An internal flows problem in OpenFOAM

Consider the geometry given in Figure 1. Assume an inlet velocity of 7.5 m/s, fluid density of 1 kg/m^3, and kinematic viscosity of 1.5 m^2/s. Use snappyHexMesh or cfMesh to mesh the provided stl files (depicted above). Set up a simpleFoam simulation to process and report the following results.

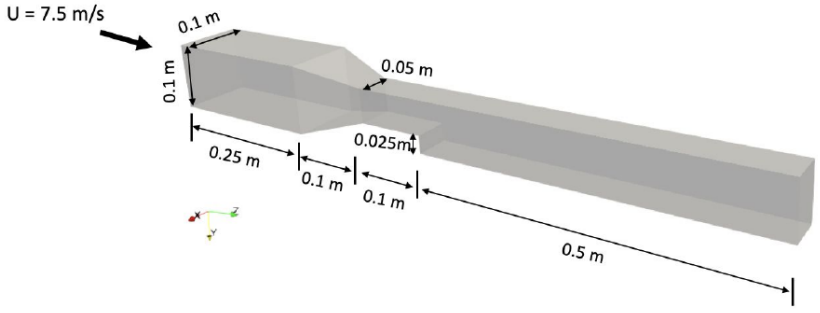

Figure 1: Geometry of the problem

Disclaimer: results presented in this report are for the sole purpose of demonstration and are not validated.

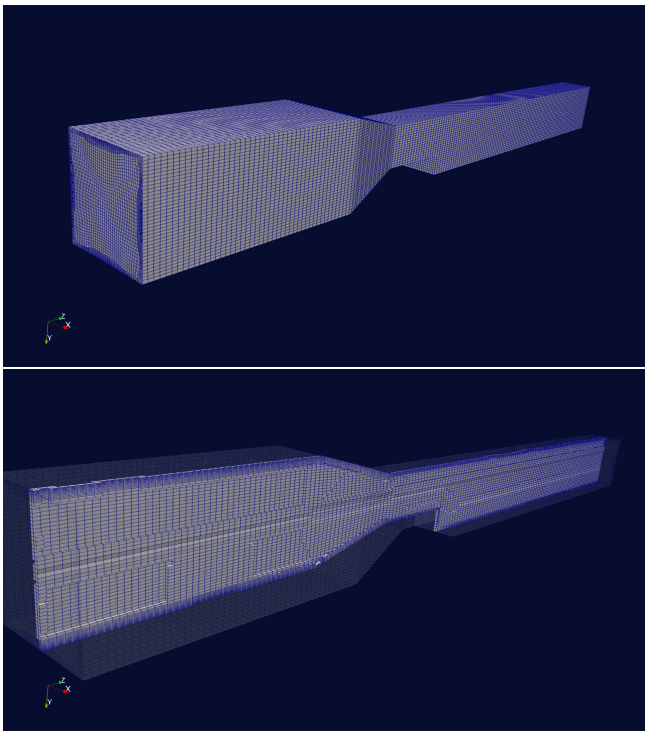

Figure 2: Visual details of the generated mesh for the input geometry via `snappyHexMesh`

- Total pressure drop across the domain
  This is reported by the `Allrun` script, see also Figure 3

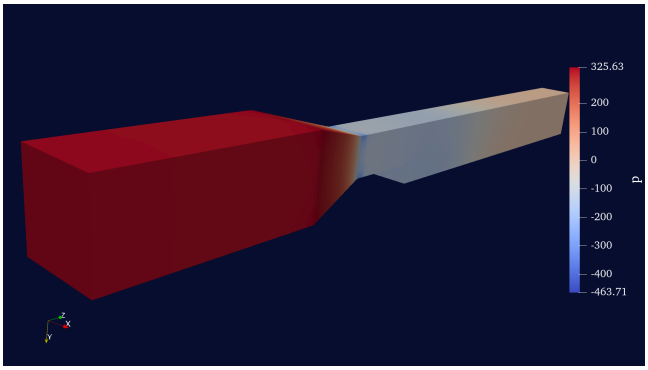

Figure 3: Pressure distribution in the domain

- Reattachment point after the backward facing step

Reattachment point is computed through the wall shear stress and when $\frac{\partial u}{\partial z}$ changes sign, see also Figure 4.

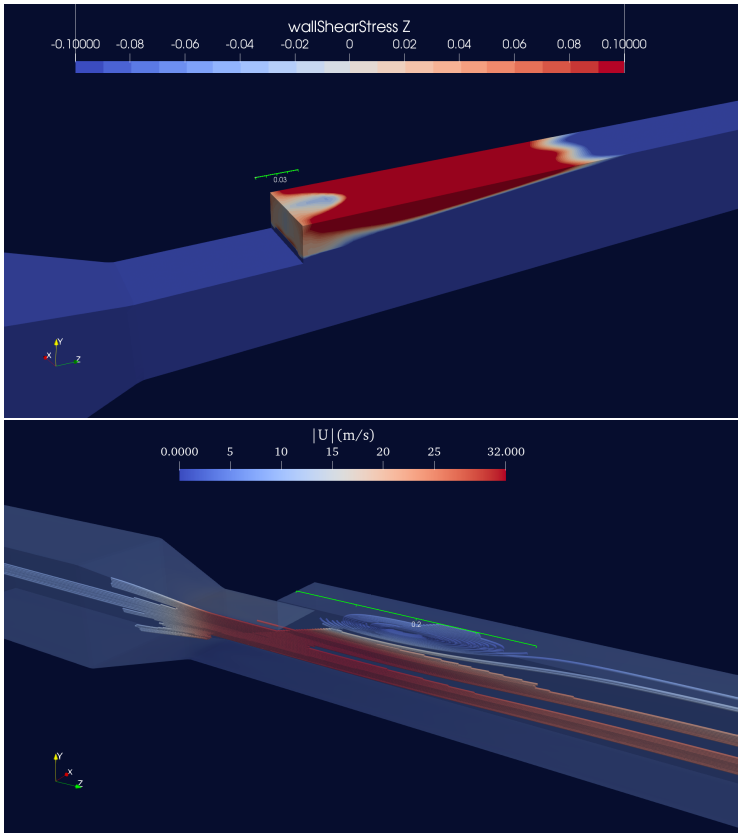

Figure 4: Estimation of the reattachment length through the wall shear stress (top), and Illustration of the size of the main vortex through velocity streamlines (bottom).

- Streamlines emanating from the inlet of the domain See Figure 5 which can be generated from `streamLines` post processing utility, see also the `Allrun` and `system/controlDict/streamLines` scripts.

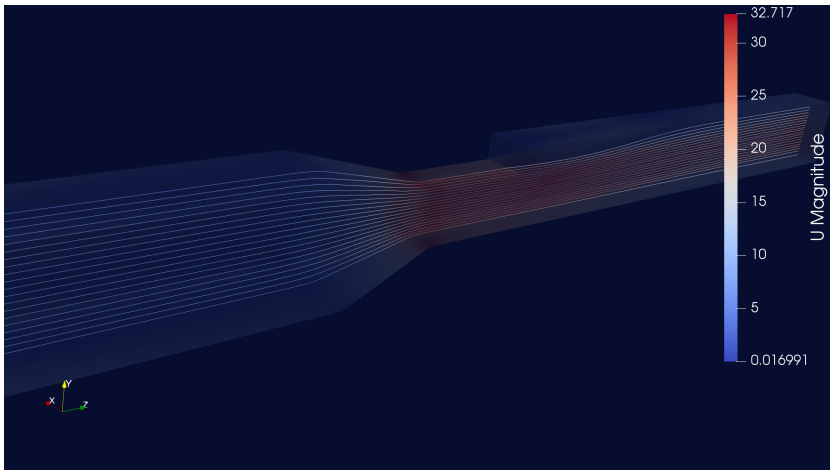

Figure 5: Streamlines emanating from the inlet of the domain

Key Questions

- How does the meshing procedure and set up change if only the total pressure drop is desired? If only the total pressure drop is desired, there is not any real post-processing involved and we can just rely on `system/controlDict/pressureDifferencePatch` to compute the pressure drop and ignore the command line inputs of `Allrun` script related to streamLines. The meshing procedure and the rest of the set up do not change

- How would the meshing procedure change for different inlet velocities and fluid properties? Depending on the flow regime (laminar/turbulent) that is dictated by the flow velocity and the fluid properties such as density and viscosity through the Re number, the underlying mesh may not need to include very thin boundary cells at the walls (laminar flow) or may need to be extra conservative on the size of the closest cell to the wall (turbulent flows where wall functions are used). In this case, one can compute the necessary spacing through y^+ calculators (see this link for instance) and apply the necessary spacing in the `addLayersControls` section of `system/snappyHexMeshDict`.

- How would you encapsulate this meshing procedure into a software workflow to automatically write the requisite meshing files? More specifically, what CAD, fluid, and boundary inputs are needed? And what output meshing parameters are they used to determine? Sorting out all the meshing criteria will be a pre-processing stage to meshing. In such a stage, the bounding volume of the domain is computed (through the CAD

files) with an approximate size of cells that matches a user-defined mesh resolution and is subsequently used in `system/blockMeshDict` for creating the background mesh. Inlet and outlet patch(es) are asked to be imported by the user via ply/stl files and are fed into lists that consequently make it into the `geometry` section of `system/snappyHexMeshDict`. The inlet/outlet property of each (set of) patch(es) are translated into the boundary conditions defined in `0.org` files for pressure, velocity and turbulent variables.

- What are other features and capabilities we can build leveraging snappyHexMesh, cfMesh, and/or OpenFOAM to simplify simulations for design engineers? In many cases, design engineers do not exactly know the final setup in advance and look for some general guidelines. In these cases, it will be very valuable to allow users to define a range for a parameter. For instance, an engineer may be looking to select a pump for an application for which the domain geometry is not fully specified. This is a very tedious task even for an experienced CFD engineer. In this scenario, making it easy for them to run a batch of simulations on the cloud in parallel to find an optimum design is very valuable and powerful. OpenFOAM and its many applications and utilities makes it easy to leverage CFD for such as tasks.