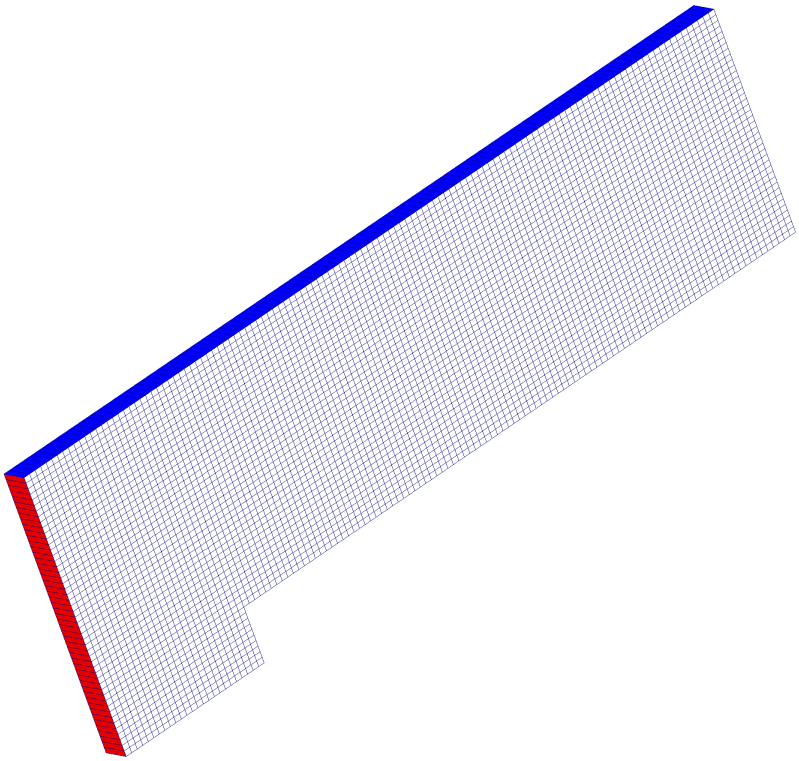
Tutorial Two

Built in Mesh



5th edition, Sep. 2019

Editorial board:
- Bahram Haddadi
- Christian Jordan
- Michael Harasek

Compatibility:
- OpenFOAM® 7
- OpenFOAM® v1906

Cover picture from:
- Bahram Haddadi

Contributors:
- Bahram Haddadi
- Clemens Gößnitzer
- Jozsef Nagy
- Vikram Natarajan
- Sylvia Zibuschka
- Yitong Chen

ISBN 978-3-903337-00-8

Publisher: chemical-engineering.at

For more tutorials visit: www.cfd.at

Background

1. What is mesh?

The partial differential equations that describe fluid flow and heat transfer are the conservation equations of mass, energy and momentum. However we are usually unable to solve them analytically, except in very simple cases. This is when discretization comes in. The flow region is broken up into smaller sub-regions, with the equations solved in each sub-region. One of the methods used to solve the equations is the finite volume method, which we will cover in detail below. The sub-regions are later on referred to as grid cells, with a collection of grid cells forming a mesh.

2. Finite volume method

As discussed in Tutorial One, OpenFOAM® uses finite volume method. The starting point of this method is the transport equation for a conserved fluid property ϕ, for example pressure and flow velocity. It is shown below as:

$$\frac{\partial(\rho\varphi)}{\partial t} + \nabla \cdot (\rho\varphi \boldsymbol{u}) = \nabla \cdot (\Gamma \nabla\varphi) + S_\varphi$$

Rate of change of φ inside fluid element	+	**Net rate of flow of φ out of fluid element**	=	**Rate of change of φ due to diffusion**	+	**Rate of change of φ due to sources**

As you can see, the transport equation is essentially a manifestation of conservation of a fluid flow property within the problem domain.

The key step of the finite volume method is the integration of the transport equation over a three-dimensional control volume (CV). At discrete places values are calculated on a meshed geometry. The small volume which surrounds each node of the mesh is the grid cell.

In each grid cell, the volume integral of the divergence term is replaced by a surface integral, using the Gauss divergence theorem. These terms are then evaluated as fluxes at the surfaces. This not only ensures the conservation of fluxes entering and exiting the grid cell, but allows for easy formulation of the balances on unstructured meshes.

In time-dependent problems, it is also necessary to integrate with respect to time t over a small interval Δt from, say, t until $t + \Delta t$. This yields the most general integrated form of the transport equation.

$$\int_{\Delta t} \frac{\partial}{\partial t}\left(\int_{CV} \rho\varphi \, dV\right) dt + \int_{\Delta t} \int_A \boldsymbol{n} \cdot (\rho\varphi \boldsymbol{u}) \, dA dt = \int_{\Delta t} \int_A \boldsymbol{n} \cdot (\Gamma \nabla\varphi) \, dA dt + \int_{\Delta t} \int_{CV} S_\varphi dV \, dt$$

3. Discretization of transport equations

Discretization of the transport equations is critical to the finite volume method, as it provides a more cost-effective and rapid approach to numerical evaluation on digital computers. Discretization is done through the use of the mesh, which involves dividing the domain into smaller regions.

The mesh used in OpenFOAM® can be simple grid structures based on the Cartesian co-ordinate system, or complex unstructured grid arrangement that can handle curvature and geometric complexity. The mesh is generated using the in-house OpenFOAM® tool (*blockMesh* and *snappyHexMesh*) or external software, such as GAMBIT®. In this tutorial, we are going to learn how to use the *blockMesh* tool in OpenFOAM®. Refer to Tutorial Twelve for information on the *snappyHexMesh* tool.

For this tutorial, we chose to run the *sonicFoam* solver, which is slightly more complicated than *icoFoam*, as it analyses the flow of a compressible gas/fluid.

4. rhoPimpleFoam solver

rhoPimpleFoam is a transient solver. It solves trans-sonic/supersonic, turbulent flow of a compressible gas/fluid.

OpenFOAM® v1906: *sonicFoam* should be used for compressible trans-sonic/supersonic simulations!

rhoPimpleFoam – forwardStep

Tutorial outline

Using rhoPimpleFoam solver, simulate 10 s of flow over a forward step.

OpenFOAM® v1906: use sonicFoam!

Objectives

- Understand blockMesh
- Define vertices via coordinates as well as surfaces and volumes via vertices.

Data processing

Import your simulation into ParaView, and examine the mesh and the results in detail.

1. Pre-processing

1.1. Copying *tutorial*

Copy the tutorial from the following folder to your working directory:

`$FOAM_TUTORIALS/compressible/rhoPimpleFoam/laminar/forwardStep`

OpenFOAM® v1906:
`$FOAM_TUTORIALS/compressible/sonicFoam/laminar/forwardStep`

1.2. Case structure

1.2.1. 0 directory

The file T includes the initial temperature values. Internal pressure and temperature fields are set to 1, and the initial velocity in the domain as well as the inlet boundary is set to 3.

Note: As it can be seen, the p unit is the same as the pressure unit (kg m^{-1} s^{-2}), because rhoPimpleFoam/sonicFoam is a compressible solver.

Note: Do not forget that, this example is a purely numeric example (you might have noticed this from the pressure values).

1.2.2. constant directory

By checking *thermophysicalProperties* file, different properties of a compressible gas can be set:

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}
// Note: these are the properties for a "normalized" inviscid gas
//       for which the speed of sound is 1 m/s at a temperature of 1K
//       and gamma = 7/5
mixture
{
    specie
    {
        molWeight       11640.3;
    }
    thermodynamics
    {
        Cp              2.5;
        Hf              0;
    }
    transport
    {
        mu              0;
        Pr              1;
    }
}

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

In the thermoType, the models for calculating thermo physical properties of gas are set:

- type: Specifies the underlying thermos-physical model.
- mixture: Is the model which is used for the mixture, whether it is a pure mixture, a homogeneous mixture, a reacting mixture or ….
- transport: Defines the used transport model. In this example a constant value is used.
- thermo: It defines the method for calculating heat capacities, e.g. in this example constant heat capacities are used.
- equationOfState: Shows the relation which is used for the compressibility of gases. Here ideal gas model is applied by selecting perfectGas.
- energy: This key word lets the solver decide which type of energy equation it should solve, enthalpy or internal energy.

After defining the models for different thermos-physical properties of gas, the constants and coefficients of each model are defined in the sub-dictionary mixture. E.g. molWeight shows the molecular weight of gas, Cp stands for heat capacity, Hf is the heat of fusion, mu is the dynamic viscosity and Pr shows the Prandtl number.

By opening the turbulenceProperties the appropriate turbulent mode can be set (in this case it is laminar):

```
simulationType   laminar;
```

1.2.3. system directory

In this tutorial the mesh is not imported from other programs (e.g. GAMBIT®). It will be created inside OpenFOAM®. For this purpose the blockMesh tool is used. blockMesh reads the geometry and mesh properties from the *blockMeshDict* file (found in the system directory):

```
>nano blockMeshDict
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
convertToMeters 1;
vertices
(
    (0 0 -0.05)
    (0.6 0 -0.05)
    (0 0.2 -0.05)
    (0.6 0.2 -0.05)
    (3 0.2 -0.05)
    (0 1 -0.05)
    (0.6 1 -0.05)
    (3 1 -0.05)
    (0 0 0.05)
    (0.6 0 0.05)
    (0 0.2 0.05)
    (0.6 0.2 0.05)
    (3 0.2 0.05)
    (0 1 0.05)
    (0.6 1 0.05)
    (3 1 0.05)
);
blocks
(
```

```
    hex (0 1 3 2 8 9 11 10) (25 10 1) simpleGrading (1 1 1)
    hex (2 3 6 5 10 11 14 13) (25 40 1) simpleGrading (1 1 1)
    hex (3 4 7 6 11 12 15 14) (100 40 1) simpleGrading (1 1 1)
);
edges
(
);
boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 8 10 2)
            (2 10 13 5)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (4 7 15 12)
        );
    }
    bottom
    {
        type symmetryPlane;
        faces
        (
            (0 1 9 8)
        );
    }
    top
    {
        type symmetryPlane;
        faces
        (
            (5 13 14 6)
            (6 14 15 7)
        );
    }
    obstacle
    {
        type patch;
        faces
        (
            (1 3 11 9)
            (3 4 12 11)
        );
    }
);

mergePatchPairs
(
);
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
```

As noted before units in OpenFOAM® are SI units. If the vertex coordinates differ from SI, they can be converted with the `convertToMeters` command. The number in the front of `convertToMeters` shows the constant, which should be multiplied with the dimensions to change them to meter (SI unit of length). For example:

```
convertToMeters    0.001;
```

shows that the dimensions are in millimeter, and by multiplying them by 0.001 they are converted into meters.

In the `vertices` part, the coordinates of the geometry vertices are defined, the vertices are stored and numbered from zero, e.g. vertex `(0 0 -0.05)` is numbered zero, and vertex `(0.6 1 -0.05)` points to number 6.

In the `block` part, blocks are defined. The array of numbers in front each block shows the block building vertices, e.g. the first block is made of vertices `(0 1 3 2 8 9 11 10)`.

After each block the mesh is defined in every direction. e.g. `(25 10 1)` shows that this block is divided into:

- 25 parts in x direction
- 10 parts in y direction
- 1 part in z direction

As it was explained before, even for 2D simulations the mesh and geometry should be 3D, but with one cell in the direction, which is not going to be solved, e.g. here number of cells in z direction is one and it's because of that it's a 2D simulation in x-y plane.

The last part, `simpleGrading(1 1 1)` shows the size function.

In the `boundary` part each boundary is defined by the vertices it is made of, and also its `type` and `name` are defined.

Note: For creating a face the vertices should be chosen clockwise when looking at the face from inside of the geometry.

2. Running simulation

Before running the simulation the mesh has to be created. In the previous step the mesh and the geometry data were set. For creating it the following command should be executed from the case main directory (e.g. forwardStep):

```
>blockMesh
```

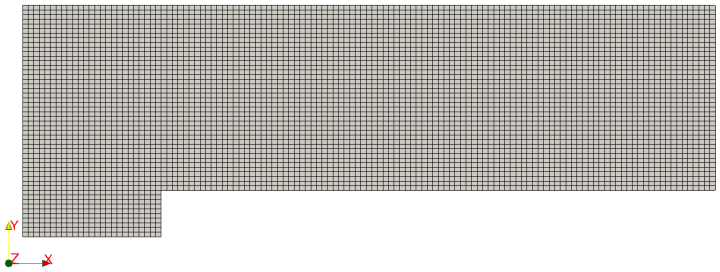
After that, the mesh is created in the constant/polyMesh folder. For running the simulation, type the solver name form case directory and execute it:

```
>rhoPimpleFoam
```

OpenFOAM® v1906: `>sonicFoam`

3. Post-processing

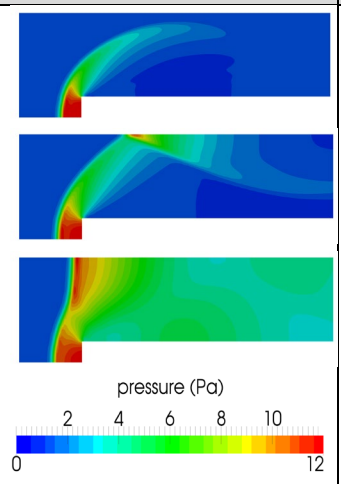
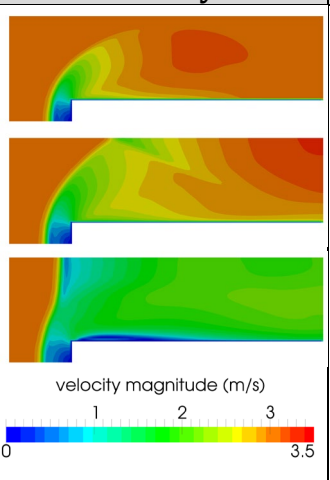
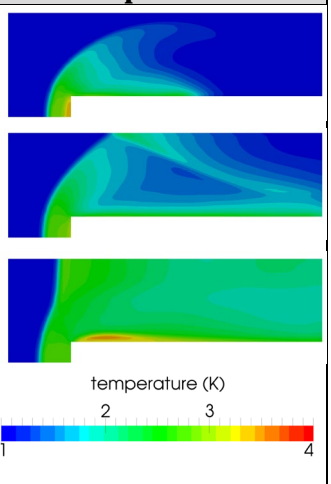
The mesh is presented in the following way in ParaView, and you can easily see the three blocks, which were created.



Mesh generated by blockMesh

Note: When a cut is created by default in ParaView, the program shows the mesh on that plane as a triangular mesh even if it is a hex mesh. In fact, ParaView changes the mesh to a triangular mesh for visualization, where every square is represented by two triangles. For avoiding this when creating a cut in ParaView in the Slice properties window, uncheck "Triangulate the Slice".

The simulation results are as follows:

Time	Pressure	Velocity	Temperature
0.5 s			
1 s			
10 s			

Pressure, velocity and temperature contours at different time steps

ISBN 978-3-903337-00-8