ME702 CFD project
2D Shock Tube (Sod problem) in OpenFOAM

Luisa Capannolo
Astronomy Department, Boston University, Boston, MA 02215
luisacap@bu.edu

ABSTRACT

In this project, I studied the 2D shock tube problem. The same problem has been studied in depth by Gary Sod in the 70s, providing the community with the analytical solutions for pressure, density and speed in the 1D case. The goal of the project is to extract the problem to two dimensions and compare the outputs to the 1D solutions. This report describes the shock tube problem, the set up implemented in OpenFOAM and the results achieved. The simulations were performed on the SuperComputer Cluster (SCC) at Boston University.

I analyzed the solution as a function of time to see how it behaves as time progresses, checking that a shock wave moves towards the right at a faster rate than the rarefaction wave moving towards the left. I also run multiple cases for different grid resolutions and analyzed how the number of cells considered affects the solution.

I obtained a qualitative agreement with Sod's analytical solution and showed that we need a resolution of at least 300x300 to have a reasonable result.

I also proved that the only variability in the solution is in the direction where we impose the initial conditions for the internal flow. The extra dimension added for the 2D case does not really have an effect to the overall solution of the problem, making the Sod's solution applicable from the 1D to the 2D or even 3D case.

1. INTRODUCTION

The shock tube problem or Sod problem (from Gary A. Sod who studied it in depth in 1978) is frequently used to test the accuracy of computational methods. An analytical solution is in fact available, therefore it is possible to compare numerical results with it and understand the strengths and weaknesses of the scheme implemented.

A shock tube consists of a pipe with circular or rectangular cross section filled with a fluid (or a gas) with a diaphragm splitting the tube in two halves (Figure 1). The diaphragm is numerically simulated as a discontinuity in different fluid conditions (temperature, pressure and density) across that specific surface. Generally, the left side of the tube has higher values for the fluid properties. Given such initial conditions, the system is allowed to evolve in time. Two waves are generated and can be seen in all the physical variables. A shock wave will move towards the right (low pressure region) and a refractive wave will move to the left (high pressure region). A contact discontinuity (moving towards the right side) separates the two regions and is visible in density and temperature (or energy) only.

The analytical solutions can be calculated by solving the three governing fluid equations men-

tioned in the following section and with given initial Dirichlet conditions that mimic the discontinuity positioned in the middle of the shock tube (pressure of 1.0 Pa and 0.1 Pa respectively in the left half and the right half of the x axis, density of 1.000 kg/m^3 and 0.125 kg/m^3 respectively in the left half and the right half of the x axis and null speed everywhere). The results for the 1D case, as a function of x, are displayed in Figure 4. The solution is provided by a MATLAB code that can be found here: `http://www.mathworks.com/matlabcentral/fileexchange/46311-sod-shock-tube-problem-solver`.

2. NUMERICAL SOLUTION

In this project, I am solving the shock tube problem in 2D with the open source package OpenFOAM. The following sections describe the governing equations solved under specific assumptions and the solver used in the project.

2.1. Governing equations

The standard set of 3D fluid equations is:

$$\frac{\partial \rho}{\partial t} + \bigtriangledown \cdot (\rho \overrightarrow{u}) = 0 \tag{1}$$

$$\frac{\partial (\rho \overrightarrow{u})}{\partial t} + \bigtriangledown \cdot [\overrightarrow{u}(\rho \overrightarrow{u})] + \bigtriangledown p + \bigtriangledown \cdot \overrightarrow{T} = 0 \tag{2}$$

$$\frac{\partial (\rho E)}{\partial t} + \bigtriangledown \cdot [E(\rho \overrightarrow{u})] + \bigtriangledown \cdot (\rho \overrightarrow{u}) + \bigtriangledown \cdot (\overrightarrow{T} \cdot \overrightarrow{u}) + \bigtriangledown \cdot \overrightarrow{j} = 0 \tag{3}$$

where ρ is the mass density, $\overrightarrow{u}$ the flow speed, E the total energy density (E = e + $u^2/2$, e is the internal energy density), $\overrightarrow{T}$ is the stress tensor and $\overrightarrow{j}$ the heat flux. Eq. (1) is the conservation of mass, Eq. (2) is the conservation of momentum and Eq. (3) is the conservation of energy.

The shock tube problem analyzed in this projects considers inviscid flows (viscosity is null) and adiabatic processes. The governing equations in this case reduce to the classic fluid equations, where there is no stress tensor and no heat flux.

The equations to be solved are:

$$\frac{\partial \rho}{\partial t} + \bigtriangledown \cdot (\rho \overrightarrow{u}) = 0 \tag{4}$$

$$\frac{\partial (\rho \overrightarrow{u})}{\partial t} + \bigtriangledown \cdot [\overrightarrow{u}(\rho \overrightarrow{u})] = 0 \tag{5}$$

$$\frac{\partial (\rho E)}{\partial t} + \bigtriangledown \cdot [E(\rho \overrightarrow{u})] + \bigtriangledown \cdot (\rho \overrightarrow{u}) = 0 \tag{6}$$

Temperature is directly related to energy, given that e = C_VT = (γ - 1)RT:

$$T = \frac{1}{C_V}\left(\frac{E}{\rho} - \frac{u^2}{2}\right) \tag{7}$$

2.2. Solver: rhoCentralFoam

The solver used to model the 2D shock tube problem is rhoCentralFoam, based on finite volume as all openFOAM solvers and fully described in Greenshields et al. 2010. This solver considers compressible fluids, that is: $\bigtriangledown\cdot\ (\rho\ \overrightarrow{u}) \neq 0$. It is a density-based solver in the sense that ρ is the first variable whose solution if found. From that, T and u are calculated. It is "central" in the sense that the variables (ρ, T and u) are calculated and given as outputs at the centroid of the finite volume.

In this solver, finite volumes are contiguous polyhedral cells with an arbitrary number of faces and cells. A cartoon of the finite volumes is in Figure 5. The finite volume method for `rhoCentralFoam` consists in solving the the differential equations within an integral over a cell volume, assumed fixed in space. The divergence and gradient terms in the governing equations are converted from volume integrals to surface integrals with Gauss's theorem. During the integral calculation, one must know the fluxes at cell faces. These are evaluated by interpolating the cell centre values to the surface points.

3. SHOCK TUBE SET-UP

The case I analyzed in this project is a 2D shock tube. Since OpenFOAM is intrinsically a 3D solver, I had to consider a 3D shock tube for the geometry (Figure 2), but without putting any conditions on the walls in z, I have essentially solved a 2D case.

A tutorial for the shock tube case is already available in the OpenFOAM package (under the compressible cases listed in the `rhoCentralFoam` folder), but it is a 1D case and does not have implemented the internal fields for p, T and u. This section describes how I set up the shock tube for the project. All the source files needed are listed at the end of the report.

3.1. Geometry and mesh

The geometry of the problem is described in the `blockMeshDict` file.

Figure 2 shows the size of the shock tube used: the cross section is a rectangle of 0.2x2 m and the pipe extends for 10 m in the x direction. The x and y directions are the only ones that are refined (later in the report I will show a comparison between different resolution grids, i.e. 20x20, 30x30, 50x50, 100x100, 300x300 and 400x400). Keeping the z axis with only one cell guarantees the problem is geometrically 3D, but has no variation in the third dimension.

Referring to Figure 3, the boundary walls in the x direction are formed by the vertexes 0473 and 1265 and the ones in the y direction are identified by 3762 and 0154. The walls 5674 and 0321 are in the z direction, therefore considered empty (no physical condition is fixed here).

The mesh is computed by OpenFOAM with the command `blockMesh` and the blocks are divided into hexahedral elements.

3.2. Initial and boundary conditions

The initial and boundary conditions are described in the files `p`, `T` and `u` in the folder `0` of the solver. I assume a uniform internal field of null velocity everywhere and a nonuniform internal field for pressure and temperature. Like represented in Figure 1, the red region has pressure of 100,000 Pa and temperature of 348.432 K, while the blue region has pressure of 10,000 Pa and temperature of 278.746 K. The temperature values are obtained from the ideal gas law, considering Sod's initial conditions in section 1. The solver `rhoCentralFoam`, in fact, requires initial and boundary conditions for the `T` field rather than the density one.

All the boundary walls are set to have zero gradient and the walls in the third dimension are kept as "empty".

3.3. Solver controls

All the controls relative to the Δt of the scheme, the output formats and length of the simulation are set in the file `controlDict` in the folder `system`. The simulation then it is run calling

the command `rhoCentralFoam`.

I set the start time of the simulation at 0 and let the simulation run for 10 ms for all the grid resolutions except for 400x400 (which requested to be submitted as a batch job). I sampled the solution every 0.5 ms, but displayed the results every 2 ms only.

Δt is set at 10^{-6} s for all cases, allowing the study on the spatial resolution only.

3.4. Post-processing

To study the results the software ParaView is the most straightforward, however for an incompatibility problem, I could not make use of it. I instead extracted data from the outputs of the simulation with the `sampleDict` script in the `system` folder. I run the script with the command `sample` and find the data files in the `postProcessing/set` folder. Here, there is a folder for each time step sampled.

I extracted line cuts along the x axis for different grid resolutions and time steps, and slice cuts of the plane xy for the 300x300 case at the time step of 4 ms. With the `sampleDict` script it is possible to select a specific direction, beginning and end points of the line cut, the number of sampled points and the normal vector in case of the plane extraction.

I then plotted my results through Python 2.7 scripts I coded.

4. RESULTS

In this section, I present the results from the simulations, showing the evolution of the solution at different time steps, the comparison between different resolutions at the same time step and the pressure, density, temperature and speed 2D fields at a specific time step and grid resolution. I also compare the OpenFOAM results to the MacCormack 2-step scheme implemented during the CFD course.

4.1. Comparison at different time steps: line-cuts

Data have been extracted along the x axis, from -5 to 5 m (y and z are both at 0) with the `sampleDict` script. The sampling is of 500 points in all the plots.

The purpose of the Figures 6, 7, 8, 9, 10, 11 is to show how the solution evolves as time progresses. The rarefaction wave is displayed by the decreasing ρ, p and T right at x = 0. This wave progressively moves towards the left. The shock wave is instead the first front on the right for all the physical variables. It is the fastest wave and moves towards the right side of the x axis. The speed plots show the speeds of both waves: the shock wave shows a discontinuity in speed that moves towards the right, while the rarefaction wave has a smoother trend and shows a progressively decreasing speeds on the left.

The temperature and density profiles also show the contact discontinuity. This additional feature is the second step in the density trend and the sharp increase in temperature.

The time step at 10 ms shows the wave hitting the boundary at 5 m. Since we are not considering any reflection at the edges of the shock tube, this wave is not reflected and progressively exits the considered x range.

As time progresses, the shock wave and the rarefaction wave become more and more distant between each other and eventually the solution will smooth out, if the time is allowed to run long enough (not shown in this project).

4.2. Comparison at different grid resolutions: line-cuts

I have run the same simulation for different grid values. Since the project is focused on the 2D shock tube problem, I have refined only the x and y directions. The number of cells in each direction is defined in the file `blockMeshDict`. I have selected resolutions of 20x20, 30x30, 50x50, 100x100, 300x300 and 400x400. The third direction (z) is always refined at 1 cell only.

In Figure 12, I compare the solution at different grid resolutions extracted along the x axis between -5 and 5 m, in the middle of the shock tube (0 m for y and z) for the time step 4 ms. Keeping in mind how the analytical solution looks like (Figure 4), a higher resolution in the grid is preferable. The low resolution cases do not even show the shock wave step function on the right side at all, completely failing at reproducing the analytical solutions. However, to obtain an acceptable solution, one does not need to have an extremely high resolution. The difference between the 300x300 and the 400x400 case is almost non existent, indeed, showing that a solution

is already acceptable by refining the axes x and y only with 300 cells.

4.3. Visualizing the shock wave: plane-cuts

I extracted data in the plane xy with the `sampleDict` script, switching from the `line` command to the `surface` one. I also have turned on the `interpolate` feature which allows me to have 300x300 points in the data file. For an incompatibility issue, the plots are manually elaborated with Python 2.7 rather than ParaView.

The Figure 13 shows the 2D cuts for the 300x300 resolution case at the time step 4 ms. Since there is no significant difference between the 300x300 resolution case and the 400x400 one, I am showing here the results for the 300x300 grid only.

I display in colors the physical property values at each x, y coordinate. As expected, there is no variability along the y axis since the initial conditions show a discontinuity along the x axis only.

With these color plots, it is possible to appreciate the shock and rarefaction waves even better. The speed plot, in fact, shows on the right a sharp edge in color around x = 2.2 m and a smoother transition between red and blue on the left between 0 and -1.8 m. Similarly, pressure has a gradient in colors on the left and a sharp edge on the right. The density color plot shows a smooth color transition on the left and two sharp edges on the right at 1.1 and 2.2 m, matching exactly what is seen in the line cuts and reproducing respectively the rarefaction wave, the contact discontinuity and the shock wave. Similarly to the density, the temperature plot shows the two sharp edges on the right (positioned where the density discontinuities are) and a smoother decrease on the left.

4.4. Comparison with MacCormack 2-step scheme

During the CFD course, we implemented the MacCormack 2-step scheme to solve the Sod's problem. The script is written in Python 2.7 and solves the Euler's 1D equations in the conservative form. Since this scheme does not provide an acceptable solution by itself, it is needed to add an artificial viscosity term on the second step. The value of the viscosity (ϵ) is arbitrary and picked by the user in order to get a solution similar to the analytical one as much as possible.

Using the same values for the spacial and temporal resolution of the OpenFOAM case (for the 300x300x1 grid refinement), I compared the Sod's solution calculated with this scheme to the 2D OpenFOAM one in Figure 14. The value for ϵ I used is 0.006 and I run the solution for 4 ms, like the OpenFOAM one.

The MacCormack scheme reproduces overall the trend of the Sod's solutions, however it introduces much more diffusion than the OpenFOAM case. The shock wave and the contact discontinuity are much sharper and step-like in the OpenFOAM case rather than the MacCormack solution.

5. DISCUSSION and SUMMARY

In this project, I have studied the Sod problem with "soft" boundary conditions (pressure ratio is rather small between the two regions) with the OpenFOAM solver `rhoCentralFoam`. The shock tube problem has been analyzed by Sod in 1D only and in the project I show that its solutions can be extended to the 2D case as well, if we maintain the initial solutions like those in Figure 1. The project goal is to obtain solutions similar to the analytical ones as much as possible, looking at the time variability of the solutions and at how significant is a variation of the spatial resolution.

I first focused my attention on looking at how the solution evolves with time. A shock wave in p is formed right away and it moves progressively to the region of lower pressure. On the other side, a rarefaction wave moves towards the left at a lower speed. The features can be found in the speed, temperature and density solutions as well: the shock wave is the first discontinuity on the right and the rarefaction wave is the progressively decreasing feature on the left. The density and temperature profiles show the contact discontinuity as well with a second step in the density trend and a sharp increase in the temperature plot. Overall, these trends are similar for all the cases I analyzed, but a high resolution is needed to appreciate the crisp shocks.

Since resolution is fundamental to obtain a good solution, I focused my attention on how the spatial resolution affects the solution. I maintained the time resolution fixed and varied only the refinement along the x and y axis. The comparison between all the different cases I have run shows that the solution becomes more and more similar to the analytical one at resolution of 300x300 and over. It is interesting to notice that the cases of resolution 300x300 and 400x400 are essentially the same, therefore it is acceptable to stop at a resolution of 300x300 to analyze the simulation outputs. This allows the solver to run faster without losing any information on the solution, since this is already well defined for a grid of 300x300. The lower the resolution, the worst the solutions are. For example, the 20x20 case basically smooths out all the interesting features and essentially does not shock a shock wave in pressure at all.

Improving the resolution determines a higher CFL number, that should be close to 0.5 for the central scheme implemented in `rhoCentralFoam`. For this specific cases, the CFL number goes from 0.0005 for the 20x20 case up to 0.1 in the 400x400 case. This shows that one does not need a very high CLF of 0.5 to obtain a reasonable solution. Obviously, a higher resolution than 400x400 would surely approach a CLF closer to 0.5, but as discussed, refining the x and y axis in 300 cells already provides an acceptable solution that resembles in all its features the analytical one.

Comparing the general trends between the 300x300 case and the analytical solutions, in fact, we can see that they are overall very similar. The biggest differences can be found whenever there is a discontinuity: for the OpenFOAM solutions, in fact, there will always be some diffusion that does not allow to have the discontinuities corners exactly squared. Another interesting difference is in the small overshoot of temperature past the shock wave, which is not found in the analytical case. This, again, is an effect of a numerical solution that cannot precisely reproduce discontinuities well.

The last portion of the project shows 2D plots of the physical quantities over the xy plane of the shock tube block. These color figures are useful to visualize better the shock wave described by

a sharp change of colors on the left region and the rarefaction wave moving towards the left and represented by a smoother transition of colors in all the physical quantities.

The variability of p, T, u and ρ, however, is only along the x axis, the only direction where the initial conditions have a discontinuity and where waves develop as the system evolves in time. Extracting data along any point of the y axis or z axis (not shown in the report), indeed, shows constant values of any of the physical outputs. Essentially, these plots are simply a 2D representation of the line cuts along the x direction: the position of the shock waves and rarefaction waves exactly match the features in the x-axis cuts, proving that the 1D solutions are the same in the 2D case too.

In addition, it is worth noting that refining in the y direction is probably only a waste of memory since having a higher number in this direction is not needed: the initial conditions here are simply constant in space.

Finally, by comparing the OpenFOAM output to the MacCormack 2-step scheme solution, we can see that both solvers reproduce the overall trend of the analytical solution. `rhoCentralFoam`, however, reproduces the shock wave and contact discontinuity step-like features much better than the MacCormack scheme. The latter solver introduces a lot of diffusion, indeed, making all the discontinuities much smoother than how they should be. More importantly, the MacCormack scheme has an artificial viscosity term in it, whose amplitude is set by the user so the solutions are intrinsically affected by the value selected.

REFERENCES

Greenshields, Christopher J. and Weller, Henry G. and Gasparini, Luca and Reese, Jason M., 2010, International Journal for Numerical Methods in Fluids, volume 63, DOI 10.1002/fld.2069

For the MATLAB code (analytical solution plots): `http://www.mathworks.com/matlabcentral/fileexchange/46311-sod-shock-tube-problem-solver`, which gets the solution from here: `http://www.phys.lsu.edu/%7Etohline/PHYS7412/sod.html`

This preprint was prepared with the AAS LaTeX macros v5.2.

FIGURES

The first three figures have been made with PowerPoint. The fourth figure is from an existing MATLAB code mentioned in the report section 1. All the figures with plots have been elaborated with Python 2.7.

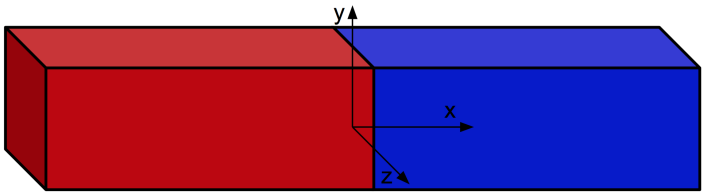

Figure 1: Cartoon of a 3D shock tube with rectangular cross section. The pipe is colored in two different regions to show the difference between the initial conditions: the red half has higher pressure, density and temperature, the blue region has lower pressure, density and temperature. The triad of vectors shows the orientation of the axes.

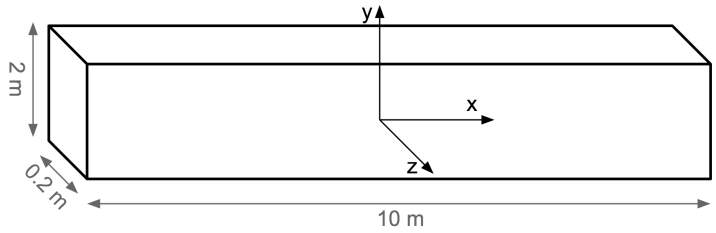

Figure 2: Geometry of the shock tube used in the project. The pipe is 10 m long, 2 m high and 0.2 m thick. The width of the tube (in the z direction) is smaller compared to the other two to simulate a 2D case rather than a 3D tube. The triad of vectors shows the orientation of the axes.

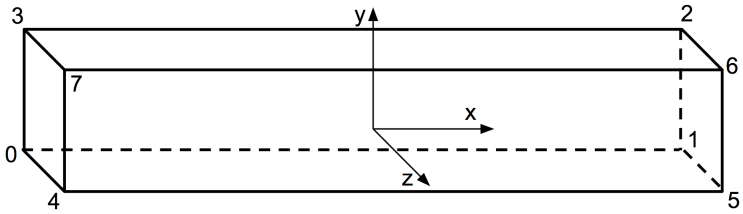

Figure 3: Vertex of the shock tube that identify the boundary walls. The boundary active walls are defined by the vertex 0473, 1562, 0154 and 3763. The walls 4765 and 0123 are empty walls since the case analyzed is a 2D case. Only the x and y directions are refined by a specific number of cells (20x20, 30x30, 50x50, 100x100, 300x300 and 400x400). The triad of vectors shows the orientation of the axes.

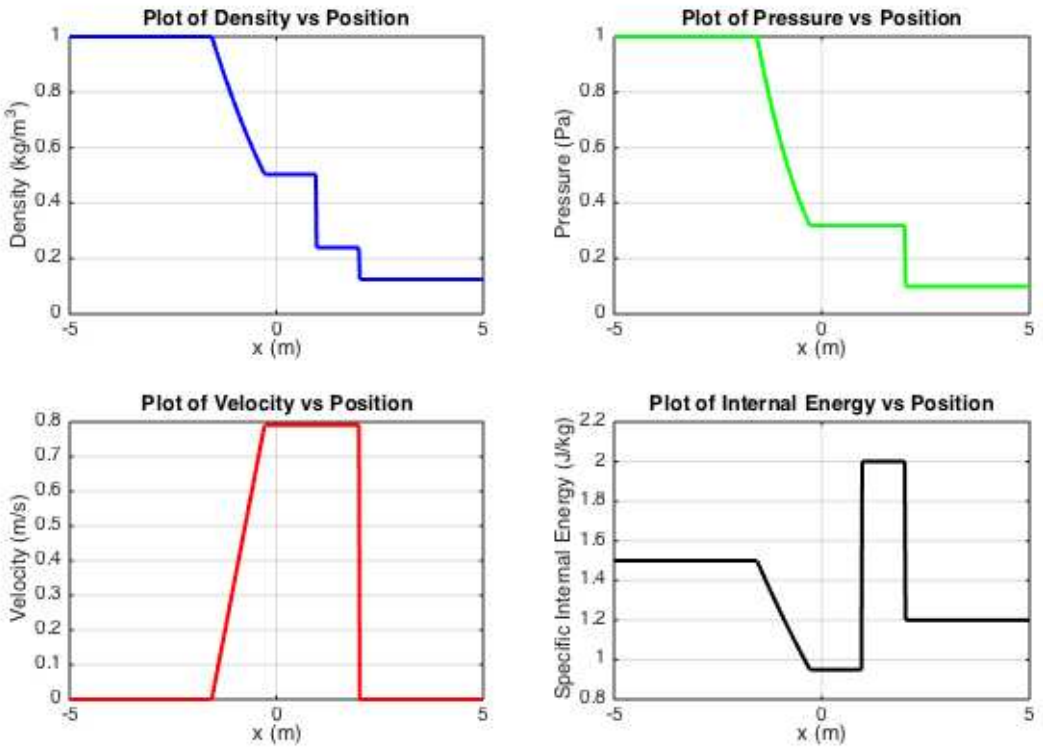

Figure 4: Analytical 1D solution of the shock tube problem from the MATLAB code here: `http://www.mathworks.com/matlabcentral/fileexchange/46311-sod-shock-tube-problem-solver`. The initial conditions have null speed along the x axis, pressure of 1 Pa in the first half and 0.1 Pa in the second half and density of 1.000 kg/m^3 in the first half and 0.125 kg/m^3 in the second half.

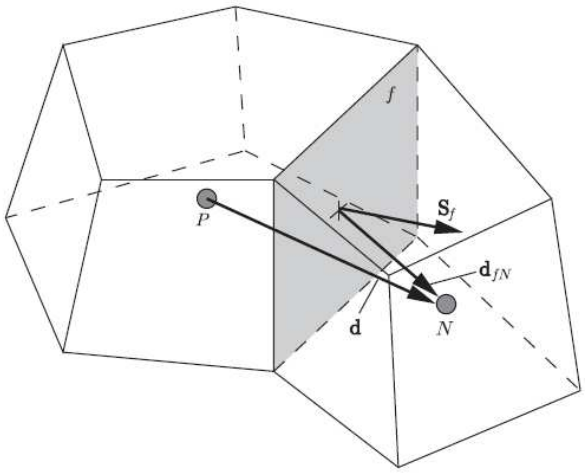

Figure 5: Cartoon of the contiguous polyhedral finite volumes in the solver `rhoCentralFoam`. P and N are indicative of the centroid of each cell. f is the neighbouring face and the vector $\overrightarrow{d_{fN}}$, $\overrightarrow{d}$ and $\overrightarrow{S_f}$ indicate respectively distances and the surface f normal unit vector. Adapted figure from Greenshields et al. 2010.

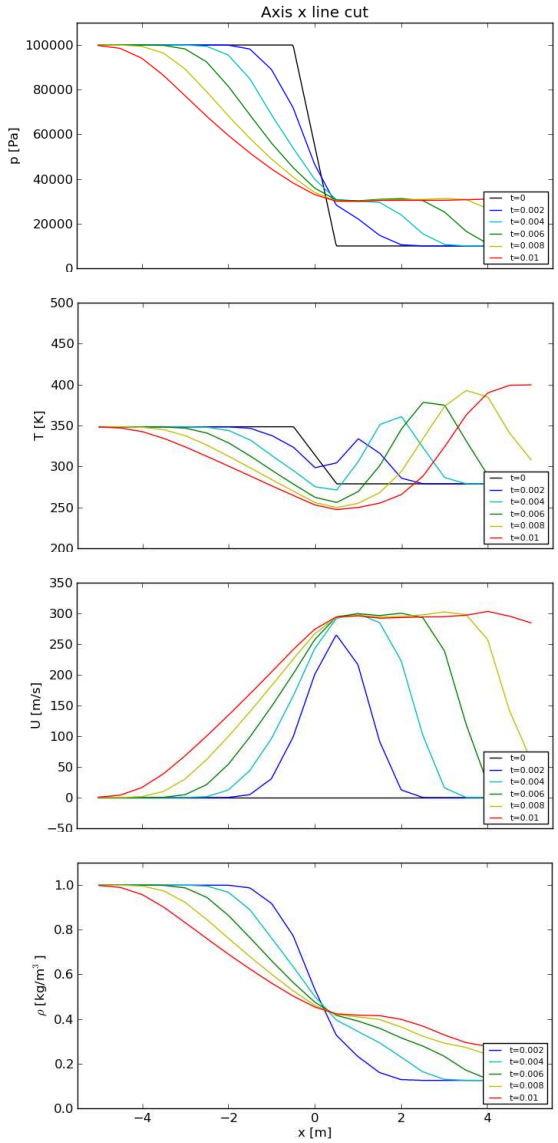

Figure 6: Line cut along the x axis between -5 and 5 m for the 20x20 case, sampled at 500 points. The different color lines represent the solutions at different times as indicated by the legend. From the top to the bottom plot, I display pressure, temperature, speed along x and density as a function of x.

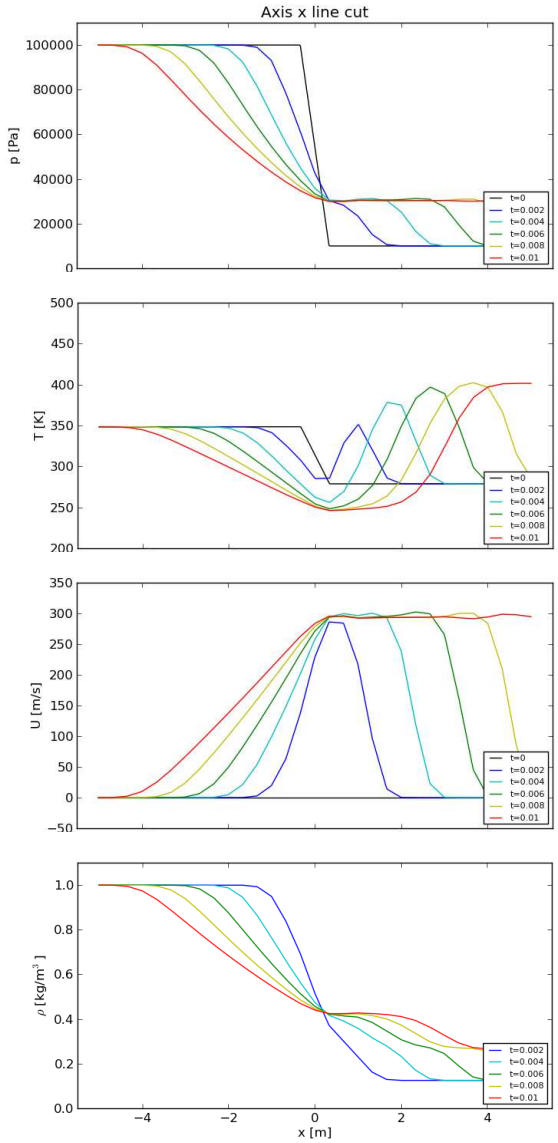

Figure 7: Line cut along the x axis between -5 and 5 m for the 30x30 grid resolution case, sampled at 500 points. The different color lines represent the solutions at different times as indicated by the legend. From the top to the bottom plot, I display pressure, temperature, speed along x and density as a function of x.

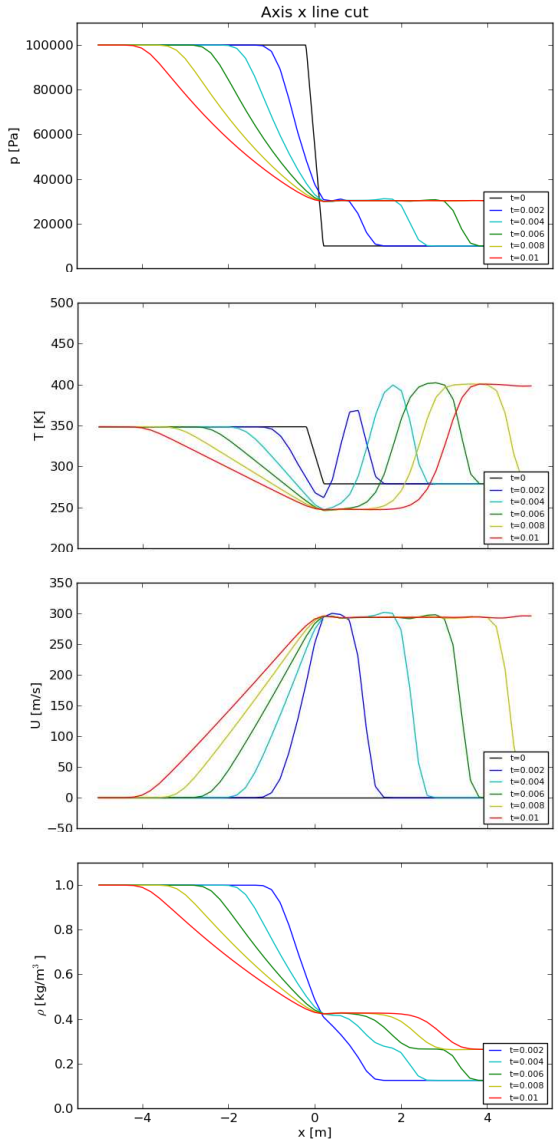

Figure 8: Line cut along the x axis between -5 and 5 m for the 50x50 grid resolution case, sampled at 500 points. The different color lines represent the solutions at different times as indicated by the legend. From the top to the bottom plot, I display pressure, temperature, speed along x and density as a function of x.

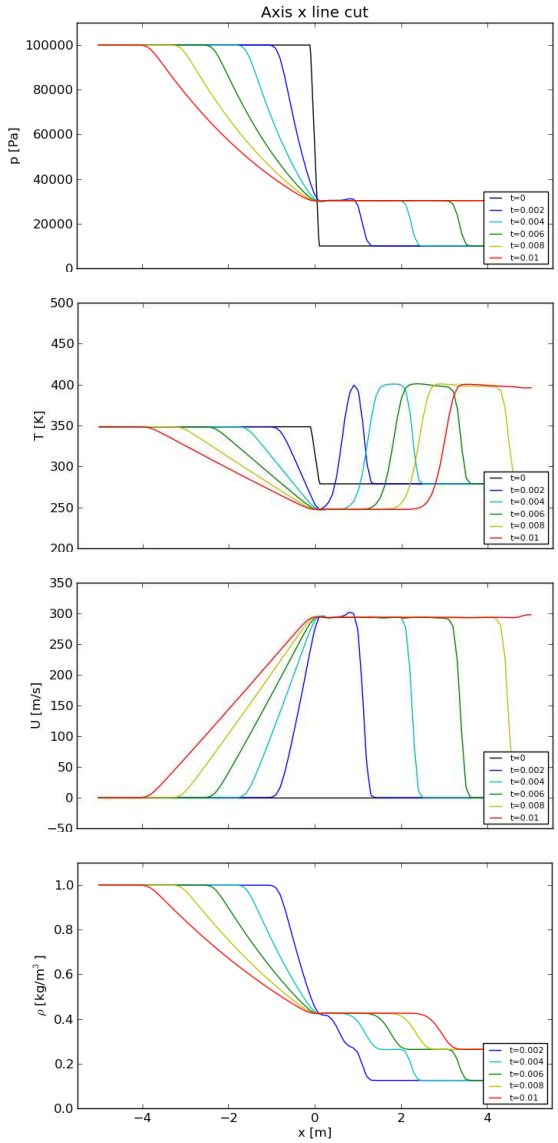

Figure 9: Line cut along the x axis between -5 and 5 m for the 100x100 grid resolution case, sampled at 500 points. The different color lines represent the solutions at different times as indicated by the legend. From the top to the bottom plot, I display pressure, temperature, speed along x and density as a function of x.

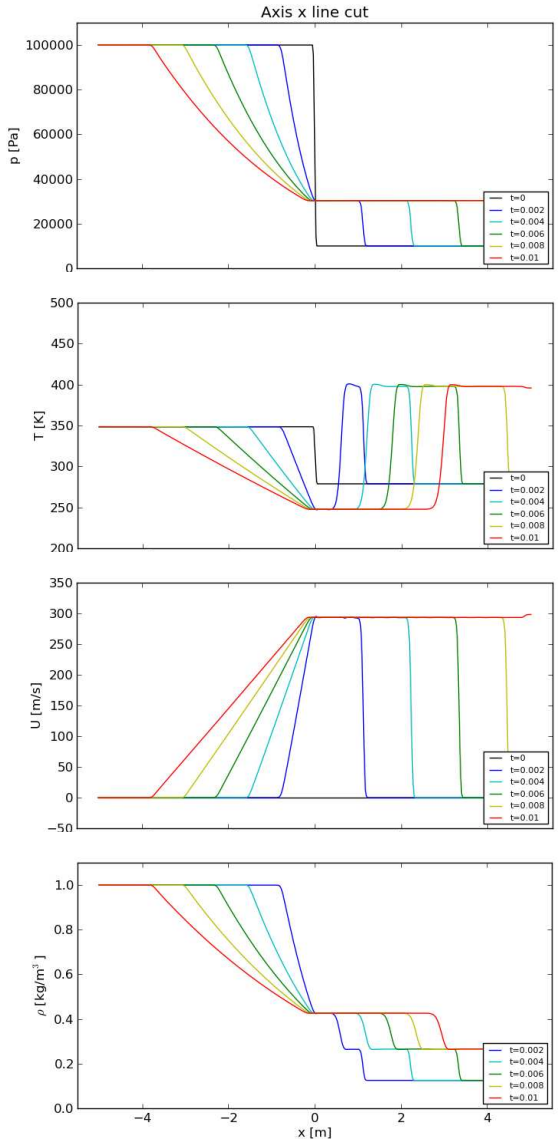

Figure 10: Line cut along the x axis between -5 and 5 m for the 300x300 grid resolution case, sampled at 500 points. The different color lines represent the solutions at different times as indicated by the legend. From the top to the bottom plot, I display pressure, temperature, speed along x and density as a function of x.

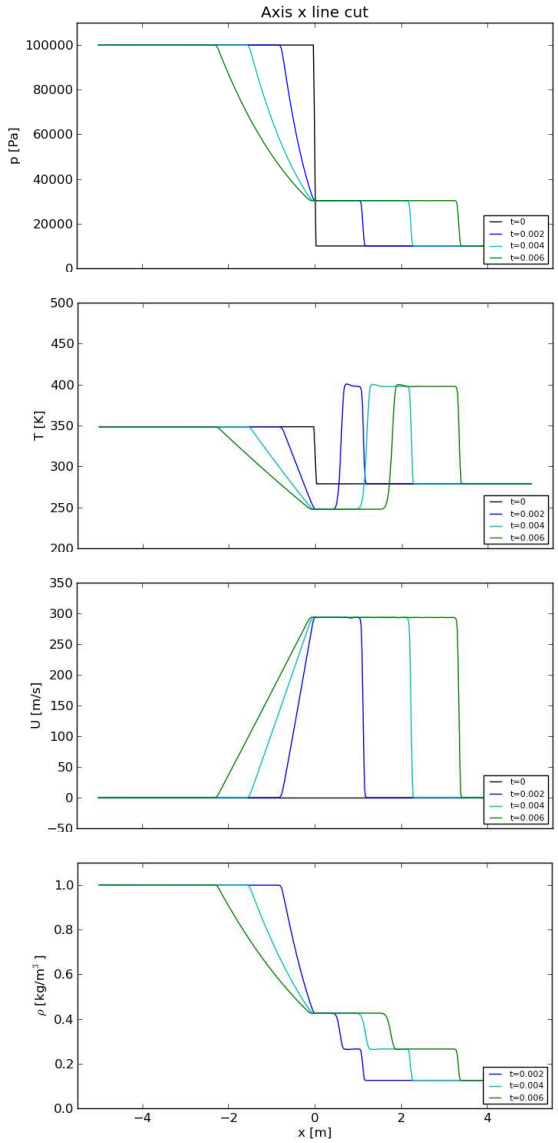

Figure 11: Line cut along the x axis between -5 and 5 m for the 400x400 grid resolution case, sampled at 500 points. The different color lines represent the solutions at different times as indicated by the legend. From the top to the bottom plot, I display pressure, temperature, speed along x and density as a function of x.

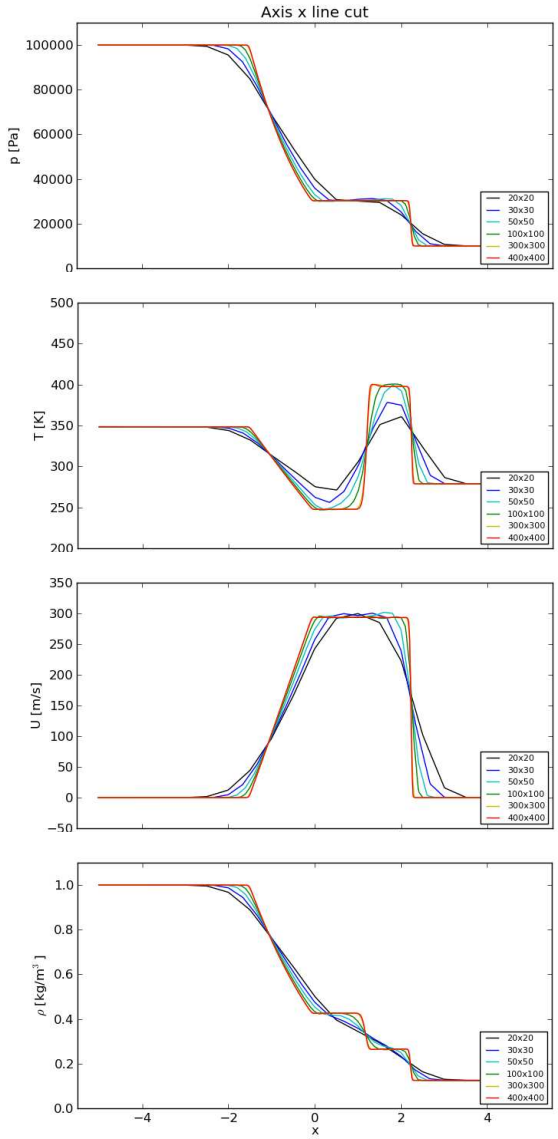

Figure 12: Line cut along the x axis between -5 and 5 m at 4 ms, sampled at 500 points. The different color lines represent the solutions at resolutions as indicated by the legend. From the top to the bottom plot, I display pressure, temperature, speed along x and density as a function of x.

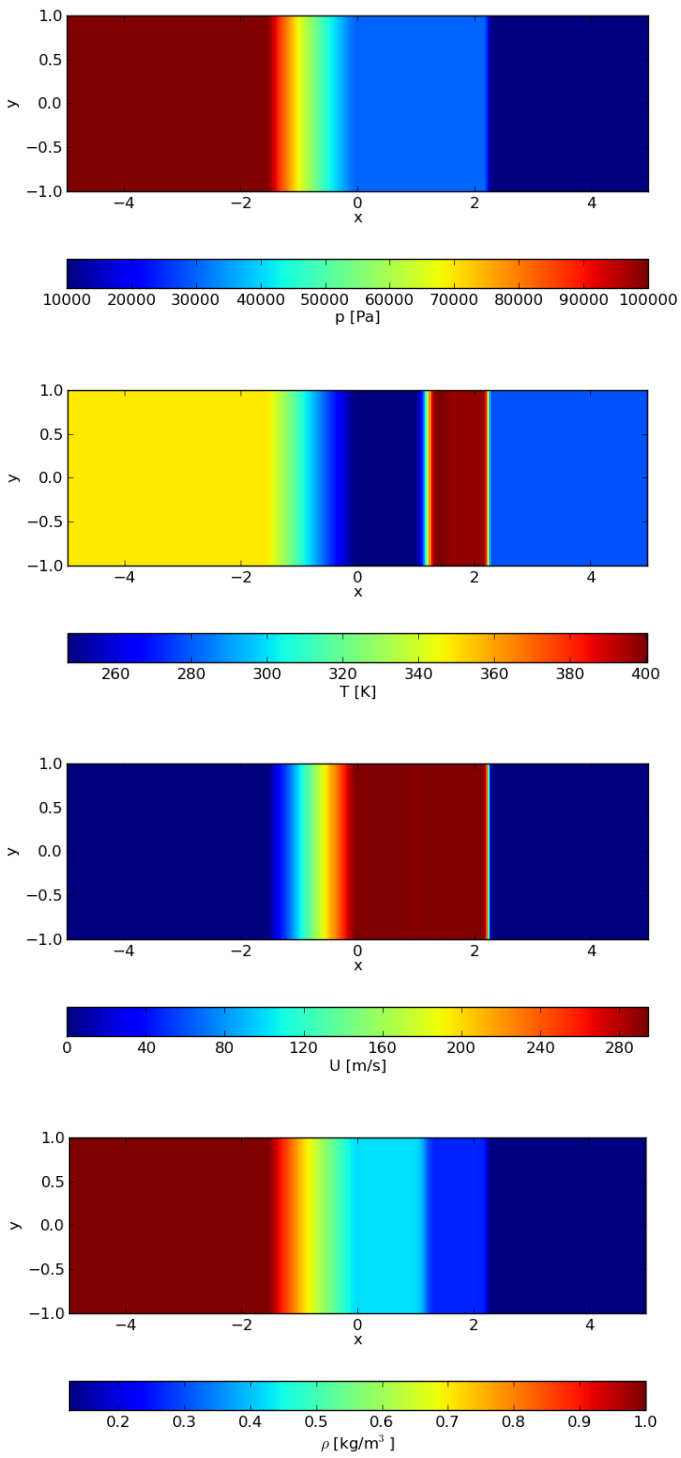

Figure 13: Slice cut of the xy plane at 4 ms between -5 and 5 m in the x direction and at y=0 m. The interpolation is turned on so the number of points is 300x300, like the resolution. From the top to the bottom plot, I display pressure, temperature, speed along x and density as a function of the coordinates x and y. The colors are indicative of the actual value of the physical property plotted.

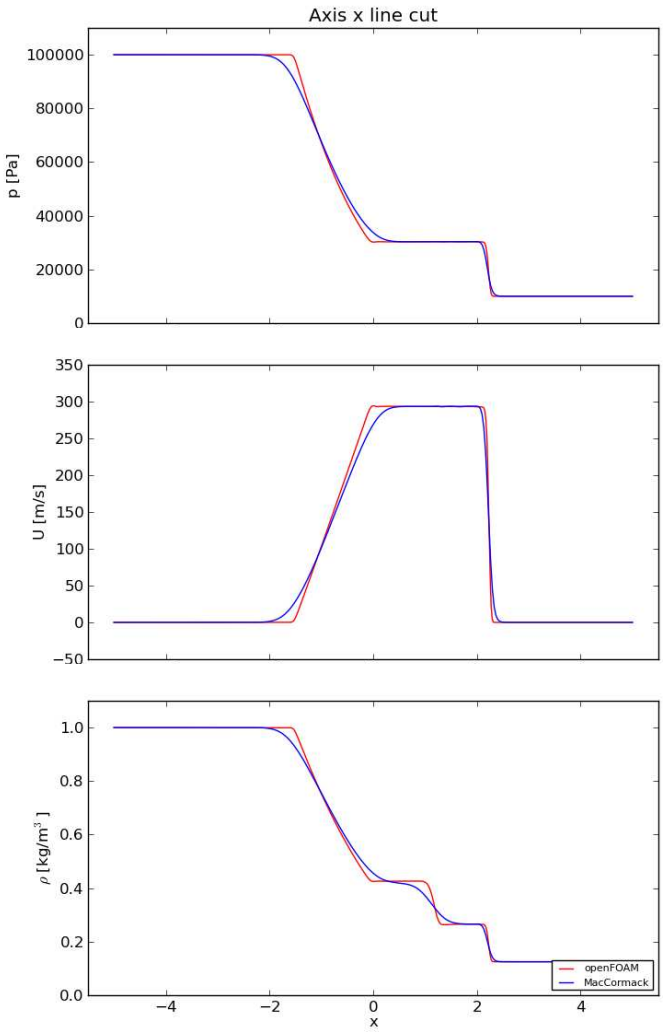

Figure 14: Comparison between the OpenFOAM simulation in the 300x300x1 grid resolution and the 2-step scheme by MacCormack, coded in Python 2.7 during the CFD course. The simulations have the same spacial and temporal resolutions, and are both run for 4 ms. The MacCormack scheme has an artificial viscosity term added to avoid oscillations at the discontinuities boundaries.

SOURCE CODES

The following source codes are relative to the resolution grid case of 300x300. They can be used for any other resolution, granted that the file `blockMeshDict` and the `p`, `U` and `T` fields are updated.

a) constant/polyMesh/blockMeshDict

This file defines the geometry and the grid of the problem, by listing the sides (or vertex) of the box used. In the section `blocks`, the number of cells are set. Thsi specific case is relative to the 300x300 resolution grid, therefore both the x and the y axis are refined into 300 points. It also sets which ones are the active boundary walls ("sides") and which are empty ones ("empty") by grouping them into labels.

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                       |
|
| \\      /  F ield               | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration           | Version:  2.4.0
|
|   \\  /    A nd                 | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation        |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
    (-5 -1 -0.1) //vertex 0
    (5 -1 -0.1) //vertex 1
    (5 1 -0.1) //vertex 2
```

```
    (-5 1 -0.1) //vertex 3
    (-5 -1 0.1) //vertex 4
    (5 -1 0.1) //vertex 5
    (5 1 0.1)  //vertex 6
    (-5 1 0.1) //vertex 7
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (300 300 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    sides
    {
        type patch;
        faces
        (
            (1 2 6 5)
            (0 4 7 3)
            (3 7 6 2)
            (0 1 5 4)
        );
    }
    empty
    {
        type empty;
        faces
        (
            (5 6 7 4)
            (0 3 2 1)
        );
    }
);
```

```
mergePatchPairs
(
);

// ************************************************************************* //
```

b) 0/p

This file describes the field relative to the pressure. The internal field for the shock tube is non uniform: the first half and second half of the x axis cells are set at 100,000 Pa and at 10,000 Pa, respectively. In order to set properly the internal field initial conditions, one has to list these values in the correct order. The value 100,000 needs to be repeated $300/2 = 150$ times, followed by the value 10,000 for 150 times. The first sequence fills out the x axis from -5 m to 5 m, with y fized at -1 m. This sequence has to be repeated 300 times such that the whole pipe is set up properly to cover the 300x300 existing cells.

The boundary conditions on the walls are also set in this file. For the shock tube, there is a zero gradient condition on the active walls, and an empty condition on the walls in the z direction, confirming the non variability of the third dimension.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.4.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
dimensions      [1 -1 -2 0 0 0 0];

internalField   nonuniform List<scalar>
90000
(
100000
100000
"
"
100000
100000
10000
10000
"
"
10000
10000
"
"
)
;

boundaryField
{
    sides
    {
        type            zeroGradient;
    }

    empty
    {
        type            empty;
    }
}

// ************************************************************************* //
```

c) 0/T

This file describes the temperature field. It is structured similarly to the `p` file, but the temperature is split between 348.432 K and 278.746 K. The boundary conditions are the same as the ones set for `p`.

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.4.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField       nonuniform List<scalar>
90000
(
348.432
348.432
"
"
348.432
348.432
278.746
278.746
"
```

```
"
278.746
278.746
)
;


boundaryField
{
    sides
    {
        type            zeroGradient;
    }

    empty
    {
        type            empty;
    }
}

// ************************************************************************* //
```

d) 0/U

This field describes the speed field. Contrary to `p` and `T`, the internal field for U is uniformly set at 0. The boundary conditions are the same as the ones set for `p`.

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.4.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
```

```
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    sides
    {
        type            zeroGradient;
    }

    empty
    {
        type            empty;
    }
}

// ************************************************************************* //
```

e) system/controlDict

This file includes all the solver instructions, such as the Δt used, the time sampling in the data outputs and the time length of a simulation.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                      |
|
| \\      /  F ield              | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration          | Version:  2.4.0
|
|   \\  /    A nd                | Web:      www.OpenFOAM.org
|
```

```
|     \\/     M anipulation   |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "system";
    object       controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     rhoCentralFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.01;

deltaT          1e-06;

writeControl    adjustableRunTime;

writeInterval   0.0005;

cycleWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;
```

```
runTimeModifiable true;

adjustTimeStep  yes;

maxCo           0.2;

maxDeltaT       1;


// ************************************************************************* //
```

f) system/sampleDict - line cuts

This file is used to extract data files from the simulation outputs through a line cut. For this specific source code, data is extracted along the x axis from -5 to 5 m and sampled at 500 points. Since the command used is `axis xyz`, the data file saves also the y and z points (always zero, in this case).

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.4.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      sampleDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
interpolationScheme cellPoint;

setFormat       raw;

sets
(
    line
    {
        type    uniform;
        axis    xyz;
        //axis  distance;
        start   ( -5. 0.0 0.0 );
        end     ( 5. 0.0 0.0 );
        nPoints 500;
    }
);

fields          ( p U T rho );


// ************************************************************************* //
```

h) system/sampleDict - plane cuts

This file is used to extract data from the simulation in a slice or plane. It is similar to the previous one, but different in that a specific plane is selected by the normal vector and the initial point of extraction is identified by the `basePoint` command. The command `interpolate` is set true such that the number of data points is exactly 300x300, like the resolution expected. The data file produced by this file has the first three columns indicating the x, y, z coordinates of the selected plane. The other columns are relative to the physical properties listed in the command `fields`.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                      |
|
| \\      /  F ield              | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration          | Version:  2.4.0
|
|   \\  /    A nd                | Web:      www.OpenFOAM.org
|
```

```
|     \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "system";
    object       sampleDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

interpolationScheme cellPoint;

surfaceFormat    raw;


surfaces
(
    myPlane
    {
        type     plane;
        basePoint (-5 0 0);
            normalVector (0 0 1);
            interpolate          true;
    }
);

fields           ( p U T rho );


// ************************************************************************* //
```

k) constant/thermoPhysicalProperties

This file describes the type of gas used. In this shock tube case, the file has not been edited and it is the same as the file in the already existing tutorial. One thing to be noted: `mu` is set to zero since the shock tube problem involves inviscid flows.

```
/*---------------------------------*- C++ -*----------------------------------*\
```

```
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.4.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        nMoles          1;
        molWeight       28.96;
    }
    thermodynamics
```

```
    {
        Cp              1004.5;
        Hf              2.544e+06;
    }
    transport
    {
        mu              0;
        Pr              1;
    }
}


// ************************************************************************* //
```

j) system/fvschemes

This file has all the information relative to the scheme used. It has not been modified for the shock tube problem.

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.4.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
fluxScheme      Kurganov;

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(tauMC)      Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    reconstruct(rho) vanLeer;
    reconstruct(U)  vanLeerV;
    reconstruct(T)  vanLeer;
}

snGradSchemes
{
    default         corrected;
}


// ************************************************************************* //
```