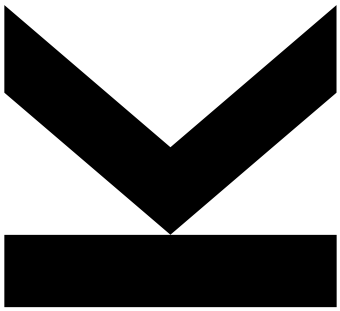
Authors:
Thomas Mitterlehner
Eva Kobler
Georg Steinbichler

Institute:
Institute for Polymer Injection Moulding Technology and Process Automation

Date:
March 2019

MANUAL



Introduction to OpenFoam® and chtMultiRegion using an application-oriented example

JOHANNES KEPLER UNIVERSITY LINZ
Altenberger Straße 69
A-4040 Linz, Austria
www.jku.at
DVR 0093696

Preface

In the course of the work on the AM 4 Industry project, research on the design and simulation of near-contour cooling channels was carried out. One objective was to provide toolmaking engineers with an instrument that could be used to further optimise cooling channel geometries. Particularly in the design of near-contour and irregular cooling channels, simulation using commercial simulation programs may not be sufficient and thus require additional simulation steps. For this additional or supplementary simulation step, the focus was deliberately placed on a non-commercial simulation program. Therefore, the program OpenFoam® (**Open** Source **F**ield **O**peration **a**nd **M**anipulation) was chosen for the simulations. This is freely available and is mainly used to solve flow problems (Computational Fluid Dynamics). It is written in C++ and comes with useful solvers even in the basic version, and a wide variety of further solvers can be adapted. One of the major advantages is that the source code and thus also the algorithms are freely accessible; and another, that the codes and calculations can be extended almost arbitrarily.
Based on an application-oriented example, this manual describes the structure and the performance of a simulation in detail. The solver chtMultiRegion was used for the simulations. This is generally used to calculate the heat exchange between a solid and a fluid. The objective of this manual is to give users in development, simulation engineers and students an application-oriented introduction to OpenFoam®, as well as an overview of how to work with it. The individual steps were grouped into nine chapters with the following contents:

- Chapters 1 and 2 give a general overview of the simulation example as well as of the simulation structures of OpenFoam®

- Chapters 3 to 6 show how a simulation case is set up in OpenFoam®, and which settings are required. The creation of the calculation mesh and the allocation of the surfaces are dealt with in detail.

- Chapters 7 and 8 address the performance of the simulation. The choice of the boundary condition and the theory of flow simulation are dealt with.

- Chapter 9 concludes by discussing the evaluation methods offered by OpenFoam®, and how the results can be exported for further processing.

Finally, it should be mentioned that the OpenFoam® environment may seem strange at first glance, especially for users accustomed to Windows® operating systems. Patience is

definitely required here. It simply takes time for the program' s processes to be perceived as logical, and to become proficient in the commands necessary for the operation and execution of the program. However, it will certainly be worthwhile to get deeply into this topic, as it offers the possibility to further refine one' s simulations and make better predictions. This may in turn provide a clear competitive advantage. Thus: Happy Foaming!

Contents

Preface 2
Contents 3
1. Introduction 4
2. Preparation 7
2.1. Construction of the simulation geometry 7
2.2. Control and post-processing 8
3. Pre-processing with OpenFoam® 13
3.1. Mesh creation – General 14
4. First calculation mesh – blockMesh 16
5. Second calculation mesh – snappyHexMesh 21
6. Creation and allocation of the regions 26
7. Solving – Simulation with chtMultiRegion 32
7.1. Theory of flow simulation 32
7.2. Input parameters 33
7.2.1. Input parameters Laminar 34
7.2.2. Input parameters Turbulent 34
8. Running the simulation 38
9. Results 39
9.1. Heating and cooling process 39
9.2. Export of measured values 43
10. Annex 44
10.1. .....
Folder 0 > fluid 44
10.2. .....
Folder 0 > solid 48
10.3. .....
Folder Constant > fluid 49
10.4. .....

Folder Constant > solid 52

11. References 54

1. Introduction

The OpenFoam software package usually runs on Linux. In the case described here, on Ubuntu version 14.04.5 LTS. For performance of simulations, installation of a Linux operating system is necessary. The following options are available for operating OpenFoam or a Linux system:

- Direct installation of OpenFoam on a computer with a Linux operating system
- Dual boot: Two or more operating systems (e.g. Windows and Ubuntu) are installed on one computer, and upon each system start you can choose which operating system to start with.
- Use of a server: Installation of a Linux operating system on a separate computer which can be accessed via a network with e.g. Putty.

It should also be noted here that OpenFoam does not have a user interface (GUI) by default. This means that all commands must be entered via the command line. However, there are commercially available GUIs that can make operation easier. These include for example CAESES [1], Visual-CFD [2], SimFlow [3], or MantiumFlow [4]. Figure 1 shows the general structure of OpenFoam.

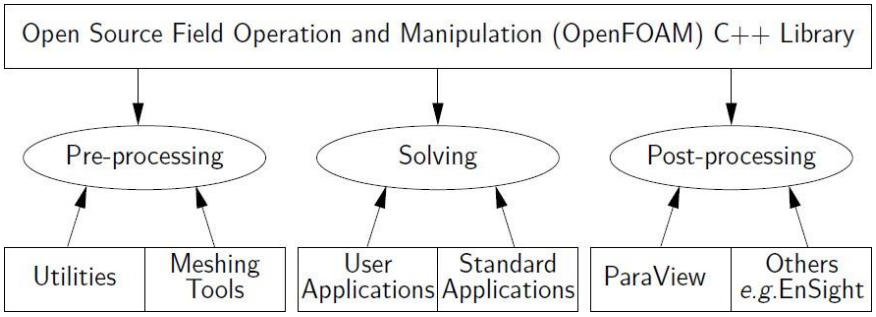


Figure 1: Overview of the structure of OpenFoam [5]

This consists of the three sub-areas of pre-processing, solving and post-processing. Pre-processing does not include the programs such as Blender or PTC Creo, but rather the software packages that prepare the geometry for the solvers. These include for example blockMesh or snappyHexMesh. The solvers comprise the software packages that perform the calculations or simulations, respectively. OpenFoam comes by default with many solvers, and more can be added from various internet sources or forums. The solver chtMultiRegion is integrated into OpenFoam by default and can be found among the "Heat transfer and buoyancy-driven flows" solvers.

This very chtMultiRegion solver is the focus of this manual. The processes are described step by step using an illustrative example. For this purpose, an application from plastics

processing or injection moulding technology, respectively, was selected. The simulation is intended to help visualise the temperature distribution inside an injection mould. More precisely, a so-called variothermal injection moulding process is described. As a matter of principle, an injection moulding process is carried out in such a way that plastic granulate, which is solid at room temperature, is molten down in a heated cylinder. This is done at approx. 200 °C. After plasticising, the plastic material is liquid and is injected into a metal mould at approx. 500 bar.

The metal mould, cooled by means of temperature control units, provides the geometric moulding on the one hand and the cooling of the moulded part on the other. However, if the moulded part has very subtle structures (microstructures), the conventional mould cooling concept is not sufficient. The variothermal process mentioned above is then applied. Here, the metal mould is heated approximately to the melting temperature before injection. This is then cooled down again already during the injection process. Figure 2 shows the events during the filling process of both process variants.

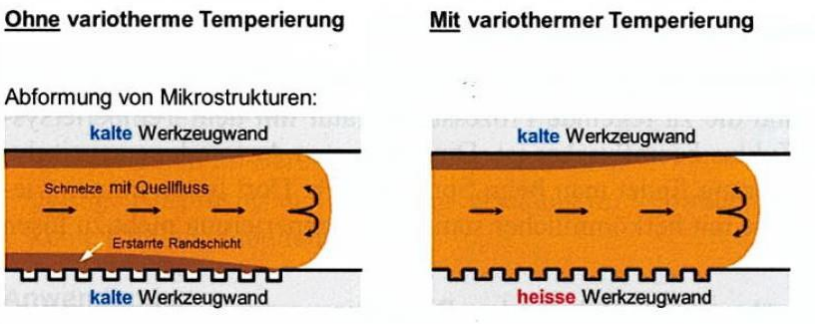


Figure 2: Filling process during injection moulding, comparison without and with variothermal temperature control [6]

Here it can be seen that the plastic melt solidifies upon contact with the cold mould wall. This reduces the ability to mould microstructures. However, if the mould wall is heated, the plastic melt has sufficient time to creep into the fine structures. The heating and cooling process is carried out by two separate cooling units. Water or oil is usually used as the cooling or heating medium, respectively. The following figure shows an example of such a temperature control concept.

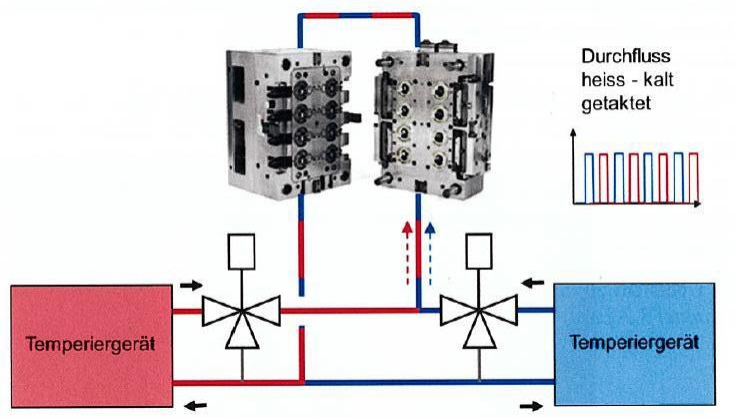


Figure 3: Principle of variothermal liquid temperature control with two temperature control units and one valve switching unit [6]

In order to be able to manufacture high-quality plastic products, it is essential to know the exact mould wall temperature. For this reason, it is worthwhile to simulate this process with OpenFoam and take a closer look at it.

This manual is intended to serve as the first basis for mapping the described processes using OpenFoam. For an initial overview, Table 1 shows the individual steps of the approach, grouped into preparation, pre-processing, solving and post-processing. With regard to modelling and the integration of geometric data into OpenFoam, a wide variety of procedures are available. The simulation case described in this paper is solved by the following approach:

Table 1: Procedure for simulation with OpenFoam and chtMultiRegion

Step	Description	Classification
1	Construction of the simulation geometry with **PTC Creo** and export of the CAD model as *.stl file	Preparation (page 7)
2	Control and post-processing of the CAD model with **Blender**	Preparation (page 8)
3	Creation of a first calculation mesh with **blockMesh**	Pre-processing (page 16)
4	Refining of the calculation mesh with **snappyHexMesh**	Pre-processing (page 21)
5	Assignment of the regions with **splitMeshRegions** and **surfaceToPatch**	Pre-processing (page 26)
6	Simulation of heat flow and exchange, respectively, between tool and coolant with **chtMultiRegion**	Solving (page 32)
7	Presentation of the results with **paraView**	Post-processing (page 39)

The geometry chosen for the simulation example is an oblong cuboid having a temperature control bore in the upper half of its end face. Water at a defined speed and temperature is planned to flow through this borehole. The goal is to determine the heating and cooling process on the upper surface. The geometry of the simulation is graphically displayed in the figure below.

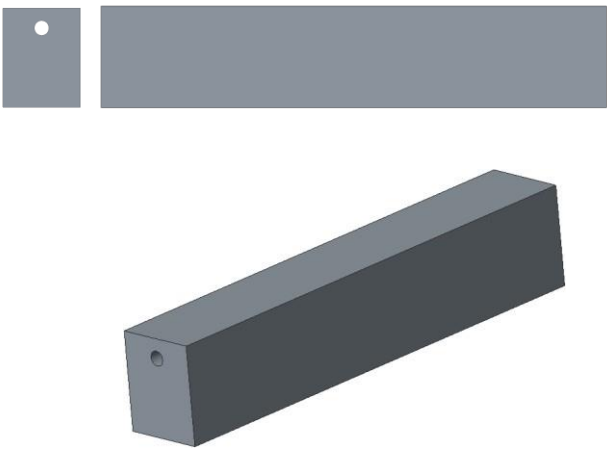


Figure 4: Illustration of the simulation geometry

In the following chapters, the simulation example described is documented step by step. The commands to be executed are also mentioned in each case. These are displayed in colour, with *PTC Creo commands shown in blue*, Blender commands in yellow and *OpenFoam commands in red*.

2. Preparation

Preparation includes all steps that do not need to be performed with OpenFoam or in a Linux environment. In this manual or in the example shown, respectively, the design activities were carried out using PTC Creo. The construction is not limited to the program Creo, but can be chosen arbitrarily. The only precondition is that the
*.stl format must be used to pass the geometry to OpenFoam. Since errors can often occur when exporting the geometry, this is then additionally checked with the freely available program Blender [7] and repaired if necessary. A typical error is, for example, a surface that is not closed.

2.1. Construction of the simulation geometry

The first step was to design the basic geometry with PTC Creo. As geometry to be simulated, the steel block shown in Figure 4 was chosen, which has a longitudinal bore. This is intended to represent a strongly simplified design of an injection mould with a cooling bore running through it. The following figure shows the dimensions of this basic geometry.

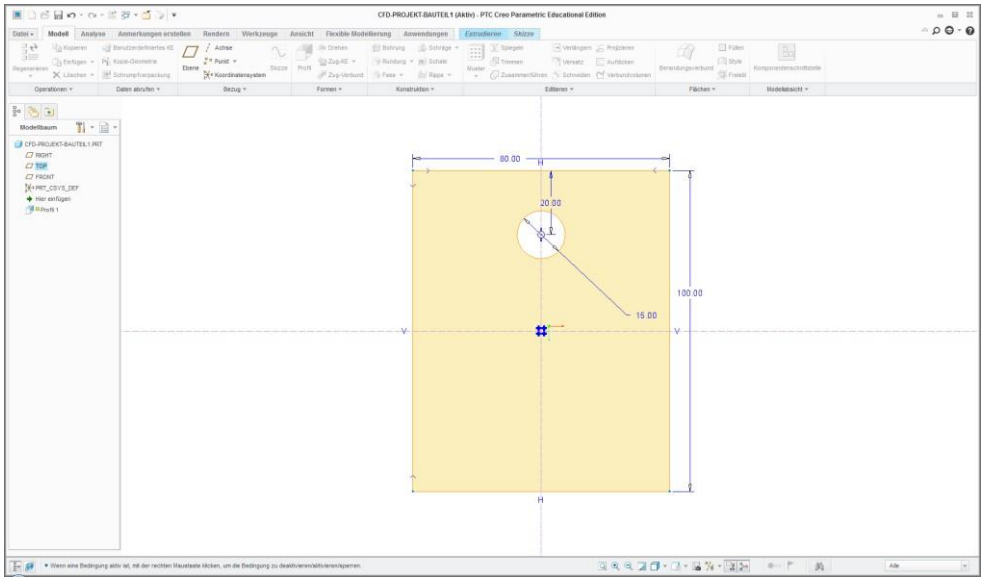
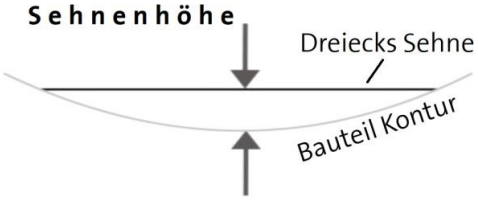


Figure 5: Basic geometry of the simulation model in the CAD program PTC Creo

The width of the basic geometry is 80 mm, and the height 100 mm. The borehole has a diameter of 15 mm and was placed 20 mm beneath the upper surface. This basic geometry was extruded to a length of 500 mm. Steel 1.2343 (X38 CrMoV 51) was selected as the material, because it is typically used for mould inserts in injection moulds.

Subsequently, this model was exported as an *.stl file. The export function can be found in the Creo CAD program under
File > Save As > Save Copy

Then the saving explorer appears. Here select the type "Stereolithography (STL)". After confirmation with "OK", a window appears in which export settings can be selected. For the later processing with OpenFoam, it is necessary that the format is set to "ASCII" (American Standard Code for Information Exchange). This determines whether the file should be



readable only in machine language (binary) or also with a text editor (ASCII). In addition, the chord height and the angle control must be adjusted. A value in the range of 0.01 must be entered for the chord height, and 1 for the angle control. By specifying the chord height, the maximum distance between a chord and a surface is specified. The lower the chord height, the smaller the deviation from the actual component surface. For better understanding, an explanation is shown in Figure 6.

Figure 6: Explanation of chord height [8]

It should also be noted that the smaller the chord height is, the larger the exported *.stl file will become. The angle control can be adjusted over a range from 0 to 1. This value determines the adjustment of the chord height for small radii. The value 1 causes an exact adjustment of the triangles to small radii. By contrast, a value of 0 does not result in an adjustment. The following Figure 7 shows a comparison of two different settings.

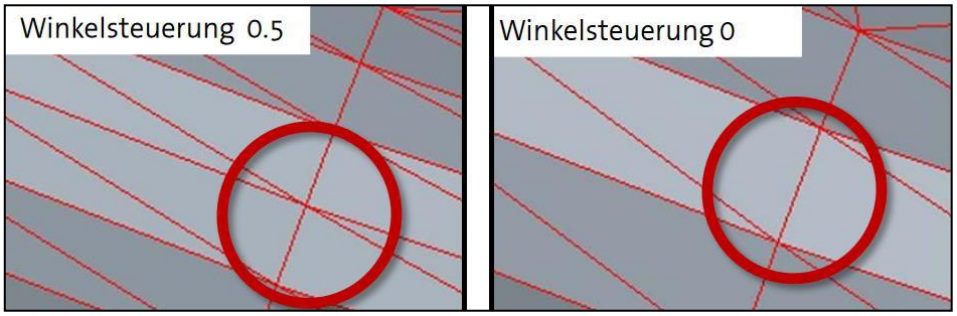


Figure 7: Comparison of two angle control settings [8]

When the format, chord height and angle control have been set, confirm the settings with "Ok", and the geometry will be exported as an *.stl file.

2.2. Control and post-processing

If the geometry is available as an *.stl file, work can begin in Blender. First the geometry must be imported. This can be done with
File > Import.
For optimal work with the model, the correct mode must first be set. The initial default setting is "Object Mode". This must be changed to "Edit Mode". This can be set in the lower screen area

(Figure 8).

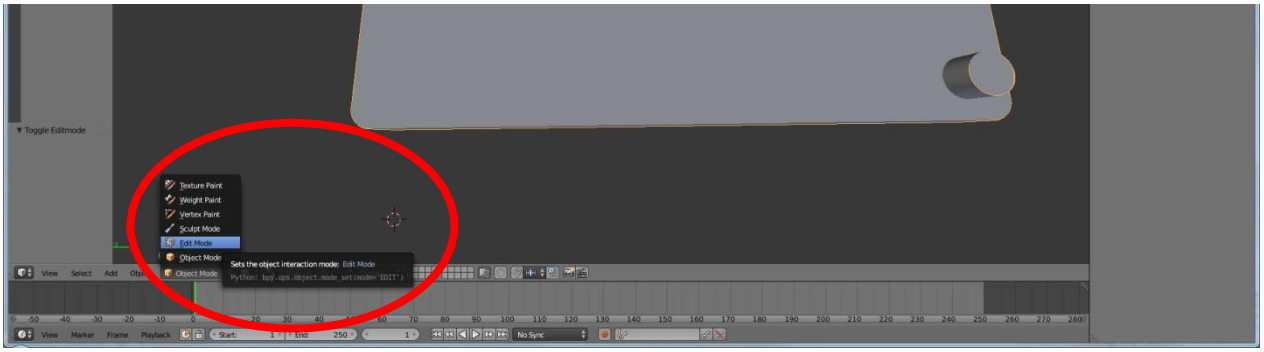


Figure 8: Selecting Edit Mode in Blender

In addition, the selection type must be set to "Face". This can in turn be selected at the bottom of the screen. With
key "5"
on the numeric keypad, the zoom behaviour can also be improved. Once these settings have been made, a surface can be selected with the right mouse button. To select a plurality of surfaces, press and hold the "Shift" key.
In order for OpenFoam to work with the CAD geometry, open geometric surfaces must first be closed. In the case of the geometry shown, for example, the geometry of the cooling channel inlet and outlet is not closed. To close these openings, the filling function is a suitable choice. To do this, first change the selection mode to point selection (Vertex select). Then press
key "c"
whereupon a selection circle appears. When the left mouse button is now pressed, all points in the selection circle are marked. The selection circle can be reduced or enlarged with the mouse wheel. Once all the necessary points have been selected, press the right mouse button to end the selection, and then
key "f"
to fill the surface enclosed by the points.

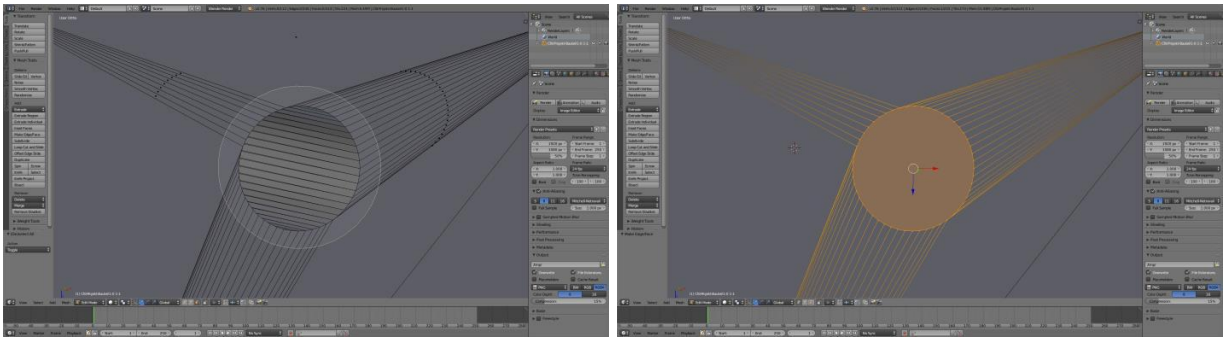


Figure 9: Selection circle by pressing the "c" key (left), filling the surface by pressing the "f" key

Between the individual fill commands, it must always be ensured that no other surfaces are selected. In order to make sure that no other items are selected, between different

commands
key "a"
should be actuated to deselect any selected points. After successfully closing all open geometries, you can use the function
Non Manifold
to check whether all the surfaces are closed. This function can be found at the bottom left of the screen at
Select > Non Manifold or Select > Select All by Trait > Non Manifold,
or by pressing Shift-Ctrl-Alt-M
If Edit Mode is activated and the view is set to Wireframe, the open areas are highlighted. By pressing
key "n"
the XYZ coordinates of the open area can also be displayed.

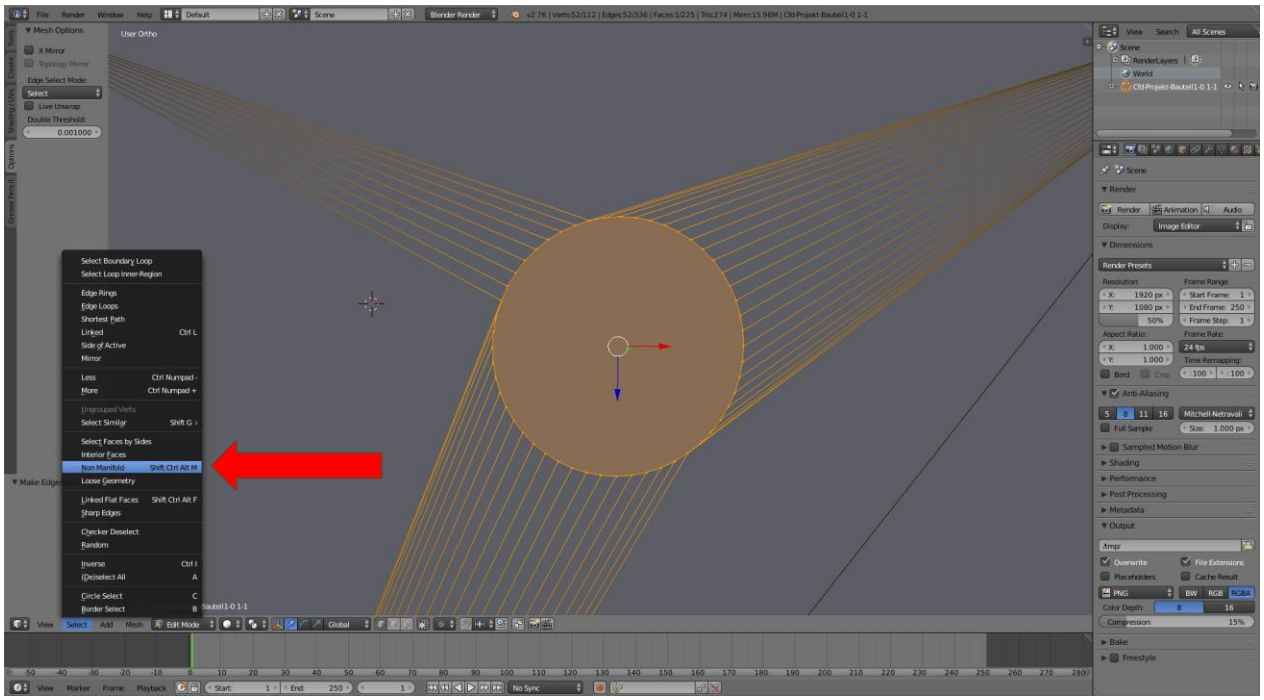


Figure 10: Menu Item Non Manifold

Once it has been ensured that the geometry is closed, the individual surfaces must be defined so that they can later be assigned certain properties in OpenFoam, such as the inlet and outlet of the coolant. This means that each surface that is to have its own property must be exported separately as an *.stl file. This includes the inlet, the outlet, the cooling tube and the steel block. To do this, switch to the surface selection mode and select the surface to be defined. Then press
key "p"
and select Section. The selected surface then appears separately in the structure tree on the right-hand side. Now each surface can be assigned a name. The following definitions are important for OpenFoam:

- inlet
- outlet
- tube
- wall

In Chapter 6, the surfaces and regions to be exported are described in detail. If individual partial surfaces are overlooked during selection of the surfaces, they can be added afterwards. Select the overlooked surfaces and press key "p". Then select the overlooked surfaces and the area where they are to be added, and click on
Tools > Edit > join
to execute the join command. This merges the two selected surfaces. In order for the model to be properly simulated, the following *.stl files must be available for import into OpenFoam:

- inlet (single)
- outlet (single)
- tube (single)
- wall
  - (all that do not belong to inlet, outlet or wall)
- solid
  - (wall – all the surfaces that should later have the same solid boundary condition)
- fluid
  - (inlet, outlet and wall – all the surfaces that should later have the fluid boundary condition)

In order to export a plurality of surfaces as a single *.stl file (e.g. fluid), they must be selected on the right-hand side in the structure tree. Selection of a plurality of files is done with
Shift + left mouse button
After selection of the individual surfaces, they can be exported under File > Export > STL (*.stl).
When exporting, make sure that in the export menu the check mark is set at "ASCII".

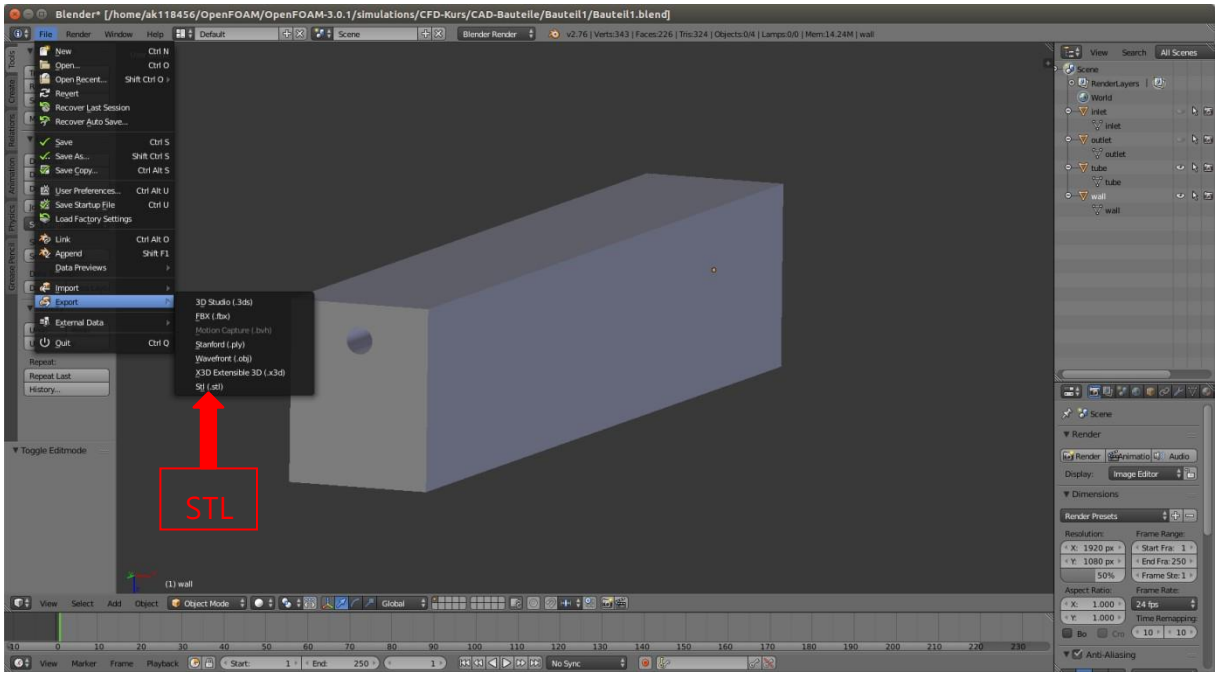


Figure 11: Blender menu item of the *.stl export

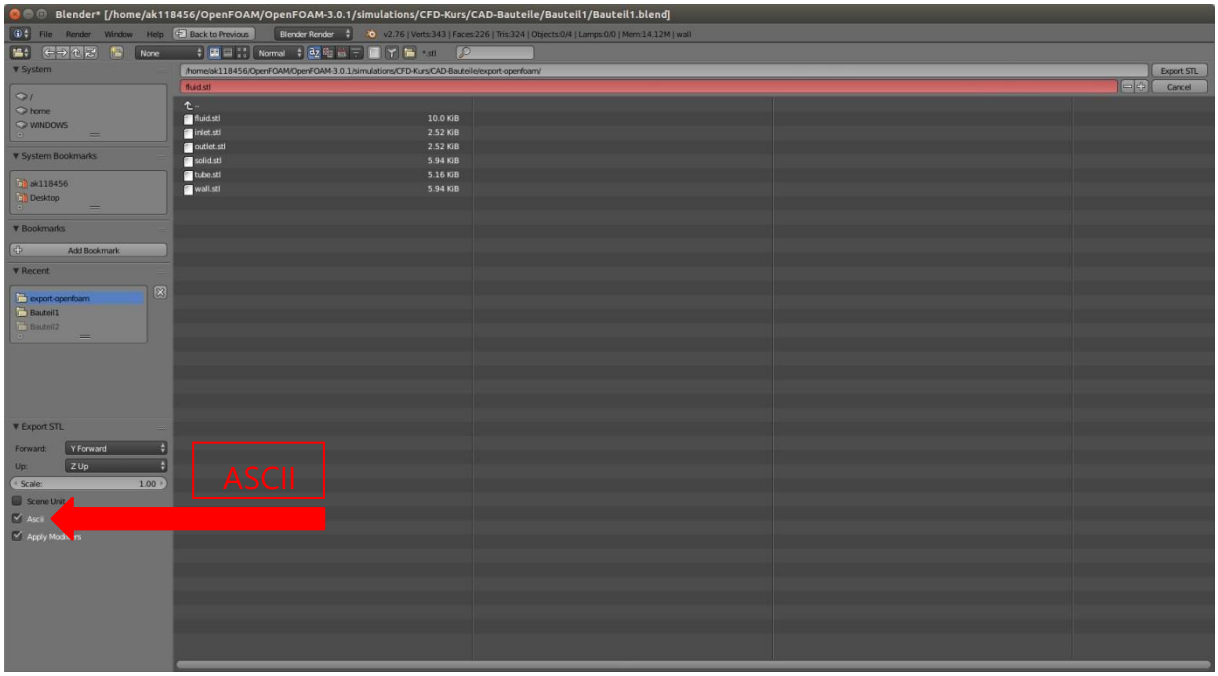


Figure 12: Blender *.stl export settings, ASCII

3. Pre-processing with OpenFoam®

In order to be able to start pre-processing in OpenFoam, all the geometries must be separately available, i.e. inlet.stl, outlet.stl, tube.stl, wall.stl, fluid.stl and solid.stl must be on hand.
Once all geometry files are available, construction of the simulation can be started. The case described in this simulation example is intended to show the heating and cooling process of a steel block through which a medium flows. To this end, the geometry is first meshed using blockMesh and snappyHexMesh. The temperature distribution is then to be displayed using the solver "chtMultiRegionFoam".

Simulations performed with OpenFoam often have the following standard structure:

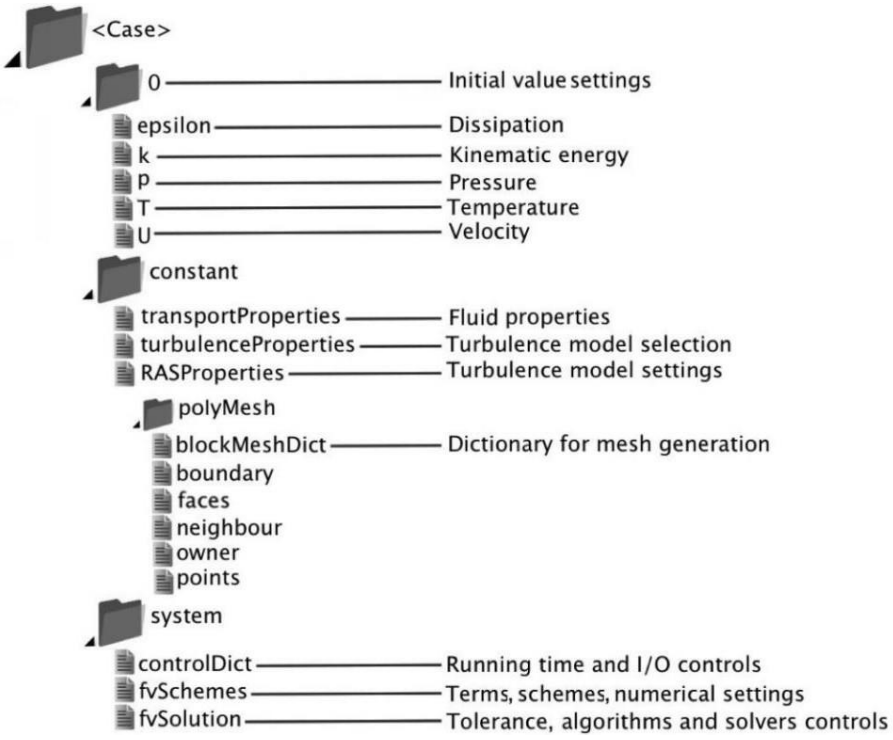


Figure 13: Folder structure of most OpenFoam simulations

When starting to set up a simulation in OpenFoam, it is advisable to copy an existing simulation case and adapt it. For most solvers, there are examples in the standard installation of OpenFoam to be found in the folder "tutorials".
The simulation example, which is described in more detail in the following steps, is structured as follows:

Name of the simulation example: cfdkurs
Storage location of the simulation example: ~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$

In the storage location of the simulation example, also known as the home directory, in the following all commands are executed in the terminal (shell). This means, for example, if the first calculation mesh has to be created with blockMesh, the cfdkurs folder has to be opened first, and then the command >blockMesh entered. The cfdkurs directory contains the folders 0, constant and system.

3.1 Mesh creation – General

In OpenFoam, the mesh can be created using the SnappyHexMesh function. The process of such a creation will be explained by means of a simple example [9], [10].

Step 1: For the first step, an *.stl file of the geometry is needed. In this file, created suing the CAD program, the surface of the geometry is described by means of triangles.

Step 2: A first background mesh is superimposed on this geometry using blockMesh. The blockMesh function can be used to create a cuboid, which is split into several cubic or cuboidal cells. In blockMeshDict, the text file used to execute blockMesh, the coordinates of the corner points of the cuboid and the number of cells in between are defined.

Step 3: The snappyHexMesh function now refines the cells surrounding the STL surface. How often these cells are to be refined is defined in snappyHexMeshDict.

Step 4: The cells either inside or outside the viewing space are then removed, and the remaining ones further refined.

Step 5: In a final step, the edges and surfaces are smoothed. In addition, additional layers can be introduced to refine the transition between the cells and the geometry surface and thus increase the resolution.

Steps 3 through 5 are all performed in one operation during the execution of snappyHexMesh and are precisely defined in the system folder in snappyHexMeshDict.

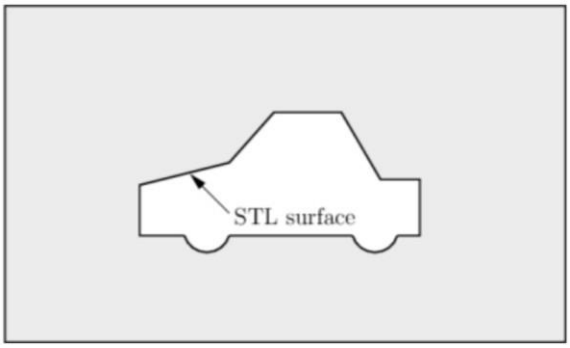


Figure 14: Step 1, existing *.stl file

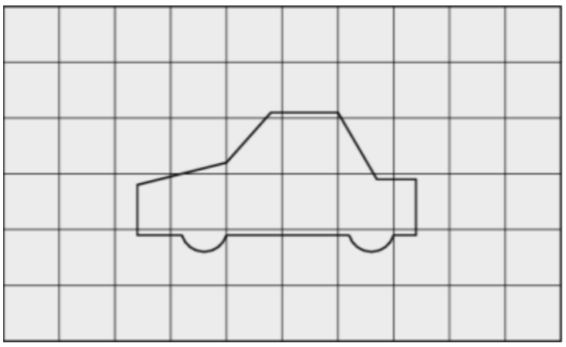


Figure 15: Step 2, creation of a blockMesh block (background mesh)

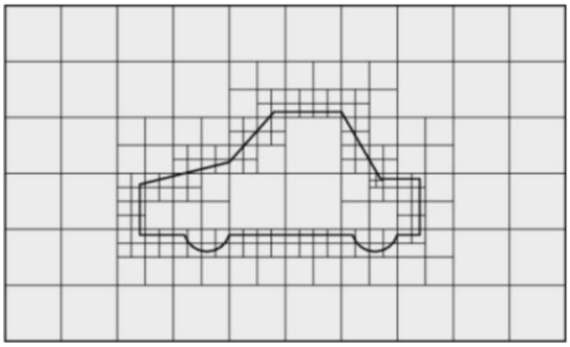


Figure 16: Step 3, refining of the mesh using the SnappyHexMesh function

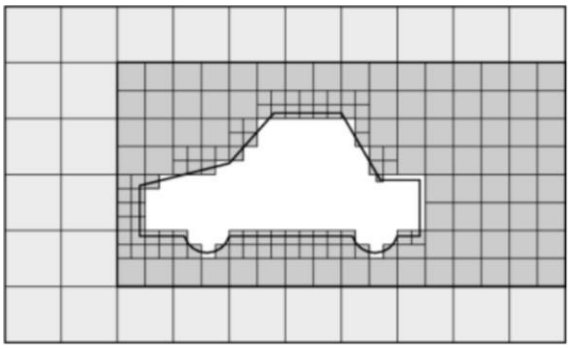


Figure 17: Step 4, removal of cells inside or outside the viewing space

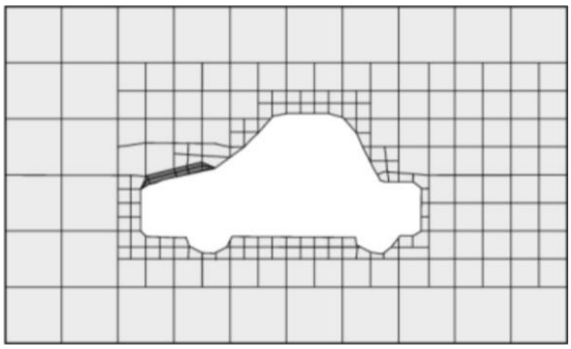


Figure 18: Step 5, smoothing and refining of the mesh

The following sections provide a more detailed description of the procedure described here.

4. First calculation mesh – blockMesh

Before the creation of a first calculation mesh is started, it must be ensured that the geometry has the correct scaling. When creating the *.stl file or when exporting, respectively, pay attention to the units with which the geometry is exported. OpenFoam works with SI units, therefore all geometric dimensions are in metres. If, for example, the geometry has been exported in millimetres, a transformation can be performed within OpenFoam. This can be done using the command >transformPoints.

```
>transformPoints -scale '(0.001 0.001 0.001)' -region fluid
>transformPoints -scale '(0.001 0.001 0.001)' -region solid
```

Using blockMesh, a background mesh is superimposed on the geometry as described above. To execute the command blockMesh, the corresponding text file blockMeshDict and the text file controlDict are needed. Both are saved in the "system" folder and are usually copied from other simulation templates and then adapted to the respective new case. With the following command, a text file can be copied into a new folder:

```
>cp Path-where-the-file-is-located Destination-folder/
```

Example:
~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs$ cp StandardVersuchBaseCaseVH/system/blockMeshDict system/

In the text file blockMeshDict, the coordinates of the corner points of the cuboid and the number of cells in between are defined. The following Figure 19 shows the blockMeshDict file.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
    (-0.1 -0.04  -0.05)         // 0
    ( 0.1 -0.04  -0.05)         // 1
    ( 0.1  0.04  -0.05)         // 2
    (-0.1  0.04  -0.05)         // 3
    (-0.1 -0.04   0.05)         // 4
    ( 0.1 -0.04   0.05)         // 5
    ( 0.1  0.04   0.05)         // 6
    (-0.1  0.04   0.05)         // 7
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (30 10 10) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
);

mergePatchPairs
(
);

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

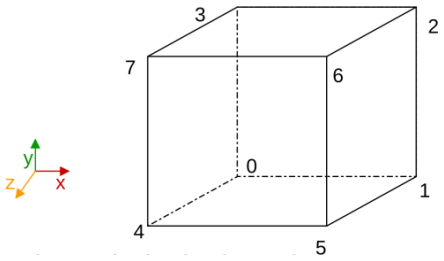


Figure 19: Text file blockMeshDict (left), description of the coordinate points (right) [11]

From line 19 to line 28, the vertices of the rectangle are given in X, Y, Z format. Line 33 contains the following information:

- hex (0 1 2 3 4 5 6 7) Definition of the hexahedral block. Pay attention to the order!
- (30 10 10) Number of cells between corner points
- simpleGrading (1 1 1) Scaling of the cells between the corner points

There are two ways to superimpose a suitable block on the geometry. Either the corner coordinates are known and entered in the blockMeshdict file, or the coordinates are unknown and must be determined. If unknown, they can be determined using Paraview. In doing so, the STL part geometry and the generated blockMesh block are visualised in Paraview, and the respective values are read. In order to display the blockMesh block, the following points must be fulfilled:

- blockMeshDict text file exists in the system folder
- controlDict text file exists in the system folder
- blockMesh executed in terminal
  - >blockMesh
- Dummy file foam.foam present in the case folder

To create a dummy file in the case folder, the following command must be executed:

>nano foam.foam

Example:
~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$ nano foam.foam

The text editor opens, and an empty text file appears.

Ctrl + O

saves the file, and

Ctrl + X

closes the editor again. Paraview can then be opened via the following command:
>paraview
The *.stl geometry data and the dummy file foam.foam must then be opened in Paraview:

File > Open > foam.foam

If the file is not displayed immediately, the opening must be confirmed with "Apply".

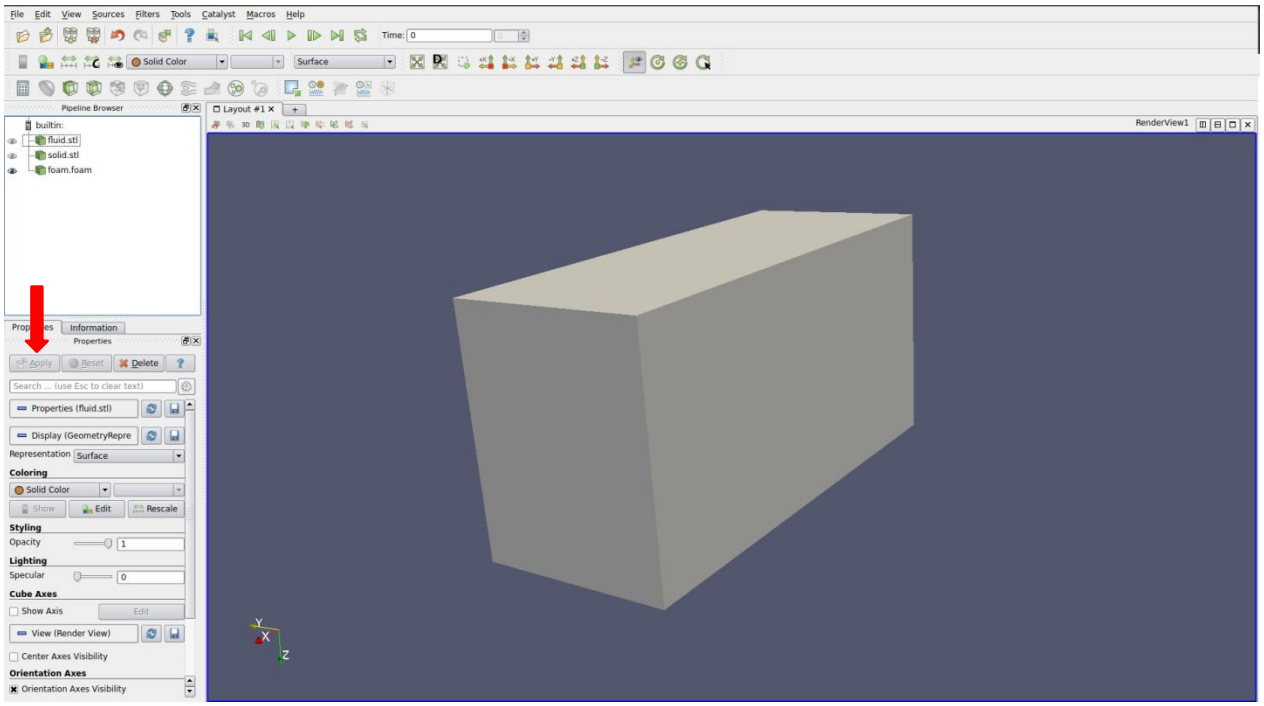


Figure 20: Created block in Paraview

The corner point coordinates of the selected geometry are displayed in the Information section in the lower left menu area. Figure 21 shows the corner point coordinates of the solid.stl geometry.

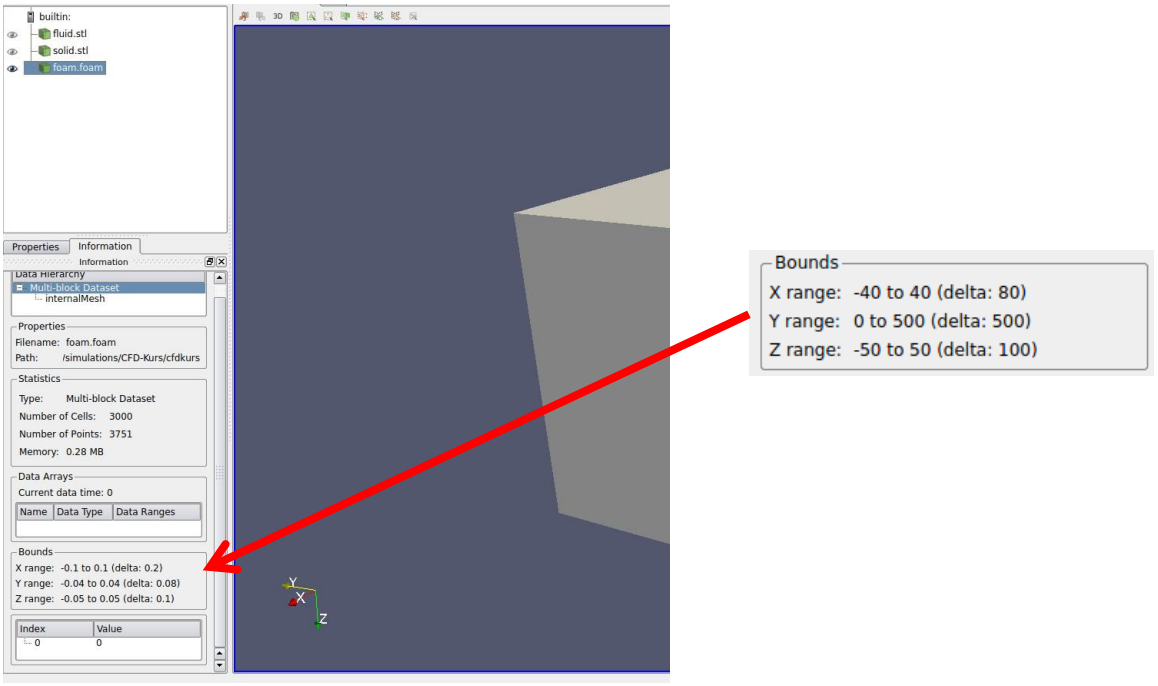


Figure 21: Corner point coordinates of the solid.stl geometry

Now the coordinates of the simulation geometry, which are read from Paraview, must be entered into the text file blockMeshDict. Here the blockMesh block should always be selected to be slightly larger than the simulation geometry. In the example shown, the blockMesh block is 5 units larger than the simulation geometry is. The block generated thus has the following dimensions:

```
19 vertices
20 (
21     (-45   -5  -55)          // 0
22     ( 45   -5  -55)          // 1
23     ( 45  505  -55)          // 2
24     (-45  505  -55)          // 3
25     (-45   -5   55)          // 4
26     ( 45   -5   55)          // 5
27     ( 45  505   55)          // 6
28     (-45  505   55)          // 7
29 );
```

Figure 22: Corner point coordinates of the generated BlockMesh block, entered in the blockMeshDict file

After the corner point coordinates have been entered into the blockMeshDict-File, the display in Paraview still has to be updated with "refresh". The previously defined block will then appear in the viewing area (Figure 23).

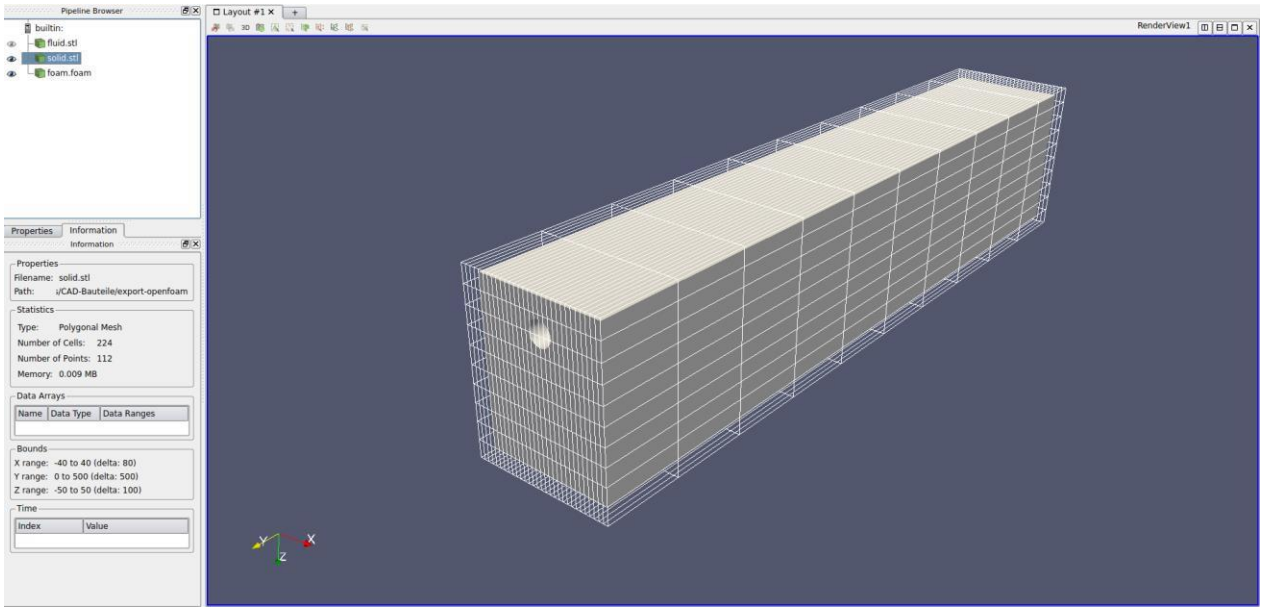


Figure 23: Simulation geometry with the BlockMesh block (mesh) superimposed

The shape of the rectangles should ideally be cubic. This can be adjusted by adjusting the splitting in the blockMeshDict text file. To this end, the text file blockMeshDict has to be opened, and the line 33 "(30 10 10) – Number of cells between the corner points" modified. By executing the blockMesh command again, the adjusted data are accepted:

>blockMes

h Example:
~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$ blockMesh

5. Second calculation mesh – snappyHexMesh

The snappyHexMesh program then refines the cells surrounding the STL geometry. In order to perform the refinement with snappyHexMesh, the following files must first be copied into the "system" folder:

- system
  - snappyHexMeshDict
  - meshQualityDict
  - surfaceFeatureExtractDict
  - fvSchemes
  - fvSolution

Furthermore, the *.stl files must be located in the folder "constant":

- constant
  - Folder "triSurface" (may have to be created)
    - solid.stl
    - fluid.stl

If the folder "triSurface" does not exist, it can be created without further ado. This can be created with the following command:

>mkdir triSurface

Example:
~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs/constant$ mkdir triSurface

Next, you can start to adapt the text files. The content of the text file „snappyHexMeshDict" is shown in the following figures:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Which of the steps to run
castellatedMesh true;
snap            true;
addLayers       false;


// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
{
    bottomAir.stl
    {
        type triSurfaceMesh;
        name bottomAir;
    }

    topAir.stl
    {
        type triSurfaceMesh;
        name topAir;
    }

    leftSolid.stl
    {
        type triSurfaceMesh;
        name leftSolid;
    }

    rightSolid.stl
    {
        type triSurfaceMesh;
        name rightSolid;
    }

    heater.stl
    {
        type triSurfaceMesh;
        name heater;
    }
};


// Settings for the castellatedMesh generation.
castellatedMeshControls
{

    // Refinement parameters
    // ~~~~~~~~~~~~~~~~~~~~~

    // If local number of cells is >= maxLocalCells on any processor
    // switches from from refinement followed by balancing
    // (current method) to (weighted) balancing before refinement.
    maxLocalCells 100000;

    // Overall cell limit (approximately). Refinement will stop immediately
    // upon reaching this number so a refinement level might not complete.
    // Note that this is the number of cells before removing the part which
    // is not 'visible' from the keepPoint. The final number of cells might
    // actually be a lot less.
    maxGlobalCells 2000000;

    // The surface refinement loop might spend lots of iterations
    // refining just a few cells. This setting will cause refinement
    // to stop if <= minimumRefine are selected for refinement. Note:
    // it will at least do one iteration (unless the number of cells
    // to refine is 0)
    minRefinementCells 10;

    // Number of buffer layers between different levels.
    // 1 means normal 2:1 refinement restriction, larger means slower
    // refinement.
    nCellsBetweenLevels 2;



    // Explicit feature edge refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies a level for any cell intersected by its edges.
    // This is a featureEdgeMesh, read from constant/triSurface for now.
    features
    (
        {
            file "bottomAir.eMesh";
            level 1;
        }
        {
            file "heater.eMesh";
            level 1;
        }
        {
            file "leftSolid.eMesh";
            level 1;
        }
        {
            file "rightSolid.eMesh";
            level 1;
        }
        {
            file "topAir.eMesh";
            level 1;
        }
    );



    // Surface based refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies two levels for every surface. The first is the minimum level
    // every cell intersecting a surface gets refined up to the minimum level
    // The second level is the maximum level. Cells that 'see' multiple
    // intersections where the intersections make an
    // angle > resolveFeatureAngle get refined up to the maximum level.

    refinementSurfaces
    {
        solid
        {
            // Surface-wise min and max refinement level
            level (1 1);

            faceZone solid;
            cellZone solid;
            cellZoneInside insidePoint;
            insidePoint (X Y Z) // irgendwelche Koordinaten im solid

        }

         fluid
        {
            // Surface-wise min and max refinement level
            level (1 1);

            faceZone fluid;
            cellZone fluid;
            cellZoneInside insidePoint;
            insidePoint (X Y Z) // irgendwelche Koordinaten im fluid

        }

    }

    // Resolve sharp angles
    resolveFeatureAngle 30;


    // Region-wise refinement
    // ~~~~~~~~~~~~~~~~~~~~~~

    // Specifies refinement level for cells in relation to a surface. One of
    // three modes
    // - distance. 'levels' specifies per distance to the surface the
    //   wanted refinement level. The distances need to be specified in
    //   descending order.
    // - inside. 'levels' is only one entry and only the level is used. All
    //   cells inside the surface get refined up to the level. The surface
    //   needs to be closed for this to be possible.
    // - outside. Same but cells outside.

    refinementRegions
    {
        //refinementBox
        //{
        //    mode inside;
        //    levels ((1E15 4));
        //}
    }


    // Mesh selection
    // ~~~~~~~~~~~~~~

    // After refinement patches get added for all refinementSurfaces and
    // all cells intersecting the surfaces get put into these patches. The
    // section reachable from the locationInMesh is kept.
    // NOTE: This point should never be on a face, always inside a cell, even
    // after refinement.
    locationInMesh (0.01 0.01 0.01); // irgendeinen Punkt im Mesh angeben


    // Whether any faceZones (as specified in the refinementSurfaces)
    // are only on the boundary of corresponding cellZones or also allow
    // free-standing zone faces. Not used if there are no faceZones.
    allowFreeStandingZoneFaces false;
}



// Settings for the snapping.
snapControls
{
    //- Number of patch smoothing iterations before finding correspondence
    //  to surface
    nSmoothPatch 3;

    //- Relative distance for points to be attracted by surface feature point
    //  or edge. True distance is this factor times local
    //  maximum edge length.
    tolerance 1.0;

    //- Number of mesh displacement relaxation iterations.
    nSolveIter 30;

    //- Maximum number of snapping relaxation iterations. Should stop
    //  before upon reaching a correct mesh.
    nRelaxIter 5;

    //- Highly experimental and wip: number of feature edge snapping
    //  iterations. Leave out altogether to disable.
```

Figure 24: Content snappyHexMeshDict file, line 1-232

```
233     //  Of limited use in this case since faceZone faces not handled.
234     nFeatureSnapIter 10;
235 }
236
237
238
239 // Settings for the layer addition.
240 addLayersControls
241 {
242     relativeSizes true;
243
244     // Per final patch (so not geometry!) the layer information
245     layers
246     {
247         maxY
248         {
249             nSurfaceLayers 3;
250         }
251     }
252
253     // Expansion factor for layer mesh
254     expansionRatio 1.3;
255
256     // Wanted thickness of final added cell layer. If multiple layers
257     // is the thickness of the layer furthest away from the wall.
258     // Relative to undistorted size of cell outside layer.
259     // See relativeSizes parameter.
260     finalLayerThickness 1;
261
262     // Minimum thickness of cell layer. If for any reason layer
263     // cannot be above minThickness do not add layer.
264     // Relative to undistorted size of cell outside layer.
265     minThickness 0.1;
266
267     // If points get not extruded do nGrow layers of connected faces that are
268     // also not grown. This helps convergence of the layer addition process
269     // close to features.
270     // Note: changed(corrected) w.r.t 17x! (didn't do anything in 17x)
271     nGrow 0;
272
273     // Advanced settings
274
275     // When not to extrude surface. 0 is flat surface, 90 is when two faces
276     // are perpendicular
277     featureAngle 30;
278
279     // Maximum number of snapping relaxation iterations. Should stop
280     // before upon reaching a correct mesh.
281     nRelaxIter 3;
282
283     // Number of smoothing iterations of surface normals
284     nSmoothSurfaceNormals 1;
285
286     // Number of smoothing iterations of interior mesh movement direction
287     nSmoothNormals 3;
288
289     // Smooth layer thickness over surface patches
290     nSmoothThickness 2;
291
292     // Stop layer growth on highly warped cells
293     maxFaceThicknessRatio 0.5;
294
295     // Reduce layer growth where ratio thickness to medial
296     // distance is large
297     maxThicknessToMedialRatio 1;
298
299     // Angle used to pick up medial axis points
300     // Note: changed(corrected) w.r.t 17x! 90 degrees corresponds to 130 in 17x.
301     minMedianAxisAngle 90;
302
303     // Create buffer region for new layer terminations
304     nBufferCellsNoExtrude 0;
305
306     // Overall max number of layer addition iterations. The mesher will exit
307     // if it reaches this number of iterations; possibly with an illegal
308     // mesh.
309     nLayerIter 50;
310 }
311
312
313
314 // Generic mesh quality settings. At any undoable phase these determine
315 // where to undo.
316 meshQualityControls
317 {
318     #include "meshQualityDict"
319
320     // Advanced
321
322     //- Number of error distribution iterations
323     nSmoothScale 4;
324     //- Amount to scale back displacement at error points
325     errorReduction 0.75;
326 }
327
328
329 // Advanced
330
331 // Merge tolerance. Is fraction of overall bounding box of initial mesh.
332 // Note: the write tolerance needs to be higher than this.
333 mergeTolerance 1e-6;
334
335
336 // ************************************************************************* //
```

Figure 25: Content snappyHexMeshDict file, line 233-336

Here, the geometry files must first be adapted. Here, under geometry (line 30)the STL geometries used must be specified. In the example, this comprises the following *.stl files:

- solid.stl
- fluid.stl

The remaining entries under geometry can be deleted. In addition, the entries for "features" (line 103) can be changed in analogy to the geometry change.

```
geometry
{
    solid.stl
    {
        type triSurfaceMesh;
        name solid;
    }

    fluid.stl
    {
        type triSurfaceMesh;
        name fluid;
    }

};
```

```
features
(
    {
        file "solid.eMesh";
        level 1;
    }
    {
        file "fluid.eMesh";
        level 1;
    }
);
```

The section "refinementSurfaces" (line 138) must also be adapted. The names must again be changed to "solid" and "fluid". In the entries "faceZone" and "cellZone", the names of the *.stl files have to be entered. In addition, coordinates of a point located inside the geometry to be meshed must be specified. This must be entered for "solid" and

"fluid". Line 201 comprises the entry "locationInMesh", where again a point inside the geometry to be meshed has to be specified. Specification of a point inside a geometry serves to define the start of meshing, or whether the inner or outer geometry is to be meshed, respectively.

Figure 26 below shows the contents of the meshQualityDict file. This is also required for the simulation, but no changes are necessary.

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  3.0.1                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     object      meshQualityDict;
14 }
15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17 // Include defaults parameters from master dictionary
18 #include "$WM_PROJECT_DIR/etc/caseDicts/meshQualityDict"
19
20 //#includeEtc "caseDicts/meshQualityDict"
21
22
23 // ************************************************************************* //
```

Figure 26: Content of the meshQualityDict file

In the text file "surfaceFeatureExtractDict" (Figure 27) again the names of the *.stl files have to be entered (lines 17 and 33).

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  3.0.1                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     object      surfaceFeatureExtractDict;
14 }
15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17 solid.stl
18 {
19     extractionMethod    extractFromSurface;
20
21     extractFromSurfaceCoeffs
22     {
23         // Mark edges whose adjacent surface normals are at an angle less
24         // than includedAngle as features
25         // - 0  : selects no edges
26         // - 180: selects all edges
27         includedAngle   150;
28     }
29
30     // Write options
31     writeFeatureEdgeMesh    yes;
32 }
33 fluid.stl
34 {
35     extractionMethod    extractFromSurface;
36
37     extractFromSurfaceCoeffs
38     {
39         // Mark edges whose adjacent surface normals are at an angle less
40         // than includedAngle as features
41         // - 0  : selects no edges
42         // - 180: selects all edges
43         includedAngle   150;
44     }
45
46     // Write options
47     writeFeatureEdgeMesh    yes;
48 }
49 // ************************************************************************* //
```

Figure 27: Contents surfaceFeatureExtractDict file

After all text files have been adjusted, the command

>surfaceFeatureExtrac

t

has to be executed.

Example:

~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$

surfaceFeatureExtract

This will generate the files

- solid.eMesh
- fluid.eMesh

in the folder "triSurface". Then the command

>snappyHexMesh -overwrite

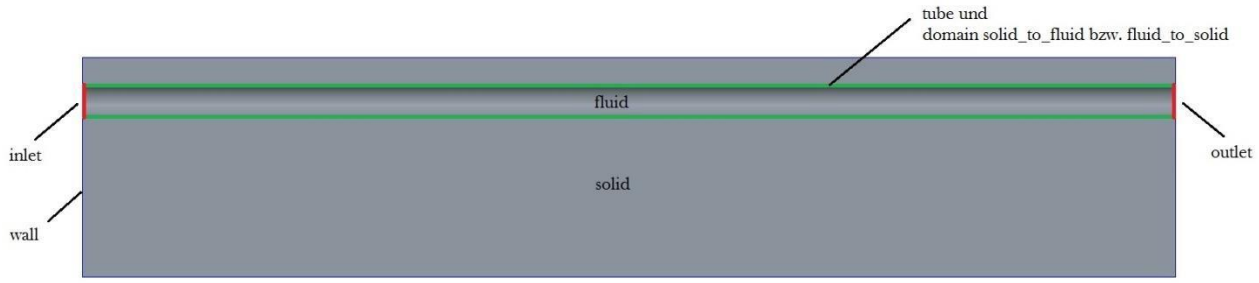
can be executed.

Example:
~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$ snappyHexMesh -overwrite

The command "-overwrite" is used, since otherwise a new folder will be created if a mesh already exists.

6. Creation and allocation of the regions

Since in this example the heat transfer from a liquid to a solid substance is simulated, the individual geometries must be classified. This serves to inform the program as to which geometry or volume is a solid or a liquid, and to be able to assign adapted material parameters to it afterwards.



With Blender, in a previous step the geometries were exported individually as *.stl files. Here, the geometry solid.stl consists of wall.stl and tube.stl. The fluid.stl geometry consists of tube.stl, inlet.stl and outlet.stl. This means that a total of 7 individual *.stl files must exist. Depending on the simulation set-up, the geometry "tube.stl" may not be required. Figure 28 illustrates the individual STL geometries.

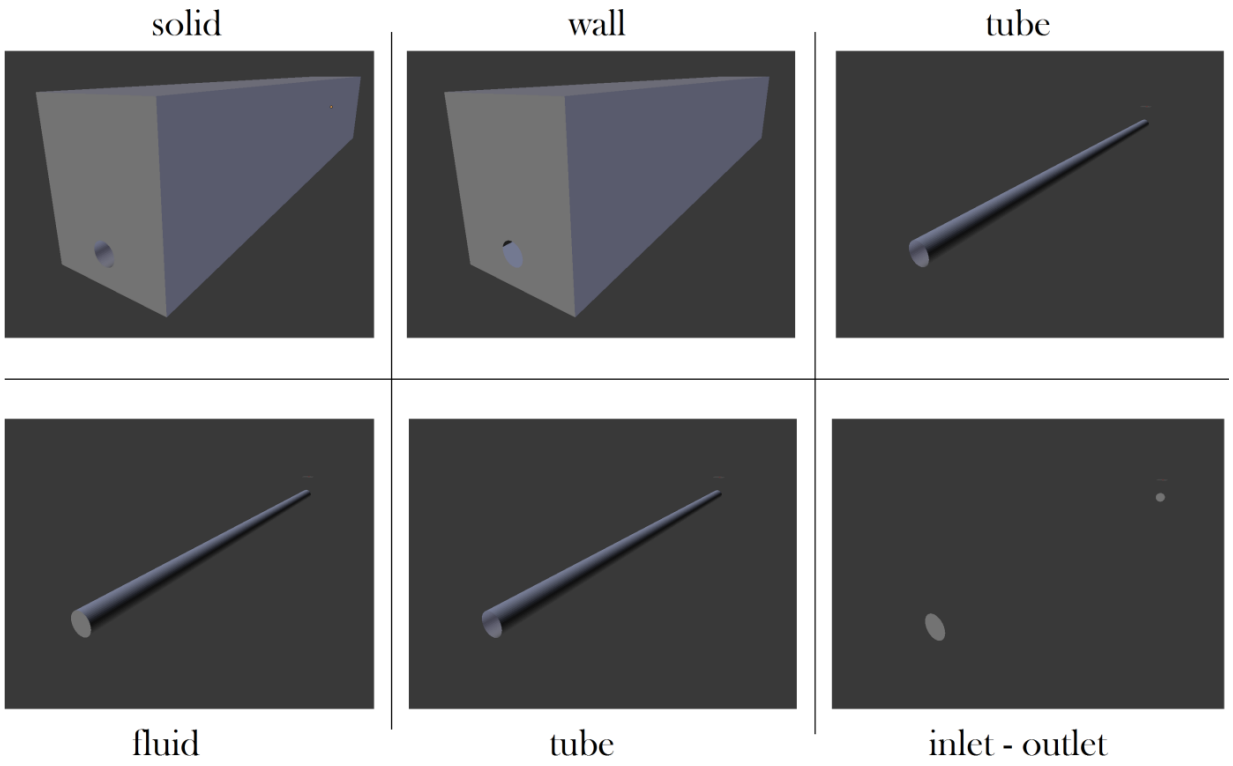


Figure 28: Illustration of the individual *.stl files

The individual *.stl files must be assigned to certain regions. A list of the *.stl files with the

corresponding regions is shown in Table 2.

Table 2: Splitting of the *.stl files into regions

*.stl file	Region
tube.stl	fluid
inlet.stl	fluid
outlet.stl	fluid
wall.stl	solid

Before these are assigned, the zones (Fluid and Solid) must first be split. This is done with the following command:

>splitMeshRegions -cellZones -overwrite

Example:
~/OpenFoam/OpenFoam-3.0.1/simulations/CFD-Kurs/cfdkurs$ splitMeshRegions -cellZones - overwrite

The individual *.stl files or surfaces, respectively, must then be assigned to the regions. This is done with the command:

>surfaceToPatch „Location of the *.stl file " -region „fluid or solid " Example:
~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$ surfaceToPatch constant/triSurface/inlet.stl -region fluid

~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$ surfaceToPatch constant/triSurface/outlet.stl -region fluid

~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$ surfaceToPatch constant/triSurface/tube.stl -region fluid

~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$ surfaceToPatch constant/triSurface/wall.stl -region solid

Here, the following error message may appear (Figure 29):

-->FOAM FATAL ERROR:
Wrong number of arguments, expected 1
found 2 Invalid options: -region

```
ak118456@ak118456-HP-EliteBook-8570p:~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$ surfaceToPatch constant/triSurface/inlet.stl -region fluid

Usage: surfaceToPatch [OPTIONS] <surfaceFile>
options:
  -case <dir>       specify alternate case directory, default is the cwd
  -faceSet <name>   only repatch the faces in specified faceSet
  -noFunctionObjects
                    do not execute functionObjects
  -tol <scalar>     search tolerance as fraction of mesh size (default 1e-3)
  -srcDoc           display source code in browser
  -doc              display application documentation in browser
  -help             print the usage

reads surface and applies surface regioning to a mesh

Using: OpenFOAM-3.0.1 (see www.OpenFOAM.org)
Build: 3.0.1-bc1922f074df


--> FOAM FATAL ERROR:
Wrong number of arguments, expected 1 found 2
Invalid option: -region


FOAM exiting
```

Figure 29: Error message when entering the command "surfaceToPatch constant/triSurface/inlet.stl -region fluid"

In this case, an additional step must be performed, or the regions must be assigned separately. I.e., the surfaceToPatch command must be executed in a separately created folder. This allows assigning (patching) the regions separately from each other. For this reason, dummy folder structures are created (fluid > constant > polyMesh), since otherwise OpenFoam would not recognise the folder structure. If only one region existed, i.e. solid or fluid alone, the surfaceToPatch command could be executed directly in the main folder (in the example "cfdkurs").
However, since there are two regions in the example, the following step must be performed:

1. First you have to create the folders "solid" and "fluid" in the main folder.
2. A constant folder must then be created in each of these folders. A folder is created with the command >mkdir folder name.
3. The polyMesh folder of the main folder (~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs/constant) must then be copied into this constant folder.
4. Furthermore, the folder system must be created in the respective folder (solid and fluid), and the file controlDict must be copied into this folder. This can be found in the main folder (cfdkurs) in the system folder.
5. Finally the corresponding *.stl files have to be copied into the created folders fluid and solid.

After creating and copying the files, the folder structure must look as follows:

cfdkurs

- **constant**
  - polyMesh
  - fluid
    - polyMesh
  - solid
    - polyMesh

- **solid**
  - constant
    - polyMesh
  - system
    - controlDict
  - wall.stl
- **fluid**
  - constant
    - polyMesh
  - system
    - controlDict
  - inlet.stl
  - outlet.stl

The directory constant > solid may only contain the *.stl files of the outer walls. This means, for example, that the wall of an inner pipe must not be included. After the folder structure has been created, navigate to the created folder (solid or fluid) and execute the surfaceToPatch command for each contained *.stl file:

>surfaceToPatch CAD_file.stl

Example:
~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs/solid$ surfaceToPatch wall.stl

~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs/fluid$ surfaceToPatch inlet.stl

etc.

Between running this command for each individual *.stl file, the patch0 entry in the polyMesh-boundary file must be changed. The individual procedures are described step by step below:
Example for Fluid

cfdkurs > fluid > 0.1

- Open first time step folder (0.1) (not 0)
- Open polyMesh folder
- Open boundary file
- Rename patch0 to inlet
  (or to the name used in sufaceToPatch, respectively)
- Run > surfaceToPatch outlet.stl in the cfdkurs folder > fluid
- Open cfdkurs > fluid > 0.2
- Copy polyMesh folder and paste it into
  cfdkurs > constant > fluid
- Then the entire cfdkurs > fluid folder can be saved as a backup or deleted

After that, the following entries in the boundary file have to be deleted, and the number in row 18 (Figure 30) has to be changed from 4 to 3, because the number of patches has changed. If this is not changed, an error message will appear when the results are later displayed in Paraview. The following command (e.g. for the region solid) can be used to check the correctness of the generated mesh:

> checkMesh -region solid

Example:
~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$ checkMesh -region solid

```
1  /*---------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  3.0.1                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       polyBoundaryMesh;
13     location    "0.02/polyMesh";
14     object      boundary;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18 4
19 (
20     fluid_to_solid
21     {
22         type            mappedWall;
23         inGroups
24 2
25 (
26 wall
27 mappedPatch
28 )
29 ;
30         nFaces          4800;
31         startFace       16768;
32         sampleMode      nearestPatchFace;
33         sampleRegion    solid;
34         samplePatch     solid_to_fluid;
35     }
36     fluid_to_domain0
37     {
38         type            mappedWall;
39         inGroups
40 2
41 (
42 wall
43 mappedPatch
44 )
45 ;
46         nFaces          0;
47         startFace       21568;
48         sampleMode      nearestPatchFace;
49         sampleRegion    domain0;
50         samplePatch     domain0_to_fluid;
51     }
52     inlet
53     {
54         type            patch;
55         nFaces          32;
56         startFace       21568;
57     }
58     outlet
```

Figure 30: boundary file of the fluid

In order to be able to assign their properties and initial conditions to the regions, a regionProperties file is required in constant. The following entries must exist in it:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      regionProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

regions
(
    fluid       (fluid)
    solid       (solid)
);


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

Figure 31: regionProperties file in folder constant

Once these steps have been carried out, the initial and boundary conditions can be entered.

7. Solving – Simulation with chtMultiRegion

In the selected simulation case, the heating and cooling process is considered. During the heating process, water at a temperature of 90 C flows through the pipe for 10 seconds. This is followed by the cooling process, during which water at a temperature of 30 °C flows through the pipe for 10 seconds. In addition, the simulations are carried out with two different Reynolds numbers or flow velocities, respectively, in order to be able to observe the heat transfer in a laminar and turbulent flow.

7.1. Theory of flow simulation

The turbulence model standard k-epsilon was used for the simulation of the flow in the channel. This is a two-equation model and is the most commonly used model to describe flows in CFD simulations. In the transport equation, the production and diffusion for turbulent kinetic energy k are approximated.

$$P_k = -\left(2\nu_T \bar{S}_{ij}\right)\bar{S}_{ij} = -2\nu_T \left(\bar{S}_{ij}\right)^2 \tag{1}$$

$$D_k = \frac{\partial}{\partial x_j}\left(\nu \frac{\partial k}{\partial x_j} + \frac{\nu_T}{\sigma_k}\frac{\partial k}{\partial x_j}\right) = \left[\left(\nu + \frac{\nu_T}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] \tag{2}$$

Based on several assumptions (e.g. pressure-velocity correlation) and approximations (e.g. Boussinesq approach), the equations of the standard kε model are finally as follows [12]:

$$\frac{\partial}{\partial x_j}(\bar{u}_j k) = -\nu_T(\bar{S}_{ij})^2 - \varepsilon + \left[\left(\nu + \frac{\nu_T}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] \quad (3)$$

$$\frac{\partial}{\partial x_j}(\bar{u}_j \varepsilon) = \left\{c_{\varepsilon 1}\frac{\varepsilon}{k}\right\}\left[-\nu_T(\bar{S}_{ij})^2\right] - \left\{c_{\varepsilon 2}\frac{\varepsilon}{k}\right\}\varepsilon + \left[\left(\nu + \frac{\nu_T}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] \quad (4)$$

P_k	Production
D_k	Diffusion
ν_T	Vortex viscosity
$\bar{S}_{ij}$	viscous stress tensor
x_j	Place coordinate
$\bar{u}$	average speed
k	turbulent kinetic energy
ε	isotropic dissipation rate
C_μ	model constant
$c_{\varepsilon 1}$	model constant
$c_{\varepsilon 2}$	model constant
σ_k	model constant
σ_ε	model constant

Typically, the model constants are defined as follows: C_μ = 0.09; $c_{\varepsilon 1}$ = 1.44; $c_{\varepsilon 2}$ = 1.92; σ_k = 1.0; and σ_ε = 1.3.

To be able to calculate yPlus, turbulent initial values are required, which are calculated s follows [13]:

$$k = \frac{3}{2}(I * |u_{ref}|)^2$$

assumed to be 0.8. u_{ref} is the velocity,

$$\varepsilon = \frac{C_\mu{}^{0,75} * k^{1,5}}{L}$$

ε is the turbulent dissipation rate, C_μ a constant with the value 0.09, and L the length of the channel.

$$\nu_t = C_\mu \frac{k^2}{\varepsilon}$$

ν_t is the turbulent viscosity.

The following values result from the illustrated formulas for the simulation example:

k =0.273
ε =1.563
ν = 0.00324

7.2. Input parameters

To complete the simulation, the input parameters for laminar and turbulent simulation are listed below. For the inlet speed (initial condition), the following formula was used in both cases:

$$Re = \frac{u\ D}{\nu} \tag{5}$$

Here, Re is the Reynolds number in [1], u the velocity in [m/s], D the diameter of the channel in [m], and ν (nu) the kinematic viscosity in [m²/s]. The kinematic viscosity of water at different temperatures was taken from the following table:

Table 3: Substance parameters of water at different temperatures [14]

ϑ °C	ρ kg m^{-3}	c_p kJ kg^{-1} K^{-1}	β 10^{-3} K^{-1}	η 10^{-6} kg m^{-1} s^{-1}	ν 10^{-6} m^2 s^{-1}	σ 10^{-3} N m^{-1}
-30	983,78	4,817	-1,4497	8661,1	8,804	-
-20	993,62	4,418	-0,6576	4362,7	4,931	-
-10	998,14	4,277	-0,2887	2645,2	2,650	77,10
-5	999,27	4,242	-0,1665	2153,5	2,155	76,40
0	999,84	4,218	-0,0672	1792,3	1,793	75,62
5	999,97	4,203	0,0162	1518,7	1,519	74,90
10	999,70	4,192	0,0879	1306,4	1,307	74,20
15	999,10	4,185	0,1507	1138,0	1,139	73,48
20	998,21	4,181	0,2067	1002,0	1,004	72,75
25	997,05	4,179	0,2572	890,45	0,893	71,96
30	995,65	4,177	0,3034	797,68	0,801	71,15
35	994,03	4,177	0,3459	719,62	0,724	70,35
40	993,22	4,177	0,3855	653,25	0,658	69,55
45	990,21	4,178	0,4226	596,32	0,602	-
50	988,04	4,180	0,4578	547,08	0,554	67,90
55	985,69	4,182	0,4912	504,19	0,512	-
60	983,20	4,184	0,5232	466,59	0,475	66,17
65	980,55	4,187	0,5541	433,44	0,442	-
70	977,77	4,190	0,5840	404,06	0,413	64,41
75	974,84	4,193	0,613	377,9	0,388	-
80	971,79	4,197	0,6414	354,49	0,365	62,60
85	968,61	4,201	0,6693	333,48	0,344	-
90	965,31	4,206	0,6967	314,53	0,326	60,74
95	961,89	4,211	0,7238	297,4	0,309	-
99,63[1]	958,61	4,216	0,7487	282,95	0,295	58,84

The pressure of the water flowing through is 5 bar ($5 \cdot 10^5$ Pa). At the beginning of the simulation, the metal block through which the coolant flows has a constant temperature of 30 °C.

7.2.1. Input parameters Laminar

Reynolds number	Re =	500
Channel diameter	D =	0.015 m
Kinematic viscosity, water	ν =	0.801 * 10^{-6} m²/s at 30 °C 0.326 * 10^{-6} m²/s at 90 °C

This results in the following velocities:

Flow velocity	u =	0.0267 m/s at 30 °C 0.011 m/s at 90 °C

7.2.2. Input parameters Turbulent

Reynolds number	Re =	10.000
Channel diameter	D =	0.015 m
Kinematic viscosity, water	ν =	0.801 * 10^{-6} m²/s at 30 °C 0.326 * 10^{-6} m²/s at 90 °C

This results in the following velocities:

Flow velocity	u =	0.533 m/s at 30 °C 0.217 m/s at 90 °C

These input parameters or initial conditions are defined in documents located in the folder

0>fluid. This contains the files alphat, epsilon, k, nut, p, p_rgh, T and U. The files p and T are present in the folder 0>solid.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 5e5;//1e5;

boundaryField
{
    inlet
    {
        type            zeroGradient;
        value           uniform 0;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 5e5;
    }
    fluid_to_solid
    {
        type            zeroGradient;//calculated;
        value           uniform 0;
    }

}

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

$$\frac{kg}{m\,s^2}$$

	Quantity	SI Base Units						
		kg	m	s	K	mol	A	cd
1.	Density	1	-3	0	0	0	0	0
2.	Linear Momentum	1	-2	-1	0	0	0	0
3.	Pressure	1	-1	-2	0	0	0	0
4.	Force	1	1	-2	0	0	0	0
5.	Lame's Coefficients	1	-1	-2	0	0	0	0
6.	Bulk Modulus	1	-1	-2	0	0	0	0
7.	Young's Modulus of Elasticity	1	-1	-2	0	0	0	0
8.	Specific Heat Capacity	0	2	-2	-1	0	0	0
9.	Thermal Conductivity	1	1	-3	-1	0	0	0

To be able to use the k-ε model, the yPlus value, which provides a statement about the cell height nearest to the wall, must have a value between 20 and 100. The value is checked with the following command:

> yPlus -region fluid

In order for this command to be executed, the following folder structure and the listed files must be included:

0

- fluid
  - alphat
  - epsilon
  - k
  - nut
  - p
  - p_rgh

    - T
    - U
- solid
    - p
    - T

The screenshots of the text files can be found in the Annex in sections 10.1. to 10.2.
In the fluid files, the names of the geometries must be adapted for epsilon, k, nut, p, p_rgh, T, U in the entry boundaryFields. In the simulation example, inlet, outlet and fluid_to_solid thus need to be entered. For the solid files p and T, wall and solid_to_fluid must be entered. In addition, the initial conditions must be adapted, including speed [m/s], pressure [Pa], and temperature [K].

The folder constant must contain the following folders and files:

constant
- fluid
    - polyMesh (folder)
    - fvSchemes
    - fvSolution
    - g
    - thermophysicalProperties
    - transportProperties
    - turbulenceProperties
- solid
    - polyMesh (folder)
    - fvSchemes
    - fvSolution
    - thermophysicalProperties

The screenshots of the text files can be found in the Annex in sections 10.3. to 10.4.

Once all adjustments have been made, the yPlus command can finally be executed for solid and fluid:

```
>yPlus -region fluid
>yPlus -region solid
```

Since the yPlus value in the example shown was over 100, the cells at the fluid-solid boundary were refined with the following command:

```
>refineWallLayer '(fluid_to_solid)' 0.5
```

(no -region option and must therefore be executed in the folder where inlet and outlet

were patched)

After refining, the generated polyMesh folder must be copied to constant > fluid.
The refineWallLayer command inserted additional cell rows, reducing yPlus to an average value of 54.74426. Figure 32 shows the refined mesh:

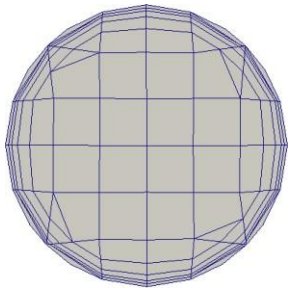


Figure 32: Refined meshing at the cooling tube geometry

A value of 30 °C is defined as the initial temperature for the solid. In addition, the boundary condition on the outer wall should be defined by changing the entry „externalHeatFluxTemperature “. It is specified that a constant heat transfer coefficient of 10 W/m²K is used for the heat transfer between the solid and the environment. This means that the environment or air, respectively, is not mapped in the simulation, but only taken into account with the constant boundary condition mentioned.

8. Running the simulation

In order to be able to start the simulation, the general conditions must be defined. These can be found in the constant folder. The following entries should be defined:

- Material data for fluid and solid are specified in thermophysicalProperties; these files are located in the constant folder. This includes values such as thermal conductivity or capacity, which are commonly used for steel and water.

- For the turbulent simulation, a file called turbulenceProperties is required in which the turbulence model to be used is defined.

- The numerical settings found in fvSchemes and fvSolution were used by default.

- In controlDict it was specified that the first part of the simulation (=heating process) takes 10 seconds and is stored every 0.1s.

- In order to be able to simulate a heating and cooling process taking a total of 20 seconds, the inlet temperature and velocity in the files T and U in folder 10 (simulation time switch folder) were adjusted after a 10-second heating phase in order to be able to finally also display the cooling phase.

The screenshots of the text files can be found in Annex 10. Finally, with

```
>chtMultiRegionFoam
```

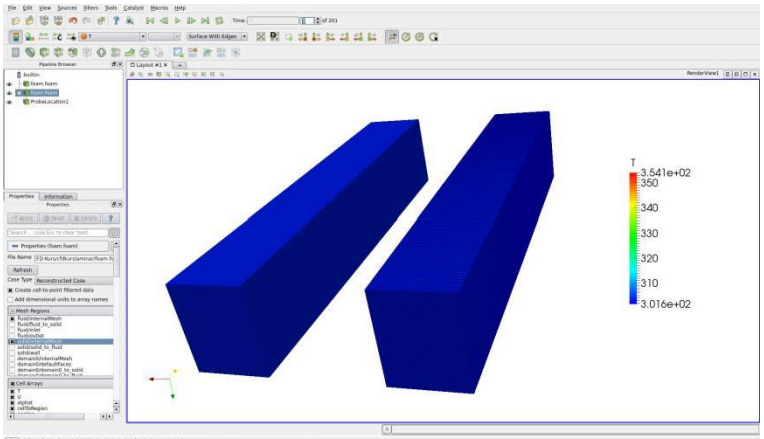
the simulation is started. After the turbulent simulation has been completed, the laminar simulation can also be created. For this purpose, a copy of the entire turbulent case (cfdkurs) is created (cfdkurslaminar). In order to perform a laminar simulation with this copied case, the following changes must be made:

- The speed in the folder 0 > U is reduced to map a laminar flow.
- In the turbulenceProperties file in the constant folder, the flow type is changed to laminar.

9. Results

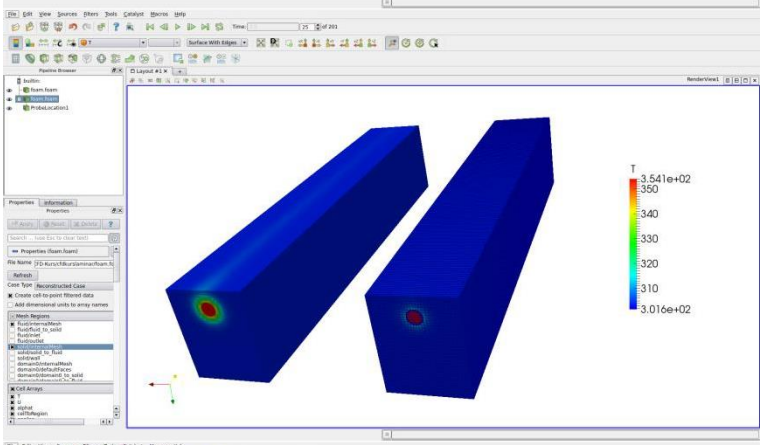
9.1. Heating and cooling process

After the simulation, the heating and cooling process was imaged in Paraview. The following figures show the results graphically in steps of 2.5 seconds. The turbulent simulation case is shown on the left and the laminar simulation case on the right.



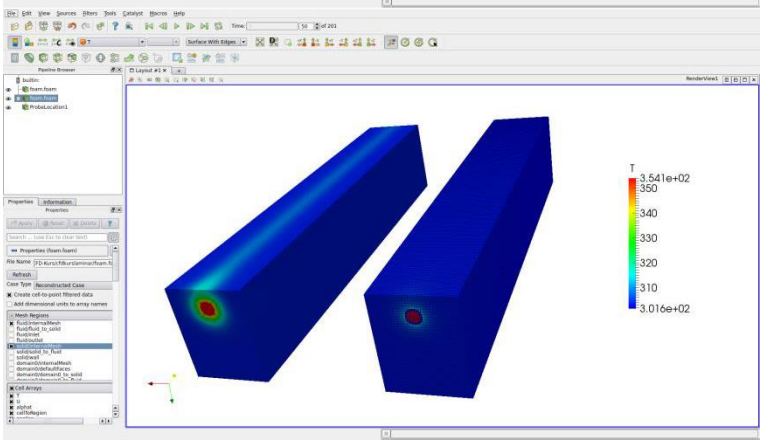
Turbulent (left)
Laminar (right)

Initial state
Constant 30 °C
t = 0 seconds



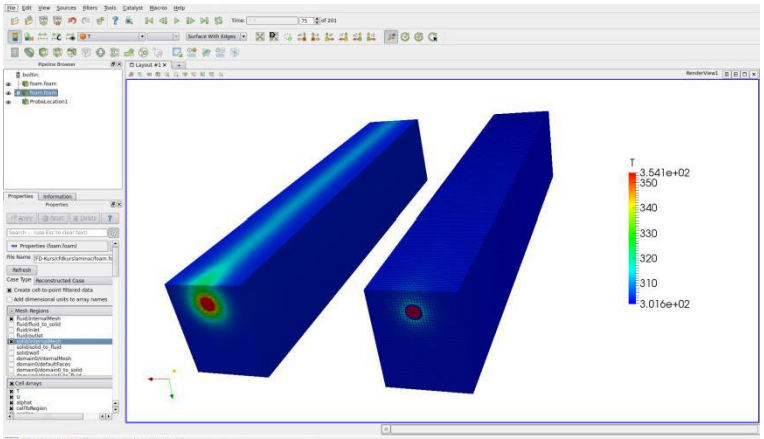
Turbulent (left)
Laminar (right)

Heating
process
t = 2.5 seconds



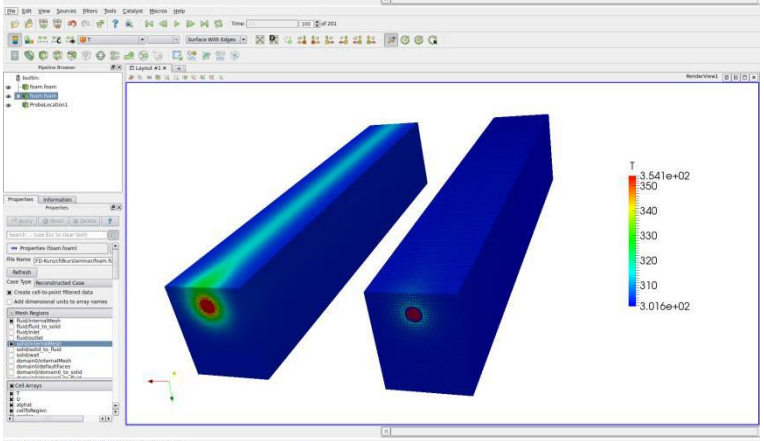
Turbulent (left)
Laminar (right)

Heating
process
t = 5 seconds



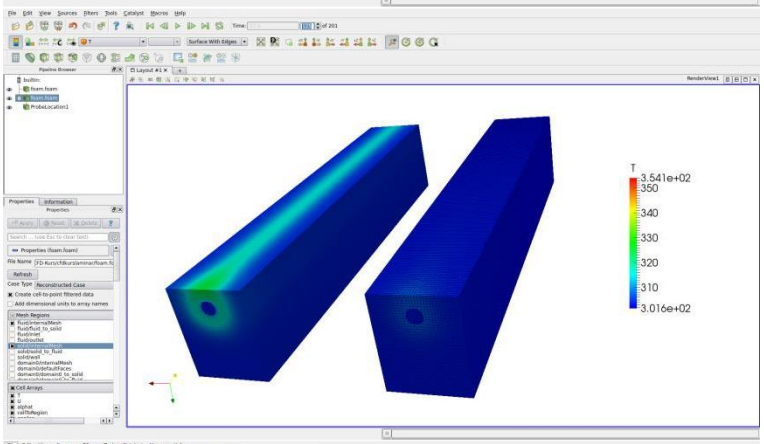
Turbulent (left)
Laminar (right)

Heating process
t = 7.5 seconds



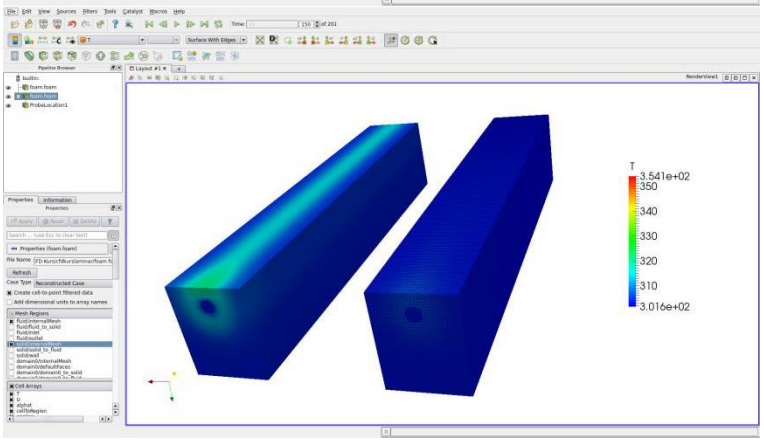
Turbulent (left)
Laminar (right)

End of heating-up process and start of cooling-down process
t = 10 seconds



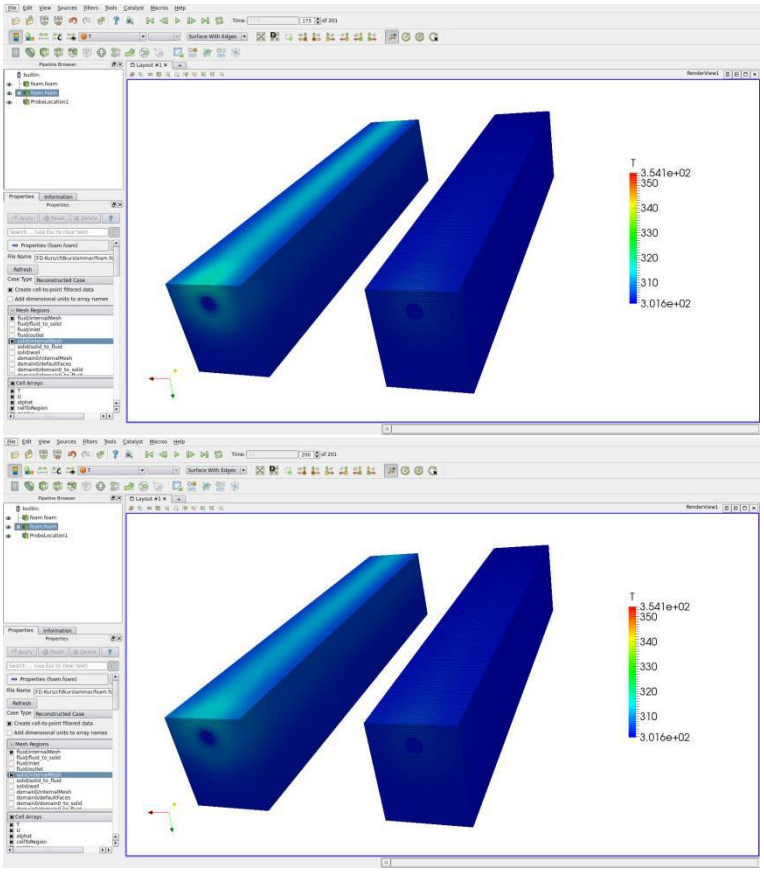
Turbulent (left)
Laminar (right)

Cooling process
t = 12.5 seconds



Turbulent (left)
Laminar (right)

Cooling process t = 15 seconds



Turbulent (left)
Laminar (right)

Cooling process
t = 17.5 seconds

Turbulent (left)
Laminar (right)

End of cooling down
t = 20 seconds

From these figures it can be seen that the turbulent variant reacts much more dynamically to the temperature change. To illustrate this fact, diagrams of measuring points can be created on the surface. The export of simulation data is described in more detail in subsection 9.2 below. Two points on the surface were defined for the export (Figure 34).

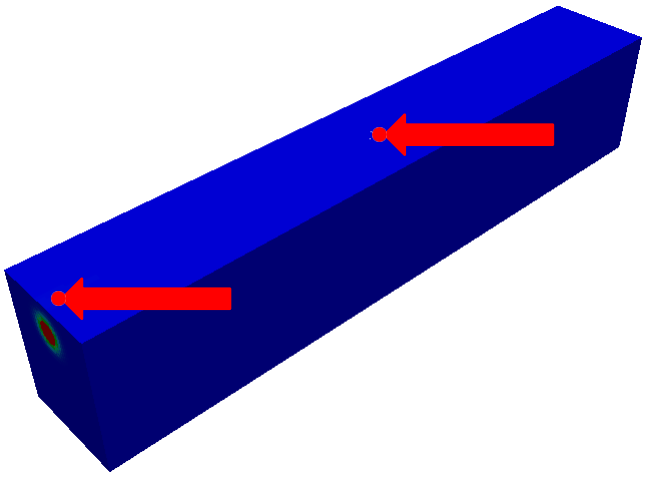


Figure 33: Defined measuring points on the surface of the simulated geometry

The first measuring point is 10 mm behind the inlet. The second measuring point was placed exactly in the middle of the block and is thus 250 mm from the inlet. For these points, the measured values are then generated and displayed in diagrams (Figure 34 and Figure 35). Figure 34 shows the temperature profile at a position 10 mm behind the inlet for the turbulent and laminar cases. In the turbulent case, the flow velocities are 0.533 m/s at 30 °C and 0.217 m/s at 90 °C. In the laminar case, the flow velocity is 0.0267 m/s at 30 °C and 0.011 m/s at 90 °C.

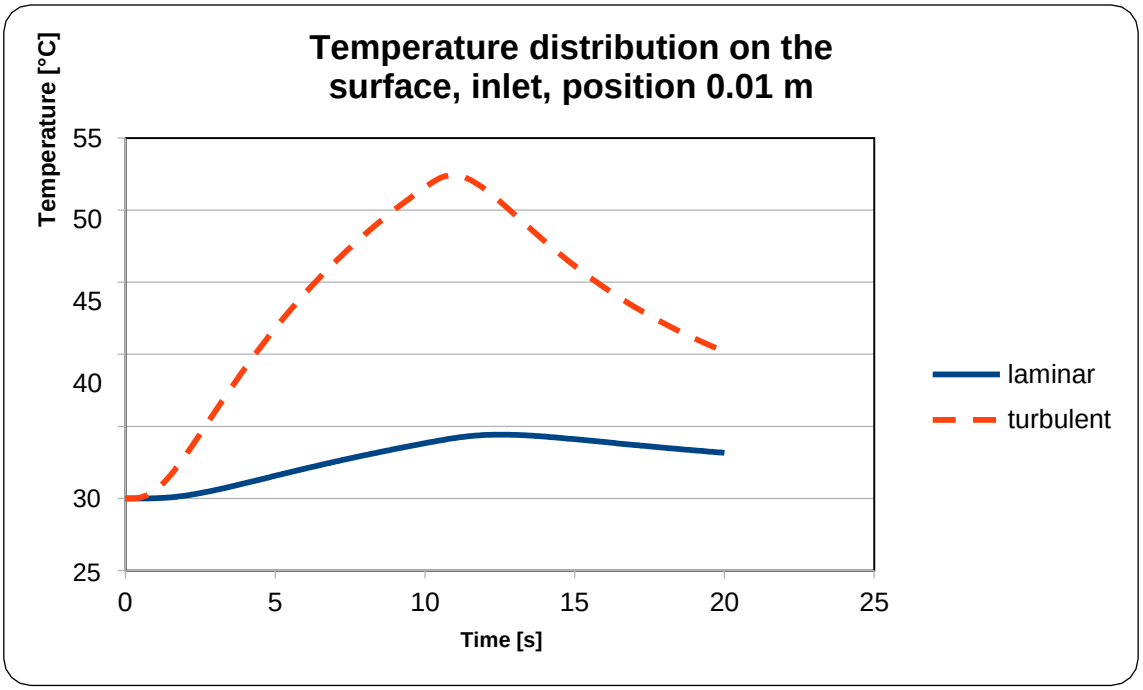


Figure 34: Temperature profile on the surface, inlet, position 0.01 m

It can be seen that the turbulent case behaves much more dynamically than the laminar one does. This is due to the increase in the heat transfer coefficient between solids and fluid at high Reynolds numbers. Figure 35 shows the temperature profile at the 250-mm position downstream of the inlet.

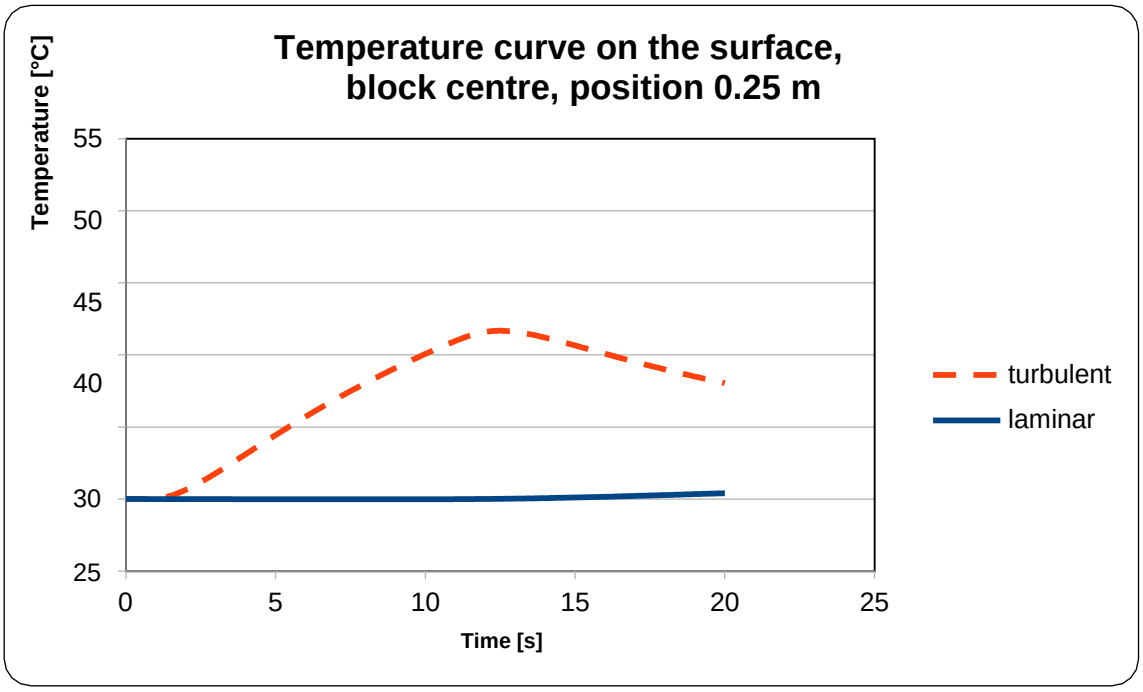


Figure 35: Temperature profile on the surface, block centre, position 0.25 m

It can be seen that the temperature increase is about 10 °C lower compared to the measuring point at 10 mm. In the laminar simulation case, there is no temperature increase, since the flow velocity is so low that the warm water arrives at the measuring point at 250 mm only in the middle of the cooling process.

9.2. Export of the measured values

After the geometry is displayed in Paraview, using the function ProbeLocation (Figure 36) a measuring point can be generated on the surface of the simulated geometry.

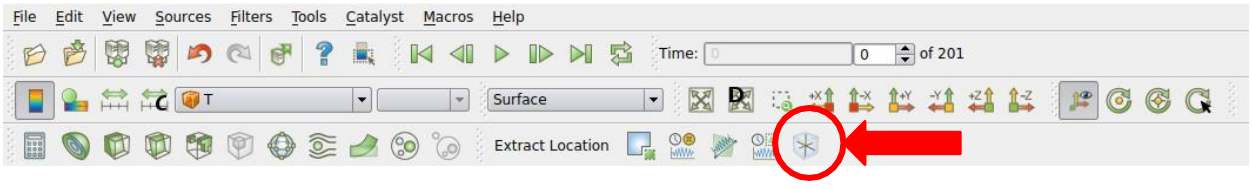


Figure 36: Paraview function ProbeLocation

Using the command at File > Save Data, selected data for the time steps can then be output. In the example shown, the temperature data are output. The drawback of exporting data in this way is that an Excel file is created for each time step. However, in the folder where the files are stored, the command

```
for i in `seq 0 1 200`; do cat T.$i.csv >> masterT01.csv; done
```

can be used to merge them into a single Excel file. However, this command also merges the headings of the 200 files, so that each measured value line has a superfluous heading line.

To delete these, the following commands must be executed:

- Open the file
  >vim createdfile.csv
- Start macro recording: Press
  key "q" twice to start
  macro recording
- Delete row
  Press key "d" twice
  Arrow down to skip line
- End macro recording: Press
  key "q" once
- Execute command several times a certain
  number: Enter the number of executions on the
  numeric keypad Start execution with @q
- Save and Exit
  :wq (write and quit)

It should be noted that Paraview has extensive options for the evaluation of various simulation cases. The examples shown here represent only very small portions of the available possibilities. Further information on the evaluation methods and tutorials for using Paraview can be found directly on the Paraview website [15].

10. Annex

10.1. Folder 0 > fluid

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      alphat;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0.56;

boundaryField
{
    ".*"
    {
        type            calculated;
        value           uniform 0.56;
    }
}


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

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0/cylinder";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 1.563;

boundaryField
{
    inlet
    {
        type            fixedValue;//epsilonWallFunction;
        value           uniform 1.563;
/*        type            uniformFixedValue;
        uniformValue    csvFile;
        csvFileCoeffs//uniformValueCoeffs
        {
                fileName        "$FOAM_CASE/T.csv"
                outOfBounds     clamp;
                nHeaderLine     1;
                refColumn       0;
                componentColumns (5); // vector example
                mergeSeparators 0;
        }
*/
    }
    outlet
    {
        type            zeroGradient;//epsilonWallFunction;
    }
    fluid_to_solid
    {
        type            epsilonWallFunction;
        value           uniform 1.563;//3.512;
    }
}


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

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0/cylinder";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0.273;

boundaryField
{
    inlet
    {
        type            fixedValue;//kqRWallFunction;
        value           uniform 0.273;
/*        type            uniformFixedValue;
        uniformValue    csvFile;
        csvFileCoeffs//uniformValueCoeffs
        {
                fileName        "$FOAM_CASE/T.csv"
                outOfBounds     clamp;
                nHeaderLine     1;
                refColumn       0;
                componentColumns (4); // vector example
                mergeSeparators 0;
        }
*/

    }
    outlet
    {
        type            zeroGradient;//kqRWallFunction;
    }
    fluid_to_solid
    {
        type            kqRWallFunction;
        value           uniform 0.273;
    }
}


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

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0/inside";
    object      mut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0.00324;

boundaryField
{
    inlet
    {
        type            fixedValue;//calculated;
        value           uniform 0.00324;
    }
    outlet
    {
        type            zeroGradient;//calculated;
    }

    fluid_to_solid
    {
        type            nutkWallFunction;
        value           uniform 0.00324;
    }
}


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

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 5e5;//1e5;

boundaryField
{
    inlet
    {
        type            zeroGradient;
        value           uniform 0;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 5e5;
    }
    fluid_to_solid
    {
        type            zeroGradient;//calculated;
        value           uniform 0;
    }

}

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

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 303.15;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 363.15;

/*      type            uniformFixedValue;
        uniformValue    csvFile;
        csvFileCoeffs//uniformValueCoeffs
        {
                fileName         "$FOAM_CASE/T.csv"
                outOfBounds      clamp;
                nHeaderLine      0;//1;
                refColumn        0;
                componentColumns (1); // vector example
                mergeSeparators 0;
        }

*/
    }
    outlet
    {
        type            zeroGradient;
    }
    fluid_to_solid
    {
//          type            fixedValue;
//          value           uniform 303.15;

        type            compressible::turbulentTemperatureCoupledBaffleMixed;
        value           uniform 311.55;
        phase           fluid;
        d               0.01;
        L               0.5;
        Cp              4181;
```

```
        Tnbr            T;
        kappa           fluidThermo;
        kappaName       k;
        kappaMethod     fluidThermo;

    }

}

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

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    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 0.533 0);

boundaryField
{
    fluid_to_solid
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    inlet
    {
        type            fixedValue;//
        value           uniform (0 0.533 0);//uniformFixedValue;
/*        uniformValue    csvFile;
        csvFileCoeffs
        {
            nHeaderLine     0;
            refColumn       0;
            componentColumns 3(3 2 3);
            separator       ",";
            mergeSeparators 0;
            fileName        "$FOAM_CASE/T.csv";
        }*/
    }
    outlet
    {
        type            zeroGradient;
    }
}


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

10.2. Folder 0 > solid

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 1e5;//12.66;

boundaryField
{
    solid_to_fluid
    {
        type            zeroGradient;//calculated;//fixedValue;
        value           uniform 1e5;//12.66;
    }
    wall
    {
        type            calculated;//fixedValue;
        value           uniform 1e5;//12.66;
    }


}

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

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    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   uniform 303.15;

boundaryField
{
    wall
    {
        type            externalWallHeatFluxTemperature;
        kappa           solidThermo;
        Ta              uniform 293.15;
        h               uniform 10.0;
        value           uniform 303.15;
        kappaName       none;
    }
    solid_to_fluid
    {
        type            compressible::turbulentTemperatureCoupledBaffleMixed;
        value           uniform 303.15;
        phase           solid;
        d               0.01;
        L               0.5;
        Cp              490;
        Tnbr            T;
        kappa           solidThermo;
        kappaName       k;
        kappaMethod     solidThermo;


    }
}


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

10.3. Folder constant > fluid

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;

    div(phi,U)      Gauss upwind;
    div(phi,K)      Gauss linear;
    div(phi,h)      Gauss upwind;
    div(phi,k)      Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div(phi,R)      Gauss upwind;
    div(R)          Gauss linear;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
    div((muEff*dev2(T(grad(U))))) Gauss linear;
    div(phiv,p)    Gauss linear;
    div(phi,e)     Gauss linear;
}

laplacianSchemes
{
    default        Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;

}

fluxRequired
{
    default         no;
    p_rgh;
}
// ************************************************************************* //

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    rho
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-7;
        relTol          0.1;
    }

    rhoFinal
    {
        $rho;
        tolerance       1e-7;
        relTol          0;
    }

    p_rgh
    {
        solver           GAMG;
        tolerance        1e-7;
        relTol           0.01;

        smoother         GaussSeidel;

        cacheAgglomeration  true;
        nCellsInCoarsestLevel 10;
        agglomerator     faceAreaPair;
        mergeLevels      1;

    }

    p_rghFinal
    {
        $p_rgh;
        tolerance        1e-7;
        relTol           0.01;
    }

    "(e|U|h|k|epsilon|R)"
    {
        solver           PBiCG;
        preconditioner   DILU;
        tolerance        1e-7;
        relTol           0.1;
    }

    "(e|U|h|k|epsilon|R)Final"
    {
        $U;
        tolerance        1e-7;
        relTol           0;
    }
}

PIMPLE
{
    momentumPredictor   yes;
    nCorrectors         2;
    nNonOrthogonalCorrectors 1;
}

relaxationFactors
{
    equations
    {
        "h.*"           0.1;
        "U.*"           0.5;
    }
}

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

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           (0 -9.81 0);

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

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant/bottomAir";
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState rhoConst;
    specie          specie;
    energy          sensibleInternalEnergy;//Enthalpy;
}

mixture
{
    specie
    {
        nMoles          1;
        molWeight       18.0153;
    }
    equationOfState
    {
        rhoCoeffs<8>    (-294.2911 13.14555 -0.04759 7.33E-5 -4.3E-8 0 0 0);//( 2.5039 -0.010587 2.0643e-05 -1.8761e-08 6.4237e-12 0 0 0 );
        rho             1000;
    }
    thermodynamics
    {
        Hf              -13423000;
        Sf              10482;
        CpCoeffs<8>     ( 1563.1 1.604 -0.0029334 3.2168e-06 -1.1571e-09 0 0 0 );
        Cp              4181;
    }
    transport
    {
        muCoeffs<8>     ( 1.5068e-06 6.1598e-08 -1.8188e-11 0 0 0 0 0 );
        kappaCoeffs<8>  ( 0.0037972 0.00015336 -1.1859e-08 0 0 0 0 0 );
        mu              959e-6;
        kappa           0.6;
        Pr              6.62;
    }
}
// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


        transportModel  Newtonian;
        nu              0.8e-6;


```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM Extend Project: Open Source CFD        |
|  \\    /   O peration     | Version:  1.6-ext                               |
|   \\  /    A nd           | Web:      www.extend-project.de                 |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RAS;


RAS

    {
        RASModel kEpsilon;
        turbulence on;
        printCoeffs on;
    }

 turbulentFields  k epsilon;
//turbulentFields rhoCr rhoCu;

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

10.4. Folder constant > solid

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
}

laplacianSchemes
{
    default             none;
    laplacian(alpha,h)  Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}


// ************************************************************************* //

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    h
    {
        solver           PCG;
        preconditioner   DIC;
        tolerance        1e-06;
        relTol           0.1;
    }

    hFinal
    {
        $h;
        tolerance        1e-06;
        relTol           0;
    }
}

PIMPLE
{
    nNonOrthogonalCorrectors 1;
}

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

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant/bottomAir";
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

thermoType
{
    type            heSolidThermo;
    mixture         pureMixture;
    transport       constIso;
    thermo          hConst;
    equationOfState rhoConst;
    specie          specie;
    energy          sensibleEnthalpy;
}



mixture
{
    specie
    {
        nMoles          1;
        molWeight       55.85;//28.9;
    }
    thermodynamics
    {
        Cp              460;
        Hf              0;
    }
    transport
    {
//        mu              1.8e-05;
//        Pr              0.7;
        kappa           45;//50;

    }
    equationOfState
    {
        rho             7850;
        kappa           50;
    }
}
// ************************************************************************* //
```

11. References

[1] „CAESES Software › CAESES “.

[2] E. S. I. Group, „Visual-CFD for OpenFOAM® “, *ESI Group*, 10-JUN-2016. [Online]. Available at: https://www.esi-group.com/software-solutions/virtual-environment/cfd-multiphysics/visual-cfd-openfoam. [Accessed on: 03-DEC-2018].

[3] „SimFlow CFD Software - OpenFOAM® GUI “, *simFlow CFD*. [Online]. Available at: https://sim-flow.com/. [Accessed on: 03-DEC-2018].

[4] „MantiumFlow, OpenFOAM CFD Simulations, download demo and tutorials “, *MantiumFlow, CFD Simulation Software with OpenFOAM®*. [Online]. Available at: https://mantiumflow.com/. [Accessed on: 03-DEC-2018].

[5] „OpenFOAM® Documentation “. [Online]. Available at: https://www.openfoam.com/documentation/. [Accessed on: 03-DEC-2018].

[6] P. Geisser, *Temperiertechnik – Theorie und Praxis*, 3rd ed., HB-Therm AG, 2016.

[7] B. Foundation, „blender.org - Home of the Blender project - Free and Open 3D Creation Software “, *blender.org*.

[8] A. Pflanzer & F. Schneider, „Export von Creo-Dateien in das STL-Format “, 2017.

[9] E. Kobler, „Abbildung des Maschinenverhaltens eines Spritzgießprozesses mittels Füllsimulation einer Stufenplatte in OpenFoam® “. Johannes Kepler University of Linz, 29-May-2015.

[10] OpenCFD, „Mesh generation with the snappyHexMesh utility “. [Online]. Available at: http://www.openfoam.com/documentation/user-guide/snappyHexMesh.php. [Accessed on: 22-SEP-2017].

[11] C. Soulaine, „Introduction to computational fluid mechanics using the OpenFOAM® technology « Simulation in porous media from pore to large scale» “, 27-JUN-2016.

[12] R. Schwarze, *CFD-Modellierung*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[13] „OpenFOAM®: k-epsilon “. [Online]. Available at: http://www.openfoam.com/documentation/cpp-guide/html/guide-turbulence-ras-k-epsilon.html. [Accessed on: 27-SEP-2017].

[14] “Material parameters of water”. [Online]. Available at: http://www.uni-magdeburg.de/isut/LSS/Lehre/Arbeitsheft/IV.pdf. [Accessed on: 26-SEP-2017].

[15] „ParaView “.