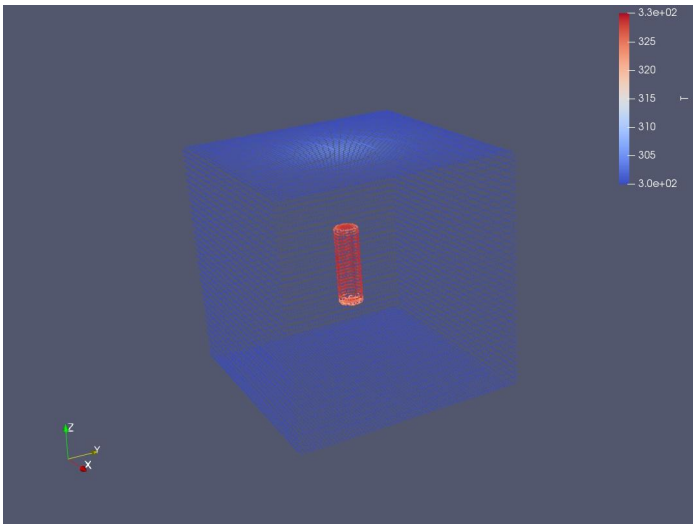
NATURAL CONVECTION

SETUP: There is a hot vertical cylinder (length=10, radius=1.5) hanging in a cubical room of dimension 30 * 30 * 30. The surface of the cylinder is maintained at the temperature of 330K and the walls of the room are maintained at 300K, also the air in the room is at the temperature of 300K. The cylinder is positioned hanging in the middle of the room. The following figure shows the geometry.



SIMULATION:

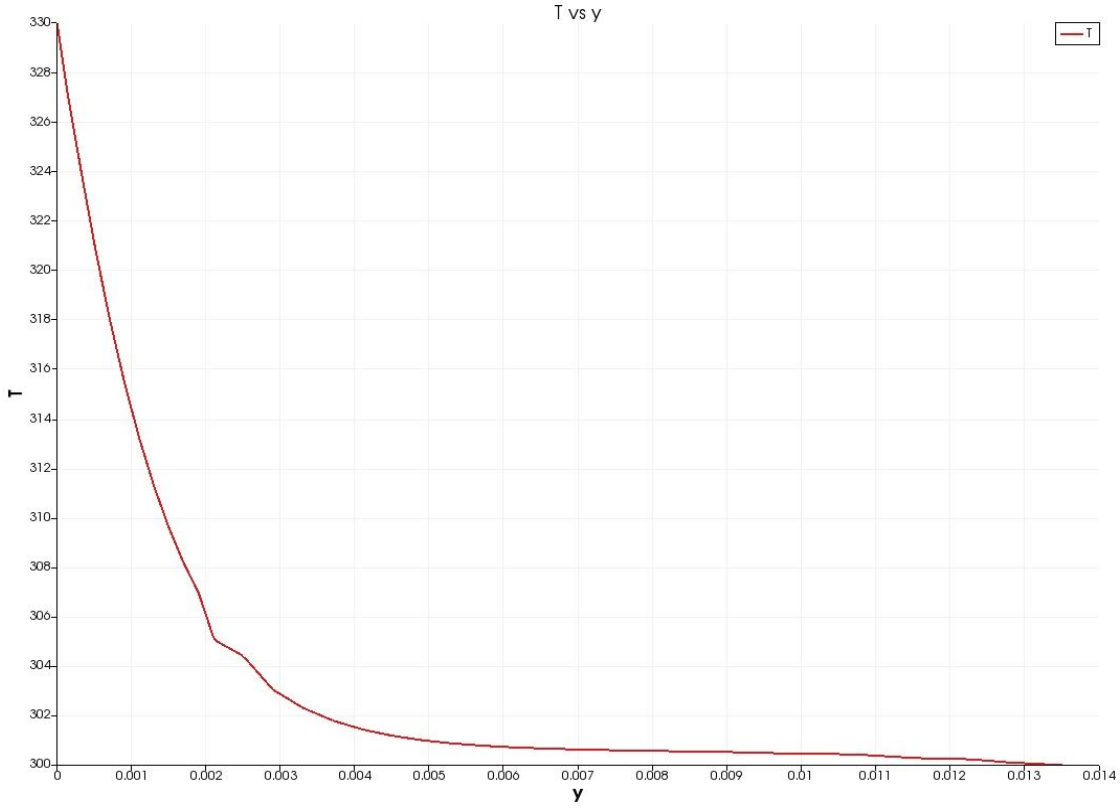
- **Directory structure:**
  - **0/:** Contains the initial values of all the scalar/vector fields. In this simulation, we are using RAS (Reynolds Averaged Simulation) as the simulation type and kEpsilon as the model. We have used the hotRoom tutorial of OpenFOAM as our base case, you can learn more about this tutorial on the OpenFOAM wiki page. This folder contains all the necessary parameters used by the kEpsilon model.
  - **constant/:** This folder contains all the constants required by the solver, every solver has its own set of parameters which need to be specified in order to use the solver. Also, the mesh created using blockMesh or snappyHexMesh or any other tool is stored in the constant folder. In this case, it contains ‘g’ which specifies the gravity direction, and ‘momentumTransport’ specifies the simulation type and the model to be used.
  - **system/:** This folder contains a snappyHexMesh file, a blockMeshDict file, a controlDict file, and fvSolution and fvScheme files. The blockMeshDict file contains specifications required by the blockMesh application of OpenFOAM to generate a mesh of the required geometry. The fvSolution contains the solvers to be used to solve for each of the fields, and the fvScheme specifies the various solution schemes to be used by the solver. The controlDict file specifies some simulation parameters like startTime, endTime, deltaT, etc and also you can

define function objects to calculate field average, max/min, etc during the run of the simulation.

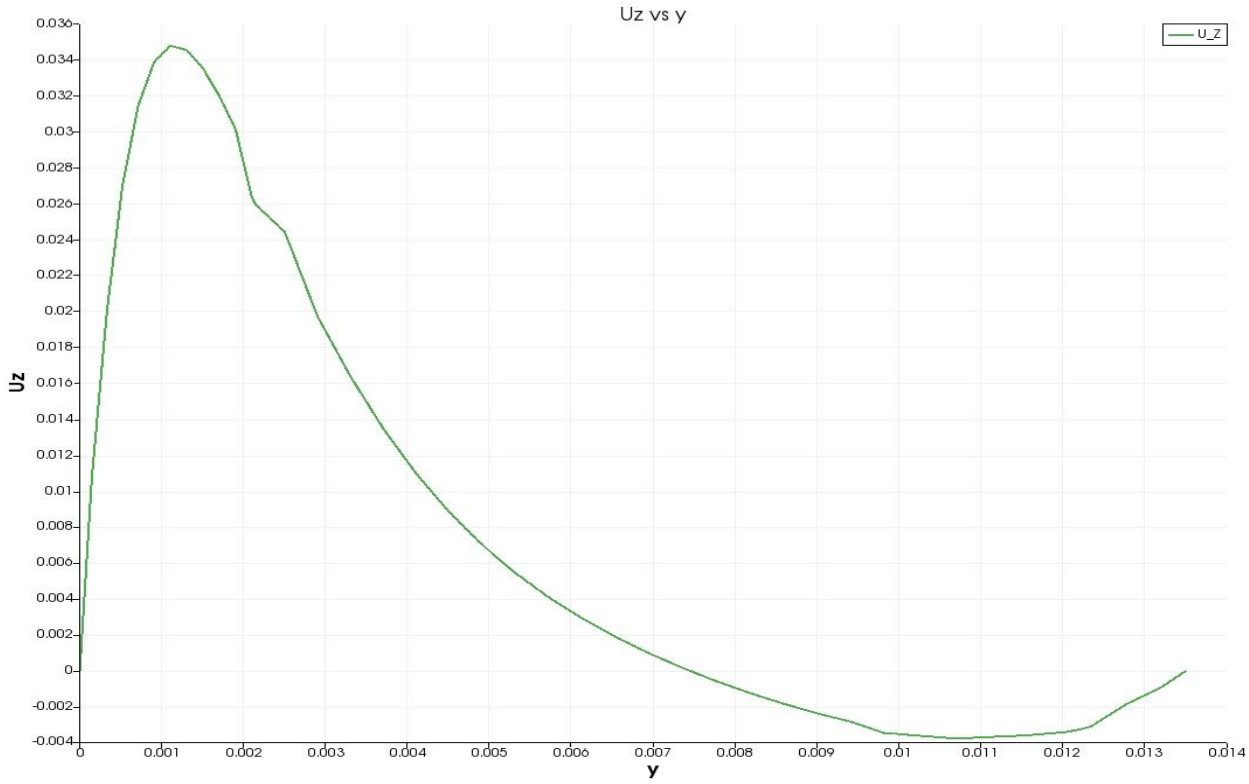
- ➢ **Meshing:** We have used snappyHexMesh to generate the mesh, as it seems quite suitable for this case. For snappyHexMesh, we generally need the stl files of the geometry. But in our case, since we need standard shapes only, we can use the 'searchableBox' and 'searchableCylinder' features of the OpenFOAM to create our geometry.
- ➢ **Solver:** For this simulation, we need **buoyantSimpleFoam,** as it is most suitable for the compressible heat transfer case. It is a steady-state solver, which can handle turbulence as well. It uses the SIMPLE (Semi-Implicit Method for Pressure Linked Equations) algorithm to solve the equations. The algorithm involves the following iterations:
    - Advance to the next iteration $t = t_n+1$
    - Initialise u_{n+1} and p_{n+1} using the latest available values of u and p
    - Construct the momentum equations
    - Under-relax the momentum matrix
    - Solve the momentum equations to obtain a prediction for u_{n+1}
    - Construct the pressure equation
    - Solve the pressure equation for p_{n+1}
    - Correct the flux for ϕ_{n+1}
    - Under-relax p_{n+1}
    - Correct the velocity for u_{n+1}
    - If not converged, go back to step 2

RESULTS OF THE SIMULATION:

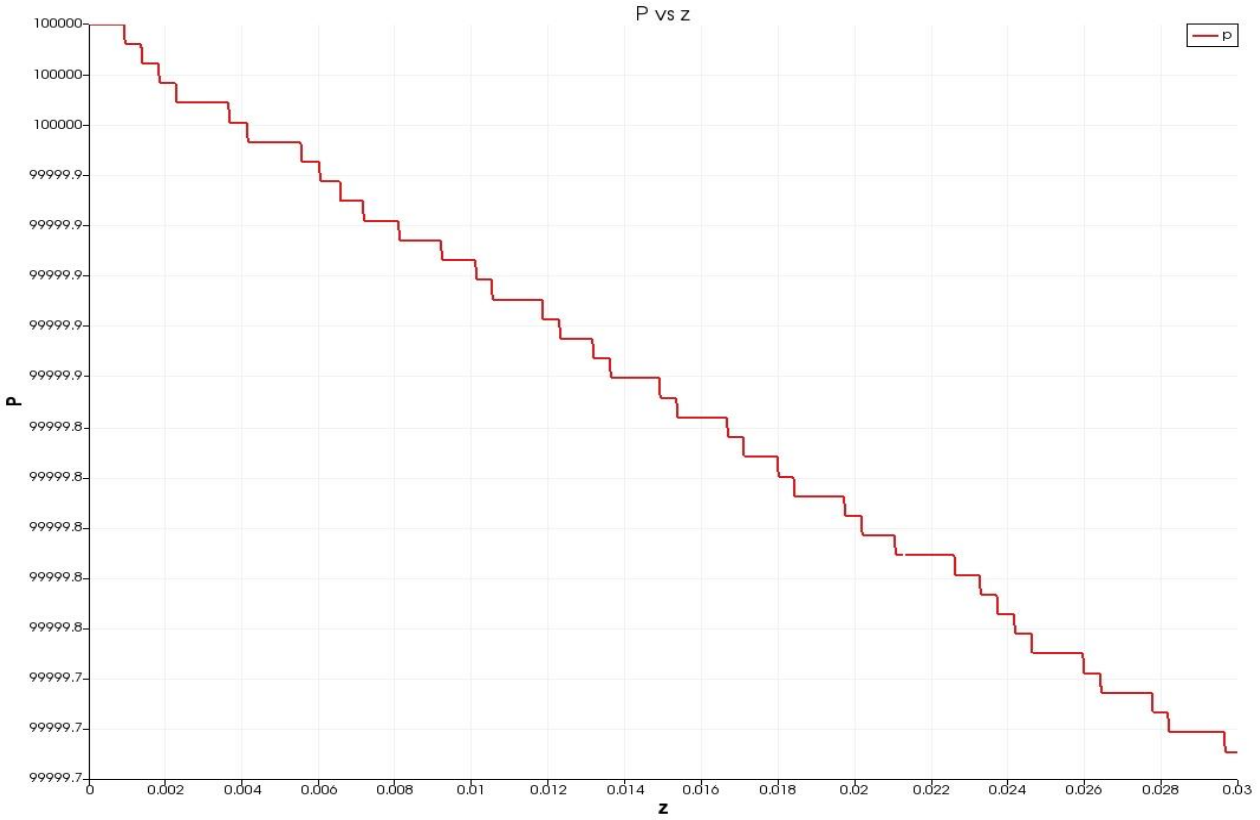
- **Temperature Plot**



- **U_z vs y**

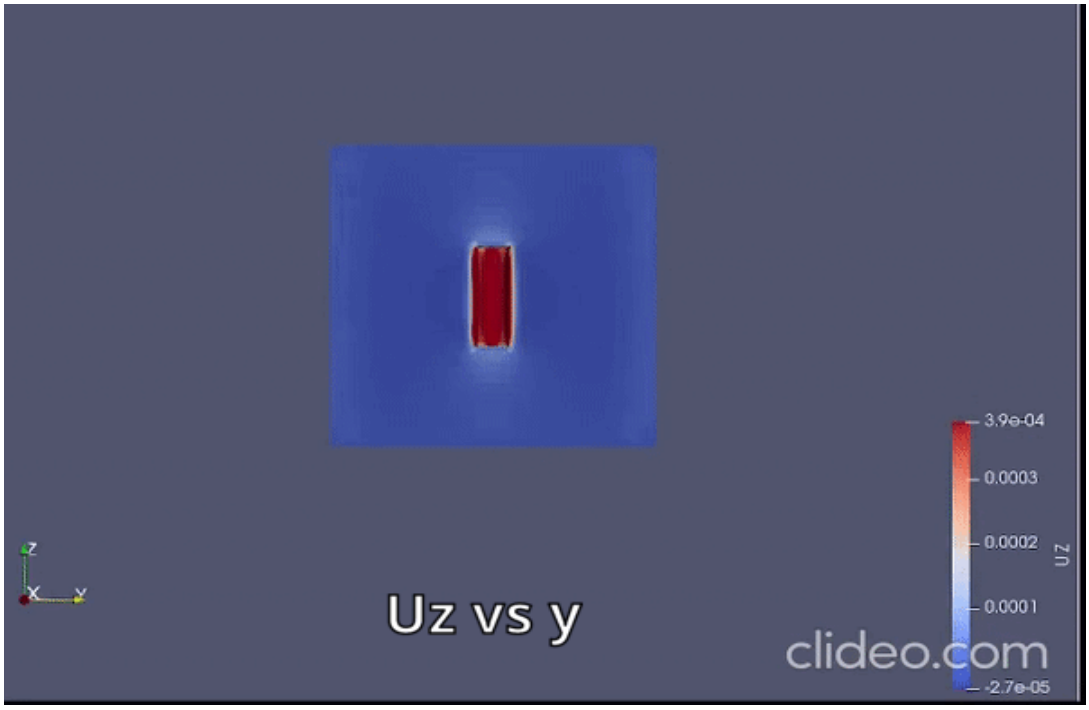

Uz vs y
U_Z
Uz
y
0.036
0.034
0.032
0.03
0.028
0.026
0.024
0.022
0.02
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
-0.002
-0.004
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
0.011
0.012
0.013
0.014

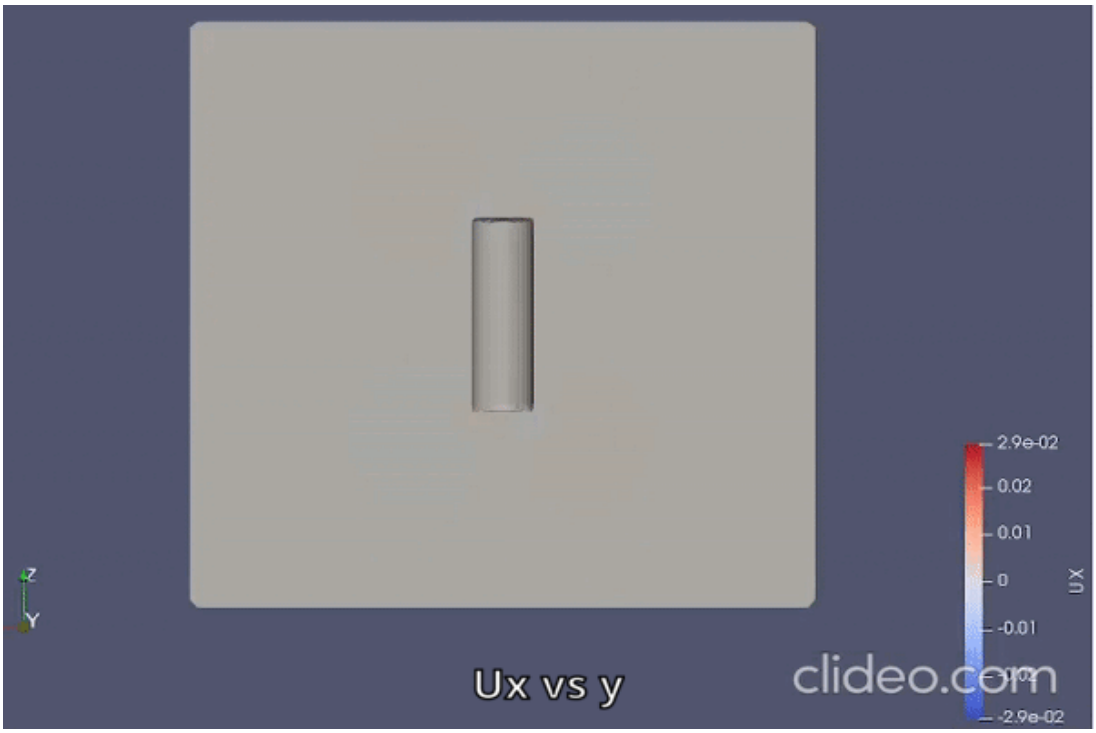
- **P vs z**

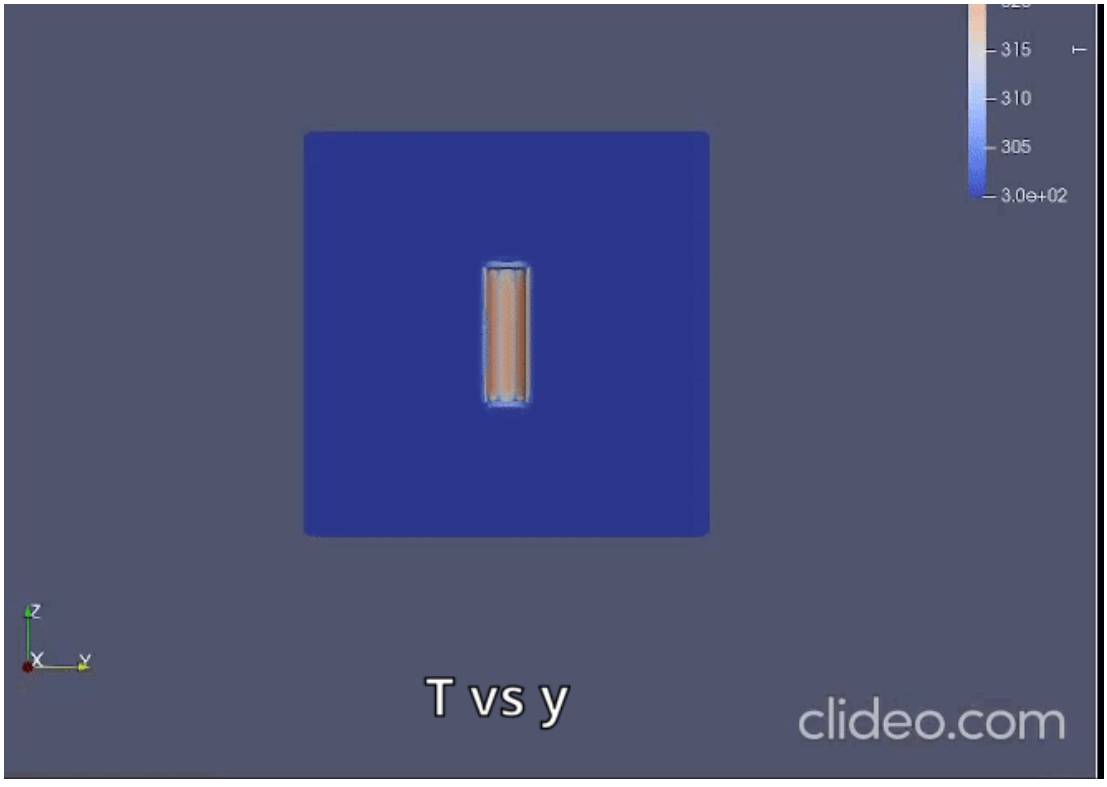

P vs z
p
P
z
100000
100000
100000
99999.9
99999.9
99999.9
99999.9
99999.9
99999.8
99999.8
99999.8
99999.8
99999.8
99999.7
99999.7
99999.7
0
0.002
0.004
0.006
0.008
0.01
0.012
0.014
0.016
0.018
0.02
0.022
0.024
0.026
0.028
0.03

Pressure decreases as z increases due to the presence of gravity. We see steps in the graph as the mesh has discrete points.

Heat Maps of Velocity and Temperature:




2.9e-02
0.02
0.01
0
-0.01
-2.9e-02
UX
Z
Y
Ux vs y
clideo.com


315
310
305
3.0e+02
T
Z
X
Y
T vs y
clideo.com