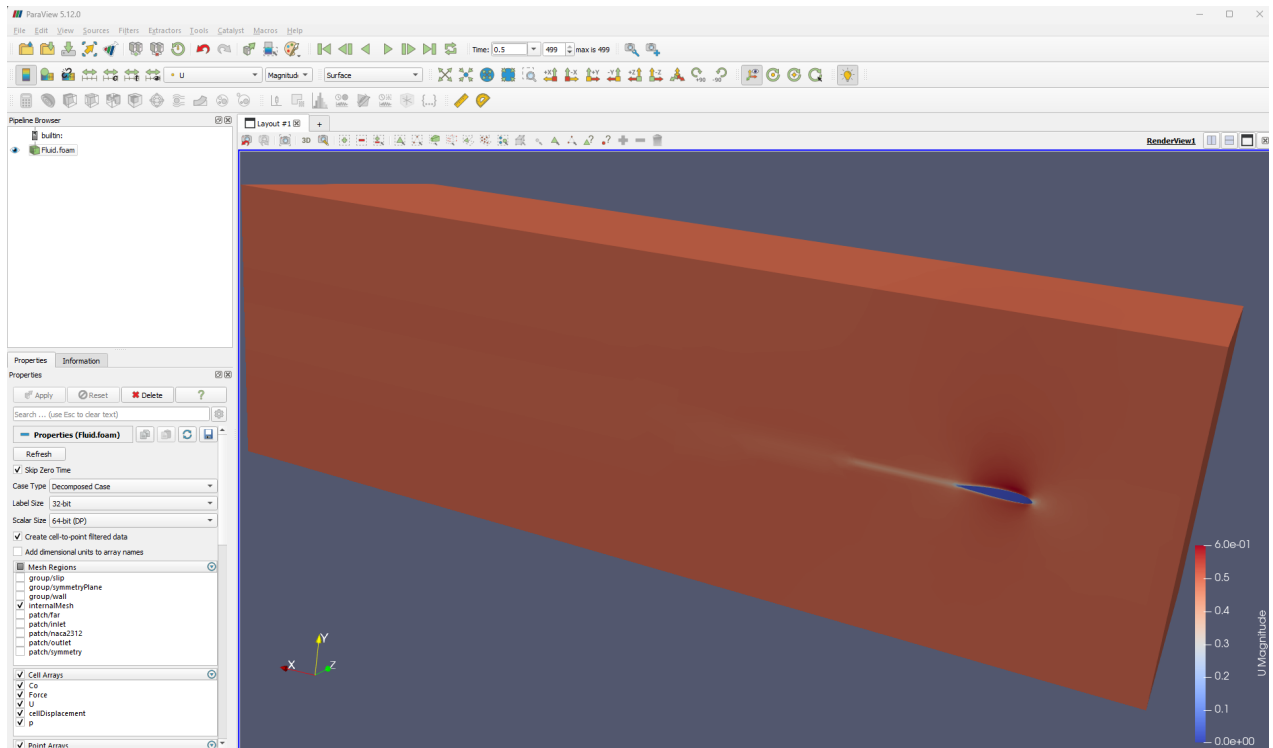


Figure 5: ParaView: Surface plot of flow velocity

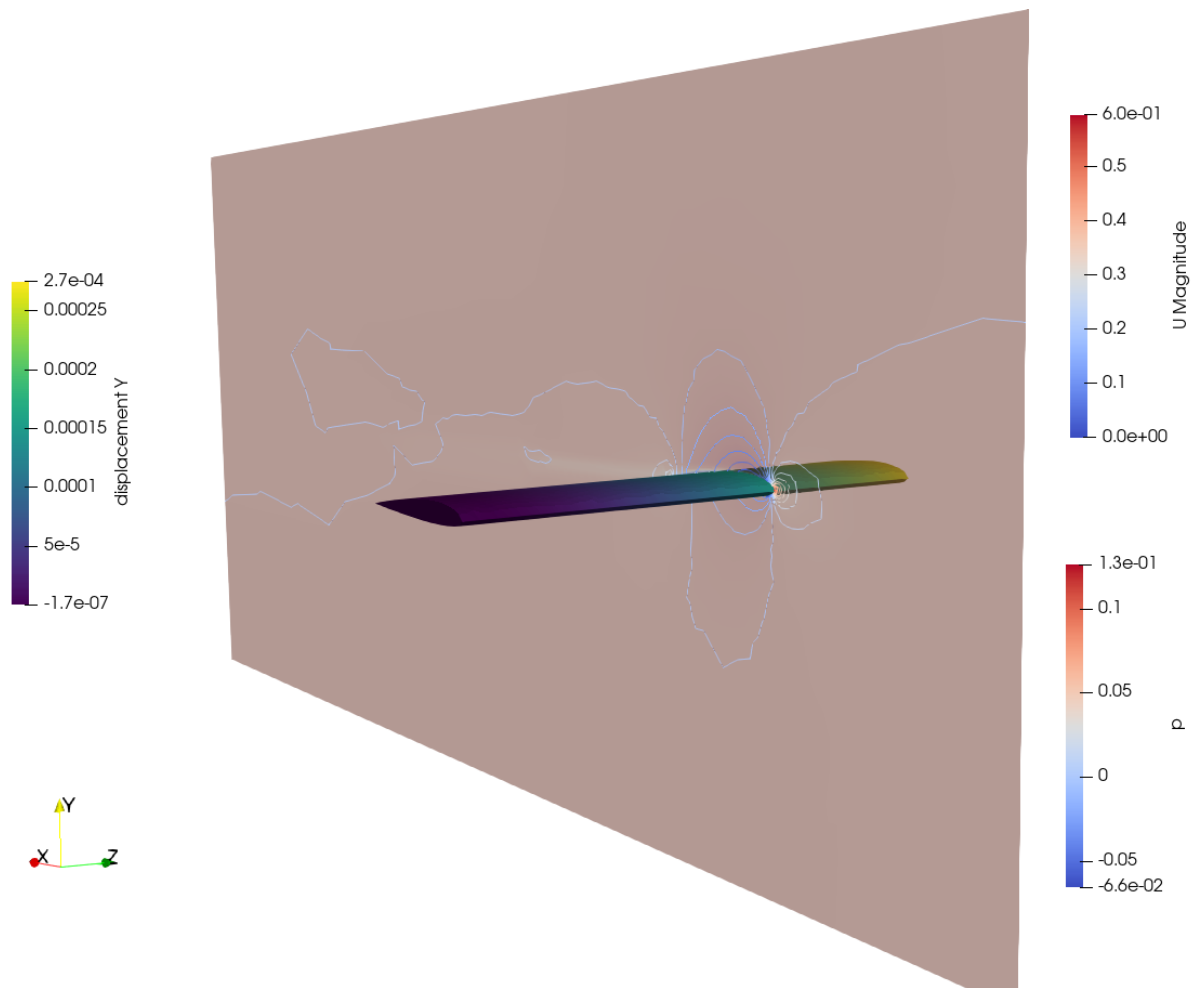


Figure 6: ParaView: Combined FSI results