Compressible flows – Sod's shock tube

- Let us run this case. Go to the directory:

```
$PTOFC/sod_shock_tube
```

- $PTOFC is pointing to the directory where you extracted the training material.
- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are human-readable, and we highly recommend you open them, get familiar with the steps, and type the commands in the terminal. In this way, you will get used with the command line interface and OpenFOAM commands.
- If you are already comfortable with OpenFOAM, run the cases automatically using these scripts.
- In the case directory, you will also find the `README.FIRST` file. In this file, you will find some additional comments.

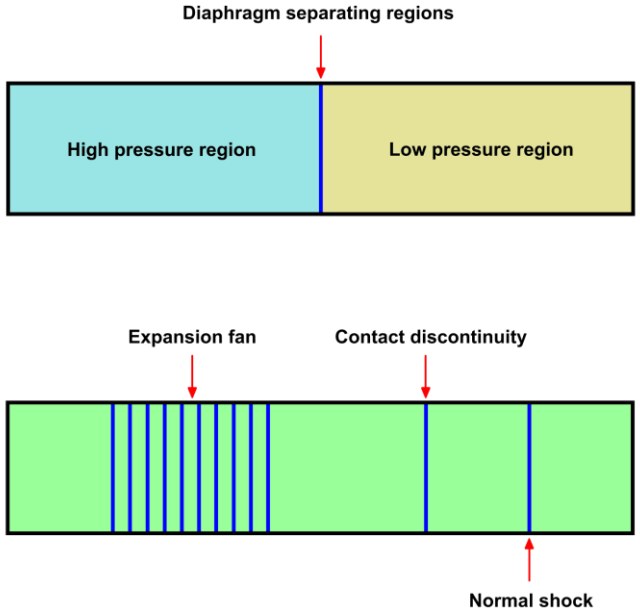
Compressible flows – Sod's shock tube

Sod's shock tube

Physical and numerical side of the problem:

- The governing equation of this test case are the Euler equations.
- This case has an analytical solution and plenty of experimental data.
- This is an extreme test case used to test solvers. Every single commercial and open-source solver use this case for validation of the numerical schemes.

Shock tube. The driver section, including vacuum pumps, controls, and helium driver gas.
Photo credit: Stanford University. http://hanson.stanford.edu/index.php?loc=facilities_nasa



After breaking the diaphragm, a complex system of shock waves is created inside the tube (an expansion wave, a contact discontinuity, and a normal shock wave).

Compressible flows – Sod's shock tube

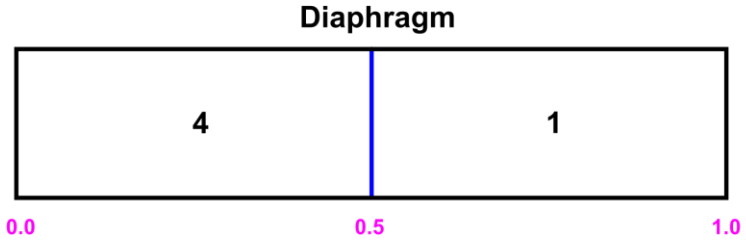
What are we going to do?

- We will use this case to learn how to setup supersonic flow cases.
- We will compare the numerical solution with the analytical solution.
- We will run the case with a robust numerics, but you are invited to try different setups and compare the different outcomes.
- To find the numerical solution we will use the solver `rhoPimpleFoam` and with zero viscosity (Euler equations).
- `rhoPimpleFoam` is a transient solver for turbulent flow of compressible fluids, with optional mesh motion and mesh topology changes.
- We will run with a 1D mesh, but you are encouraged to test with 2D and 3D mesh and study the dependency of cell type, cell number, and cell alignment on the numerical solution.
- After finding the numerical solution we will do some sampling.
- Then we will do some plotting (using gnuplot or Python) and scientific visualization.

Compressible flows – Sod's shock tube

Boundary and initial conditions

- The boundary and initial conditions are defined as follows.



All walls are slip

$$\mathbf{U_4} = \mathbf{U_1} = 0$$

$$p_4 = 1, \quad p_1 = 0.1$$

$$T_4 = 0.00348, \quad T_1 = 0.00278$$

Note: the ratio of specific heats is equal to 1.4 and the working fluid is air.

Compressible flows – Sod's shock tube

Selecting thermophysical properties

```
1   thermoType
2   {
3       type            hePsiThermo;
4       mixture         pureMixture;
5       transport       const;
6       thermo          hConst;
7       equationOfState perfectGas;
8       specie          specie;
9       energy          sensibleEnthalpy;
10  }
11
12  mixture
13  {
14      specie
15      {
16          nMoles      1;
17          molWeight   28.9;
18      }
19      thermodynamics
20      {
21          Cp          1005;
22          Hf          0;
23      }
24      transport
25      {
26          mu          0.0;
27          Pr          0.7;
28      }
29  }
```

- The thermophysical properties are set in the dictionary *thermophysicalProperties*.
- This dictionary file is located in the directory **constant**.
- In the sub-dictionary **thermoType** (lines 1-10), we define the thermophysical models. Many of these options are hardwired with the solver used.
- The **transport** keyword (line 5) concerns evaluating dynamic viscosity. In this case the viscosity is constant.
- The thermodynamic models (**thermo** keyword) are concerned with evaluating the specific heat Cp (line 6). In this case Cp is constant.
- The **equationOfState** keyword (line 7) concerns to the equation of state of the working fluid. In this case, we are using the ideal gas equation model.

$$\rho = \frac{p}{RT}$$

Compressible flows – Sod's shock tube

Selecting thermophysical properties

```
1   thermoType
2   {
3       type            hePsiThermo;
4       mixture         pureMixture;
5       transport       const;
6       thermo          hConst;
7       equationOfState perfectGas;
8       specie          specie;
9       energy          sensibleEnthalpy;
10  }
11
12  mixture
13  {
14      specie
15      {
16          nMoles      1;
17          molWeight   28.9;
18      }
19      thermodynamics
20      {
21          Cp          1005;
22          Hf          0;
23      }
24      transport
25      {
26          mu          0.0;
27          Pr          0.7;
28      }
29  }
```

- The form of the energy equation to be used is specified in line 9 (**energy**).
- In this case we are using enthalpy formulation (**sensibleEnthalpy**). In this formulation, the following equation is solved,

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho \mathbf{u} h) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho \mathbf{u} K) - \frac{\partial p}{\partial t} = \nabla \cdot (\alpha_{eff} \nabla e) + \rho \mathbf{g} \cdot \mathbf{u} + S$$

- In the sub-dictionary **mixture** (lines 12-29), we define the thermophysical properties of the working fluid (air in this case).
- In line 17, we define the molecular weight.
- In line 21, we define the specific heat **Cp**. The heat of formation **Hf** is defined in line 22 (not used in this case).
- As we are using the transport model **const** (line 5), we need to define the dynamic viscosity **mu** and Prandtl number **Pr** (lines 26 and 27).
- As we want to solve the Euler equations, we set the viscosity to zero (line 26). We also define the Prandtl number in line 27 but is not used in this case as we are solving the Euler equations.

Compressible flows – Sod's shock tube

Selecting turbulence model

- As we are solving the Euler equations (no viscosity), there is no turbulence involved.
- Nevertheless, we need to set the turbulence model to laminar in the dictionary *turbulenceProperties*.
- This dictionary is located in the directory **constant**.

```
simulationType    laminar;
```

- At this point we are done with the physical properties.
- Let us define the discretization schemes and solution method.

Compressible flows – Sod's shock tube

Selecting the discretization schemes

```
1     ddtSchemes
2     {
3         default         Euler;
4     }
5
6     gradSchemes
7     {
8         default         leastSquares;
9         grad(U)         cellLimited leastSquares 1.0;
10    }
11
12    divSchemes
13    {
14        default         none;
15        div(phi,U)      Gauss MinmodV;
16        div(phi,K)      Gauss limitedLinear 1;
17        div(phi,h)      Gauss limitedLinear 1;
18        div(phid,p)     Gauss limitedLinear 1;
19       div((nuEff*dev2(T(grad(U))))) Gauss linear;
20    }
21
22    laplacianSchemes
23    {
24        default         Gauss linear limited 1;
25    }
26
27    interpolationSchemes
28    {
29        default         linear;
30    }
31
32    snGradSchemes
33    {
34        default         limited 1;
35    }
```

- The discretization schemes are set in the dictionary *fvSchemes* located in the directory **system**.
- When dealing with compressible flows and strong discontinuities (such as shock waves), it is of paramount importance to set a robust and accurate numerics, as the one used in this case.
- In line 3, we define the time discretization scheme. In this case we are using the Euler method.
- In lines 6-10 we define the gradient discretization schemes.
- In line 8, we define the discretization scheme to be used with all variables (the default **keyword**), in this case, **grad(U)**, **grad(p)**, and **grad(h)**.
- When dealing with shock waves, it is recommended to use an aggressive limiter for **grad(U)** (line 9).
- Be careful not to add very aggressive limiters to **grad(p)** and **grad(h)** (line 8), as they may add a lot of numerical diffusion.

Compressible flows – Sod's shock tube

Selecting the discretization schemes

```
1   ddtSchemes
2   {
3       default         Euler;
4   }
5
6   gradSchemes
7   {
8       default         leastSquares;
9       grad(U)         cellLimited leastSquares 1.0;
10  }
11
12  divSchemes
13  {
14      default         none;
15      div(phi,U)      Gauss MinmodV;
16      div(phi,K)      Gauss limitedLinear 1;
17      div(phi,h)      Gauss limitedLinear 1;
18      div(phid,p)     Gauss limitedLinear 1;
19      div((nuEff*dev2(T(grad(U))))) Gauss linear;
20  }
21
22  laplacianSchemes
23  {
24      default         Gauss linear limited 1;
25  }
26
27  interpolationSchemes
28  {
29      default         linear;
30  }
31
32  snGradSchemes
33  {
34      default         limited 1;
35  }
```

- In lines 12-20 we define the discretization scheme of the convective terms.
- Notice that for velocity (line 15) we are using a TVD scheme.
- TVD schemes are highly recommended when you are dealing with strong discontinuities (such as shock waves).
- In lines 16-18 we define the discretization schemes for the variables related to the energy equation. In general, the setup used is accurate and stable.

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho \mathbf{u} h) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho \mathbf{u} K) - \frac{\partial p}{\partial t} = \nabla \cdot (\alpha_{eff} \nabla e) + \rho \mathbf{g} \cdot \mathbf{u} + S$$

`div(phi,h)` `div(phi,K)`

- Line 18 is related to the transonic correction used. this correction is set in the dictionary *`fvSolution`*.
- Line 19 is related to the Reynolds stresses (not relevant as we are solving the Euler equations).

Compressible flows – Sod's shock tube

Selecting the solution method and linear solvers

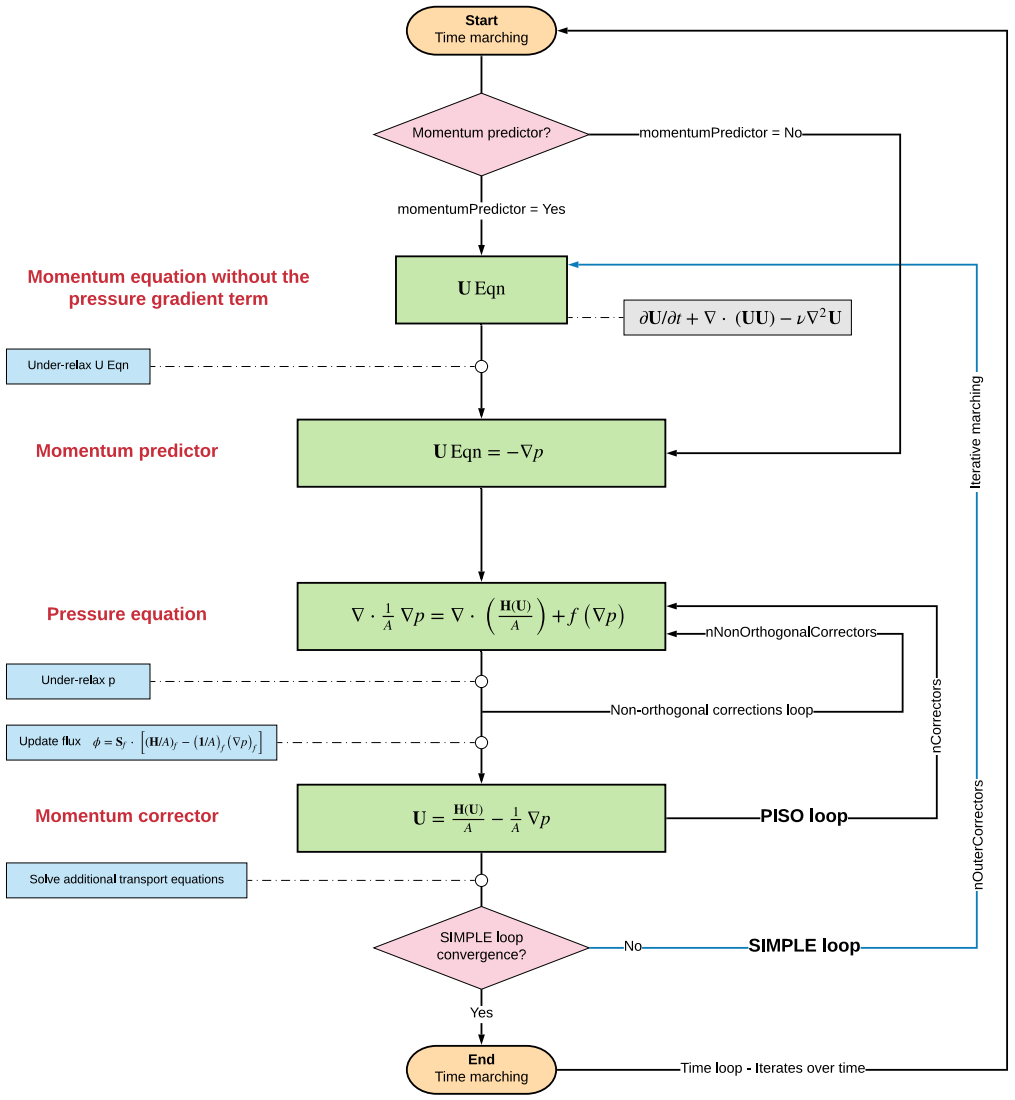
```
1     solver
2     {
3         "(p|U|e|h).*"
4         {
5             solver          PBiCGStab;
6             preconditioner  DILU;
7             tolerance       1e-06;
8             relTol          0.001;
9              minIter         2;
10        }
11
12        "rho.*"
13        {
14            solver          diagonal;
15        }
16
17    }
18
19    PIMPLE
20    {
21        transonic yes;
22        consistent yes;
23        nOuterCorrectors 3;
24        nCorrectors 1;
25        nNonOrthogonalCorrectors 1;
26    }
27
28    relaxationFactors
29    {
30        fields
31        {
32            ".*" 1;
33        }
34        fields
35        {
36            ".*" 1;
37        }
38    }
39
```

- The solution method, corrections and linear solvers are set in the dictionary *fvSolution* located in the directory **system**.
- In this case, we are using the linear solver **PBiCGStab** for all variables except **rho** (lines 3-10).
- In compressible solvers, **rho** is computed from the thermodynamical variables, therefore, we use a diagonal solver, in other words, back substitution (line 14).
- In line 21, we enable the transonic correction (for high speed compressible flows).
- In line 22, we enable the **SIMPLEC** method used in the **PIMPLE** loop.
- In lines 23-25 we define the number of corrector steps to perform in the **PIMPLE** loop.
- Finally, in lines 28-38, we define the under-relaxation factors (URF).
- In this case, we define all the URF to one (this will help in increasing the diagonal dominance of the matrix of coefficients).
- To improve stability, you can use smaller URF values, but you might loose temporal accuracy.

Compressible flows – Sod's shock tube

Selecting the solution method and linear solvers

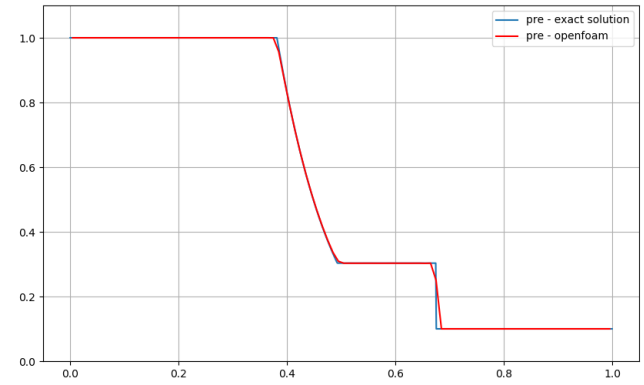
- The rationale of the **PIMPLE** loop in OpenFOAM® is shown in this workflow.
- The traditional **PISO** loop will compute the momentum corrector (velocity field) from the pressure corrector (pressure-Poisson equation).
- However, these two corrector steps depend on the information coming from the predictor step (momentum predictor).
- In the **PIMPLE** loop, we have the option to use the solution of the corrector steps to compute better approximations of the predictor steps.
- This increases the accuracy and stability of the solution, but at the cost of increasing the computing time (in a way almost proportional to the number of corrections).
- In this case, it is necessary to use better predictor estimates for the corrector steps.
- That is reason why we are looping 3 times in the **SIMPLE** loop (**nOuterCorrectors**).
- If you use one or two **nOuterCorrectors**, you will notice that the solution is less accurate.
- Most of the times is not necessary to use more than two or three **nOuterCorrector** steps.



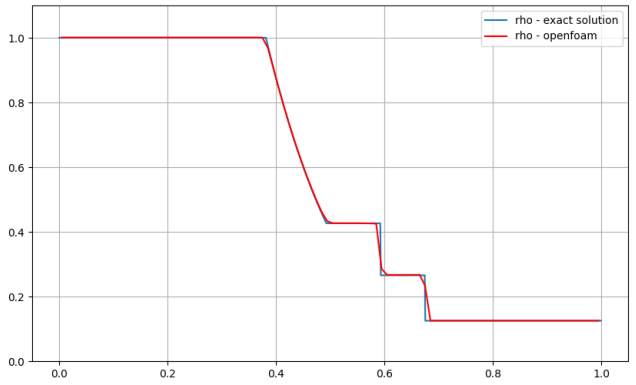
Compressible flows – Sod's shock tube

Comparison of the numerical solution against the exact solution

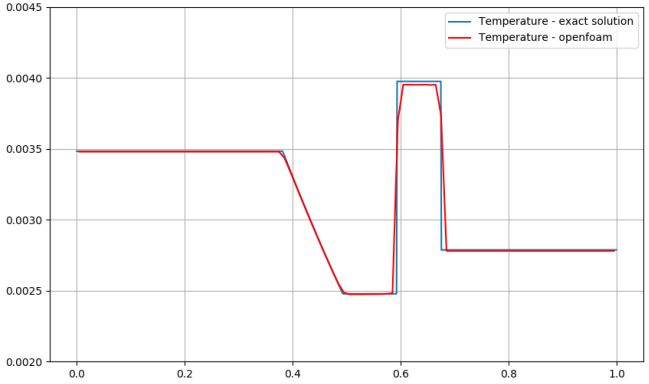
- The solutions are compared at a physical time of 0.1 seconds and in a line along the horizontal axis.



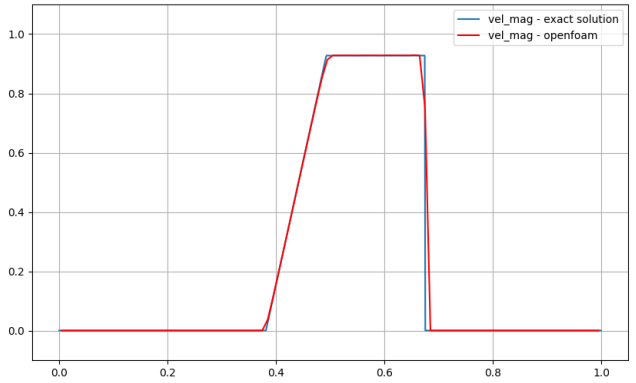
Pressure field



Density field



Velocity magnitude field



Temperature field

Compressible flows – Sod's shock tube

Visualizing the solution in Paraview

- At this point, you can visualize the solution using Paraview.

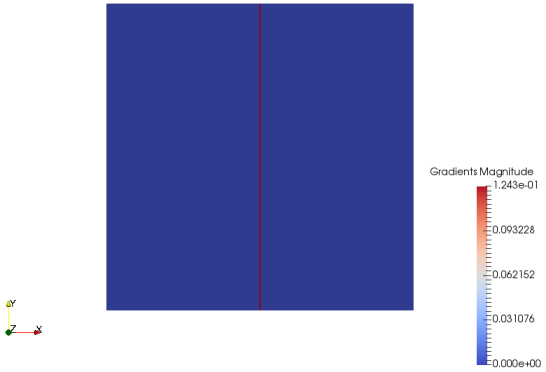
www.wolfdynamics.com/wiki/shocktube/aniT.gif

Temperature field

www.wolfdynamics.com/wiki/shocktube/aniU.gif

Velocity field

www.wolfdynamics.com/wiki/shocktube/anigt.gif



Temperature gradient

www.wolfdynamics.com/wiki/shocktube/anip.gif

Pressure field

Compressible flows – Sod's shock tube

Running the case

- Let us first generate the mesh using the meshing utility `blockMesh`.
- In the terminal window type:

```
1. $> foamCleanTutorials
2. $> rm -rf 0 > /dev/null 2>&1
3. $> cp -r 0_org 0 > /dev/null 2>&1
4. $> blockMesh
5. $> checkMesh
```

- The dictionary *`blockMeshDict`* has been already parametrized.
- In this case we are using a 1D mesh with 5000 cells.
- If you want to try a different cell count, feel free to modify the dictionary *`blockMeshDict`*.

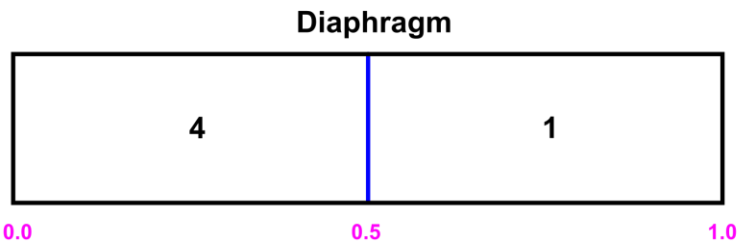
Compressible flows – Sod's shock tube

Running the case

- Let us initialize the fields using the utility `setFields`.
- In the terminal window type:

```
1.  $> setFields
```

- The utility `setFields` will initialize the solution according to the figure below.



$$\mathbf{U_4} = \mathbf{U_1} = 0$$

$$p_4 = 1, \quad p_1 = 0.1$$

$$T_4 = 0.00348, \quad T_1 = 0.00278$$

Compressible flows – Sod's shock tube

Running the case

- You will find this tutorial in the directory **`$PTOFC/sod_shock_tube`**
- In the terminal window type:

1. `$> rhoPimpleFoam | tee log.solver`
2. `$> postProcess -func 'mag(U)'`
3. `$> postProcess -func 'components(U)'`
4. `$> postProcess -func sampleDict`

- In this case we are running a fully transient simulation.
- For good accuracy, it is recommended to run this case with a CFL in the order of 0.5.
- Also, it is not recommended to use adaptive time stepping as this method might introduce instabilities in the solution.

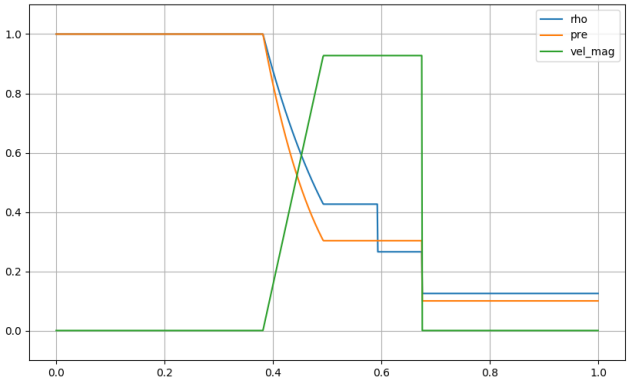
Compressible flows – Sod's shock tube

Running the case

- To compare the numerical solution with the exact solution, type in the terminal,

1. ```
   $> python python/sodshocktube.py
   ```

- To run the python script, you must use Python 3.
- At this point, you will find in the case directory five figures with the comparison of the results.
- The solutions are compared at a physical time of 0.1 seconds and in a line along the horizontal axis.



Exact solution
   ```