AutorInnen:
**Thomas Mitterlehner**
**Eva Kobler**
**Georg Steinbichler**

Institut:
**Institut für Polymer-Spritzgießtechnik und Prozessautomatisierung**

Datum:
**März 2019**

# HANDBUCH

## Einstieg in OpenFoam® und chtMultiRegion anhand eines anwendungsorientierten Beispiels

**JOHANNES KEPLER UNIVERSITÄT LINZ**
Altenberger Straße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

# Vorwort

Im Rahmen des Projekts AM 4 Industry wurde die Auslegung und Simulation von konturnahen Kühlkanälen erforscht. Unter anderem wurde das Ziel verfolgt, WerkzeugbauingenieurenInnen ein Tool zur weiteren Optimierung von Kühlkanalgeometrien zur Verfügung zu stellen. Insbesondere bei der Gestaltung von konturnahen und regellosen Kühlkanälen kann die Simulation mit kommerziellen Simulationsprogrammen nicht ausreichend sein, weshalb zusätzliche Simulationsschritte erforderlich sind. Für diese zusätzlichen oder ergänzenden Simulationsschritte wurde bewusst der Fokus auf ein nicht kommerzielles Simulationsprogramm gelegt. Daher wurde für die Simulationen das Programm OpenFoam® (**Open** Source **F**ield **O**peration **a**nd **M**anipulation) gewählt. OpenFoam® ist frei erhältlich und wird vorwiegend für die Lösung von Strömungsproblemen (CFD) verwendet. Es ist in C++ geschrieben und bringt bereits in der Basisversion nützliche Löser (Solver) mit sich. Darüber hinaus kann eine Vielzahl an weiteren Lösern adaptiert werden. Einer der größten Vorteile besteht darin, dass der Sourcecode und somit auch die Algorithmen frei zugänglich sind. Des Weiteren können die Codes und Berechnungen nahezu beliebig erweitert werden.
Anhand eines anwendungsorientierten Beispiels wird in diesem Handbuch der Aufbau und die Durchführung einer Simulation ausführlich erklärt. Für die Simulationen wurde der Löser „chtMultiRegion“ verwendet. Dieser wird allgemein für die Berechnung des Wärmeaustausches zwischen einem Festkörper und einem Fluid verwendet.
Ziel dieses Handbuchs ist es, PraktikernInnen in der Entwicklung, SimulationsingenieurenInnen und StudentenInnen einen anwendungsorientierten Einstieg in sowie einen Überblick über die Arbeit mit OpenFoam® zu geben. Die einzelnen Schritte wurden in neun Kapitel mit folgenden Inhalten eingeteilt:

- Kapitel 1 und 2 geben einen allgemeinen Überblick über das Simulationsbeispiel sowie über die Simulationsstrukturen von OpenFoam®.

- Kapitel 3 bis 6 zeigen, wie ein Simulationsfall in OpenFoam® aufgesetzt wird und welche Einstellungen erforderlich sind. Dabei wird genau auf die Erstellung des Rechengitters und die Zuordnung der Flächen eingegangen.

- Kapitel 7 und 8 behandeln die Durchführung der Simulation. Dabei wird auf die Wahl der Randbedingung sowie auf die Theorie der Strömungssimulation eingegangen.

- Kapitel 9 behandelt abschließend die Auswertungsmethoden, welche von OpenFoam® geboten werden sowie die Möglichkeiten zum Export der Ergebnisse für die weitere Verarbeitung.

Abschießend ist zu erwähnen, dass die OpenFoam®-Umgebung auf den ersten Blick durchaus befremdlich wirken kann, insbesondere, wenn üblicherweise auf Windows-Betriebssystemen gearbeitet wird. Hier ist mit Sicherheit Geduld gefragt. Es dauert eine gewisse Zeit, bis die Abläufe des Programms logisch erscheinen und bis Befehle, die für die Bedienung und Ausführung des Programms nötig sind, leicht von der Hand gehen. Dennoch wird sich eine intensive Einarbeitung in die Thematik lohnen, da sich so Simulationen weiter verfeinern und bessere Vorhersagen treffen lassen. Dies wiederum kann einen klaren Wettbewerbsvorteil bieten. In diesem Sinne: Happy Foaming!

# Inhaltsverzeichnis

# 1. Einleitung

Das Softwarepaket OpenFoam® wird üblicherweise auf Linux betrieben. In dem hier beschriebenen Fall erfolgt der Betrieb auf Ubuntu Version 14.04.5 LTS. Für die Durchführung von Simulationen ist eine Installation eines Linux-Betriebssystems erforderlich. Folgende Möglichkeiten zum Betrieb von OpenFoam® bzw. eines Linux-Systems sind möglich:

- Direkte Installation von OpenFoam® auf einem Rechner mit Linux-Betriebssystem
- Dualboot: Zwei oder mehrere Betriebssysteme (z.B. Windows und Ubuntu) sind auf einem Rechner installiert und bei jedem Systemstart kann ausgewählt werden, mit welchem Betriebssystem gestartet wird.
- Verwendung eines Serverrechners: Installation eines Linux-Betriebssystems auf einem separaten Rechner, auf den über ein Netzwerk mit z.B. Putty zugegriffen werden kann.

Hierbei ist auch anzumerken, dass OpenFoam® standardmäßig keine Benutzeroberfläche (GUI) besitzt. Somit müssen sämtliche Befehle über die Kommandozeile eingegeben werden. Es sind jedoch Benutzeroberflächen kommerziell erhältlich, die die Bedienung erleichtern können. Dazu zählen beispielsweise CAESES [1], Visual-CFD [2], SimFlow [3] oder MantiumFlow [4]. Die allgemeine Struktur von OpenFoam® ist in Abbildung 1 dargestellt.

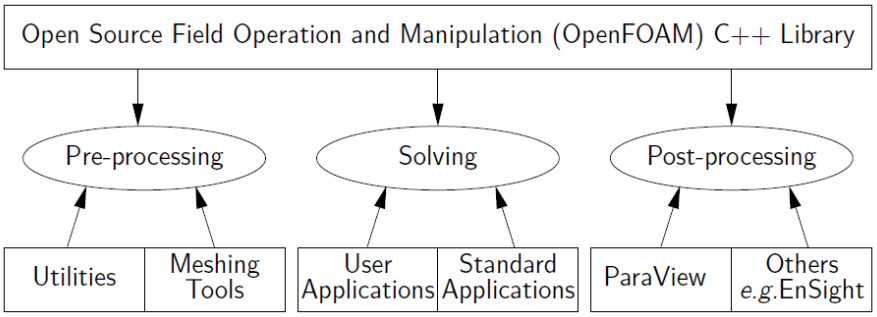



Abbildung 1: Übersicht der OpenFoam® Struktur [5]

Diese setzt sich aus den drei Teilgebieten Pre-Processing, Solving und Post-Processing zusammen. Zum Pre-Processing zählen nicht die Programme, wie z.B. Blender oder PTC Creo, sondern viel mehr die Softwarepakete, welche die Geometrie für die Löser (Solver) vorbereiten. Dazu gehören zum Beispiel „blockMesh“ oder „snappyHexMesh“. Zu den Lösern zählen die Softwarepakete, welche die Rechnungen bzw. Simulationen durchführen. OpenFoam® ist standardmäßig mit zahlreichen Lösern ausgestattet und es können aus diversen Internetquellen oder –foren weitere hinzugefügt werden. Der Löser „chtMultiRegion“ ist standardmäßig in OpenFoam® integriert und unter den „Heat transfer and buoyancy-driven flows“-Lösern zu finden.
Exakt um diesen „chtMultiRegion“-Solver dreht sich dieses Handbuch. Hierzu werden anhand eines anschaulichen Beispiels Schritt für Schritt die Vorgänge beschrieben. Zu diesem Zweck wurde ein Anwendungsfall aus der Kunststoffverarbeitung bzw. aus der Spritzgießtechnik gewählt. Die Simulation soll dazu beitragen, die Temperaturverteilung in einem Spritzgießwerkzeug darzustellen. Im Einzelnen wird ein sogenannter variothermer Spitzgießprozess dargestellt. Grundsätzlich wird bei einem Spritzgießprozess das Kunststoffgranulat, welches bei Raumtemperatur in fester Form vorliegt, in einem beheizten Zylinder aufgeschmolzen. Dies geschieht bei etwa 200°C. Nach dem Plastifizieren liegt der Kunststoff in flüssiger Form vor und wird mit ca. 500 bar in eine Metallform injiziert. Die mittels

Temperiergeräten gekühlte Metallform sorgt nicht nur für die geometrischen Abformung, sondern auch für die Abkühlung des Formteils. Weist das Formteil jedoch sehr feine Strukturen (Mikrostrukturen) auf, reicht das konventionelle Werkzeugkühlkonzept nicht aus. Hier kommt dann der zuvor genannte Variothermprozess zur Anwendung. Dabei wird die Metallform vor dem Einspritzvorgang in etwa auf die Schmelztemperatur beheizt. Noch während des Einspritzvorganges wird diese dann wieder abgekühlt. In Abbildung 2 sind die Vorgänge während des Füllvorganges beider Prozessvarianten dargestellt.

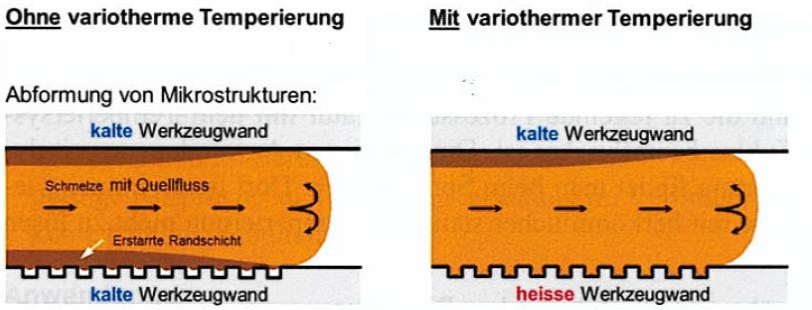



Abbildung 2: Füllvorgang beim Spritzgießen, Vergleich ohne und mit variothermer Temperierung [6]

Hierbei wird ersichtlich, dass die Kunststoffschmelze bei Kontakt mit der kalten Werkzeugwand erstarrt. Dies mindert die Fähigkeit, Mikrostrukturen abzuformen. Ist die Werkzeugwand jedoch beheizt, hat die Kunststoffschmelze genügend Zeit, um in die feinen Strukturen zu fließen. Bewerkstelligt wird der Aufheiz- und Abkühlvorgang durch zwei getrennte Kühlgeräte. Als Kühl- bzw. Heizmedium wird zumeist Wasser oder Öl eingesetzt. In der nachfolgenden Abbildung ist ein solches Temperierkonzept beispielhaft dargestellt.

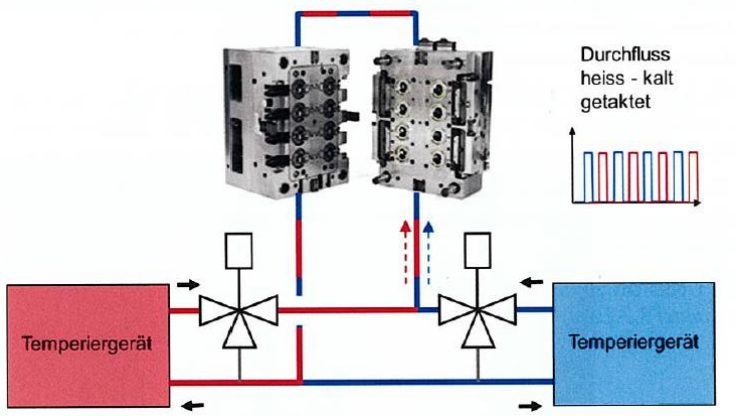



Abbildung 3: Prinzip der variothermen Flüssigtemperierung mit zwei Temperiergeräten und einer Ventil-Umschalteinheit [6]

Um qualitativ hochwertige Kunststoffprodukte herstellen zu können, ist die Kenntnis der exakten Werkzeugwandtemperatur essenziell. Aus diesem Grund lohnt es sich, diesen Prozess zum Beispiel mithilfe von OpenFoam® zu simulieren und genauer zu betrachten.
Dieses Handbuch soll als erste Grundlage dienen, um die beschriebenen Vorgänge mittels OpenFoam® abbilden zu können. Für einen ersten Überblick sind in Tabelle 1 die einzelnen Schritte der Herangehensweise, eingeteilt in Aufbereitung, Pre-Processing, Solving und Post-Processing, dargestellt. Hinsichtlich der Modellierung und Einbindung der Geometriedaten in OpenFoam® bieten sich verschiedenste Vorgehensweisen. Der in dieser Arbeit beschriebene Simulationsfall wird mit folgender Herangehensweise gelöst:

Tabelle 1: Vorgehensweise bei der Simulation mit OpenFoam® und chtMultiRegion

| Schritt | Beschreibung | Einteilung |
|---|---|---|
| **1** | Konstruktion der Simulationsgeometrie mit **PTC Creo** und Export des CAD-Models als .stl-Datei | Aufbereitung (Seite 7) |
| **2** | Kontrolle und Nachbearbeitung des CAD Models mit **Blender** | Aufbereitung (Seite 8) |
| **3** | Erstellung eines ersten Rechengitters mit **„blockMesh“** | Pre-Processing (Seite 16) |
| **4** | Verfeinerung des Rechengitters mit **„snappyHexMesh“** | Pre-Processing (Seite 21) |
| **5** | Zuordnung der Regionen mit **„splitMeshRegions“** und **„surfaceToPatch“** | Pre-Processing (Seite 26) |
| **6** | Simulation des Wärmeflusses bzw. –austausches zwischen Werkzeug und Kühlmittel mit **„chtMultiRegion“** | Solving (Seite 32) |
| **7** | Darstellung der Ergebnisse mit **„ParaView“** | Post-Processing (Seite 39) |

Als Geometrie für das Simulationsbeispiel wurde ein länglicher Quader gewählt, welcher stirnseitig in der oberen Hälfte eine Temperierbohrung besitzt. Durch diese Bohrung soll Wasser mit einer definierten Geschwindigkeit und Temperatur fließen. Ziel ist es den Aufheiz- und Abkühlvorgang an der oberen Fläche zu bestimmen. In der nachfolgenden Abbildung ist die Simulationsgeometrie dargestellt.

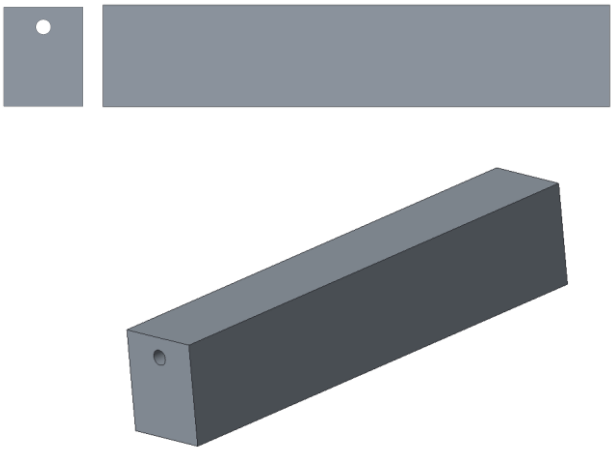


Abbildung 4: Darstellung der Simulationsgeometrie

In den folgenden Kapiteln ist das beschriebene Simulationsbeispiel Schritt für Schritt dokumentiert. Dabei werden auch immer wieder die auszuführenden Befehle erwähnt. Diese sind farbig dargestellt, wobei *PTC-Creo-Befehle in blau*, Blender-Befehle in gelb und *OpenFoam®-Befehle in rot* gekennzeichnet sind.

# 2. Aufbereitung

Unter den Punkt Aufbereitung fallen alle Schritte, die nicht mit OpenFoam® oder in einer Linux-Umgebung durchgeführt werden müssen. In diesem Handbuch bzw. im gezeigten Beispiel wurden alle Konstruktionen mit PTC Creo umgesetzt. Die Konstruktion ist jedoch nicht auf das Programm PTC Creo begrenzt, sondern kann beliebig durchgeführt werden. Einzige Bedingung ist, dass die Geometrie im .stl-Format an OpenFoam® übergeben wird. Da beim Export der Geometrie oftmals Fehler auftreten, wird diese danach zusätzlich mit dem frei erhältlichen Programm Blender [7] kontrolliert und gegebenenfalls repariert. Ein typischer Fehler ist hierbei zum Beispiel eine nicht geschlossene Fläche.

## 2.1. Konstruktion der Simulationsgeometrie

In einem ersten Schritt wurde damit begonnen, die Grundgeometrie mit PTC Creo zu konstruieren. Als zu simulierende Geometrie wurde der in Abbildung 4 gezeigte Stahlblock gewählt, welcher eine in Längsrichtung verlaufende Bohrung besitzt. Dies soll einen stark vereinfachten Aufbau eines Spritzgießwerkzeuges mit einer darin verlaufenden Kühlbohrung darstellen. In der nachfolgenden Abbildung sind die Maße dieser Grundgeometrie illustriert.

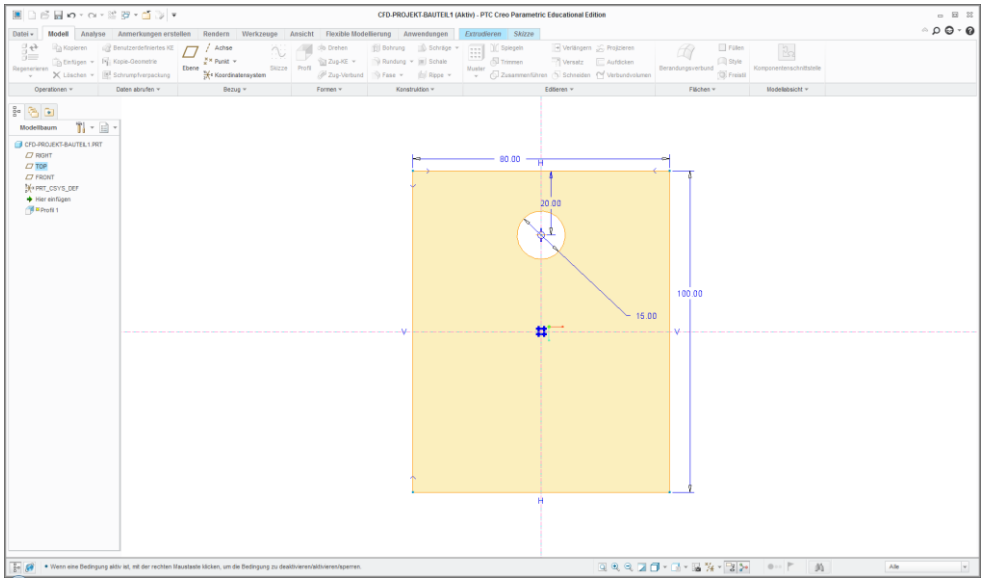



Abbildung 5: Grundgeometrie des Simulationsmodells im CAD-Programm PTC Creo

Die Breite der Grundgeometrie beträgt 80 mm und die Höhe 100 mm. Die Bohrung besitzt einen Durchmesser von 15 mm und wurde 20 mm unterhalb der oberen Fläche platziert. Diese Grundgeometrie wurde auf eine Länge von 500 mm extrudiert. Als Werkstoff wurde Stahl des Typs 1.2343 (X38 CrMoV 51) gewählt, da dieser üblicherweise für Formeinsätze bei Spritzgießwerkzeugen verwendet wird.

Im Anschluss daran wurde dieses Model als .stl-File exportiert. Die Exportierfunktion befindet sich im CAD-Programm PTC Creo unter
*Datei > Speichern als > Kopie speichern*
Danach erscheint der Speicherexplorer. In diesem ist der Dateityp „Stereolithographie (.stl)“ zu wählen. Nachdem mit „OK“ bestätigt wurde, erscheint ein Fenster, in dem Exporteinstellungen gewählt werden können. Dabei ist es für die spätere Verarbeitung mit OpenFoam® notwendig, dass beim Format „ASCII“ (Amerikanischer Standard-Code für den Informationsaustausch) ausgewählt wird. Dabei wird bestimmt,

ob die Datei nur in der Maschinensprache (Binär) oder auch mit einem Texteditor (ASCII) lesbar sein soll. Des Weiteren sind die Sehnenhöhe und die Winkelsteuerung einzustellen. Bei der Sehnenhöhe ist ein Wert im Bereich von 0.01 und bei der Winkelsteuerung der Wert 1 einzugeben. Durch die Angabe der Sehnenhöhe wird der maximale Abstand zwischen einer Sehne und einer Oberfläche angegeben. Je geringer die Sehnenhöhe gewählt wird, desto geringer ist die Abweichung von der tatsächlichen Bauteiloberfläche. Zum besseren Verständnis ist in Abbildung 6 eine Erläuterung abgebildet.

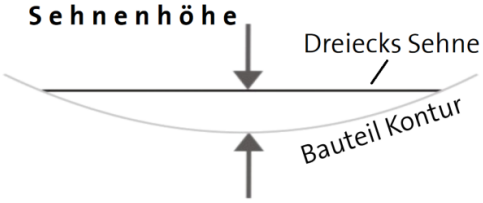



Abbildung 6: Erläuterung Sehnenhöhe [8]

Beachten Sie jedoch: Je kleiner die Sehnenhöhe gewählt wird, desto größer wird die exportierte .stl-Datei. Die Winkelsteuerung kann in einem Bereich von 0 bis 1 eingestellt werden. Dieser Wert bestimmt die Anpassung der Sehnenhöhe bei kleinen Radien. Der Wert 1 bewirkt eine genaue Anpassung der Dreiecke an kleine Radien. Hingegen bewirkt ein Wert von 0 keine Anpassung. In der nachfolgenden Abbildung 7 ist ein Vergleich zweier verschiedener Einstellungen dargestellt.

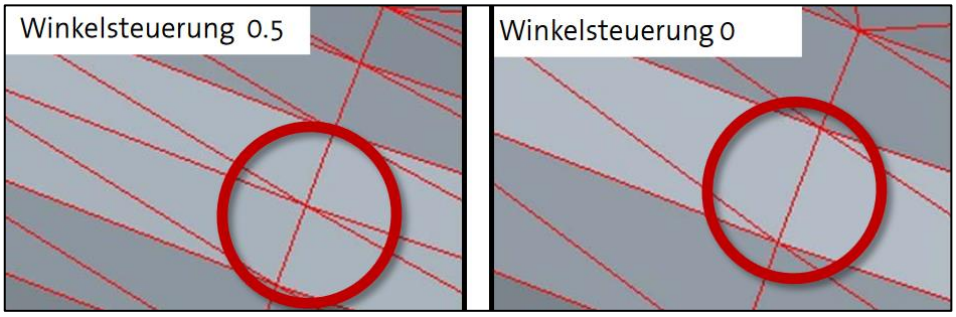



Abbildung 7: Vergleich zweier Winkelsteuerungseinstellungen [8]

Wurde das Format, Sehnenhöhe und Winkelsteuerung eingestellt, bestätigt man die Einstellungen mit „OK“ und die Geometrie wird als stl-File exportiert.

## 2.2. Kontrolle und Nachbearbeitung

Liegt die Geometrie als .stl-File vor, kann mit der Arbeit in Blender begonnen werden. Hierzu muss zunächst die Geometrie importiert werden. Dies kann mit

File > Import

durchgeführt werden. Um mit dem Modell optimal arbeiten zu können, muss zunächst der richtige Modus eingestellt werden. Standardmäßig ist zu Beginn der Modus „object mode“ eingestellt. Dieser muss auf „edit mode“ umgestellt werden. Einzustellen ist dies im unteren Bildschirmbereich (Abbildung 8).

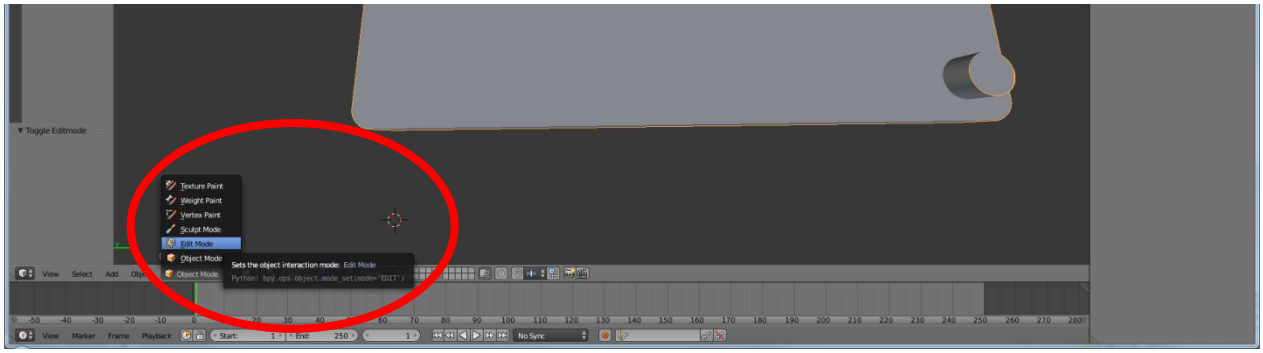



Abbildung 8: Einstellung „edit mode“ in Blender

Des Weiteren ist die Auswahl auf Fläche („face“) einzustellen. Dies kann wiederum am unteren Bildschirmrand eingestellt werden. Mit der

Taste 5

am Nummernblock kann zudem noch das Zoomverhalten verbessert werden. Sind diese Einstellungen erfolgt, kann mit der rechten Maustaste eine Fläche ausgewählt werden. Um mehrere Flächen auszuwählen, muss die Shift-Taste gedrückt und gehalten werden.

Damit OpenFoam® mit der CAD-Geometrie arbeiten kann, müssen zunächst offene Geometrieflächen geschlossen werden. Bei der angezeigten Geometrie ist beispielsweise beim Kühlkanaleinlass und –auslass die Geometrie nicht geschlossen. Um diese Öffnungen zu schließen, ist die Füllfunktion eine geeignet Wahl. Hierfür ändert man zunächst den Auswahlmodus auf Punktauswahl („vertex select“). Danach betätigt man die

c-Taste

woraufhin ein Auswahlkreis erscheint. Durch Betätigen der linken Maustaste werden alle Punkte im Auswahlkreis markiert. Mit dem Mausrad kann der Auswahlkreis verkleinert oder vergrößert werden. Sind alle notwendigen Punkte ausgewählt, muss die rechte Maustaste betätigt werden, um die Auswahl zu beenden. Danach ist die

f-Taste

zu betätigen, um die Fläche, die die Punkte einschließt, zu füllen.

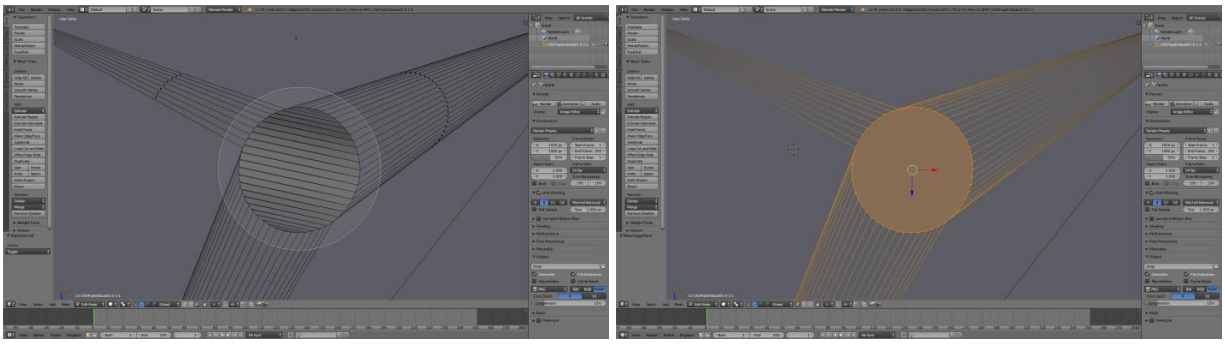


Abbildung 9: Auswahlkreis durch Betätigung der c-Taste (links), Füllen der Fläche durch Betätigung der f-Taste

Zwischen den einzelnen Füllbefehlen muss immer darauf geachtet werden, dass nicht auch noch andere Flächen ausgewählt sind. Um sicherzustellen, dass keine anderen Punkte angewählt sind, sollte zwischen verschiedenen Befehlen die

a-Taste

betätigt werden, da auf diese Weise alle gewählten Punkte abgewählt werden. Nachdem alle Geometrien erfolgreich geschlossen wurden, kann mit der Funktion

Non manifold

überprüft werden, ob tatsächlich alle Flächen geschlossen sind. Zu finden ist diese Funktion am linken unteren Bildschirmrand unter

Select > Non Manifold oder Select > Select All by Trait > Non Manifold oder durch Drücken von Shift+Ctrl+Alt+M

Ist der „edit mode“ aktiviert und die Ansicht auf „wireframe“ eingestellt, werden die offenen Bereiche hervorgehoben. Durch Drücken der

n-Taste

können zusätzlich die xyz-Koordinaten des offenen Bereichs angezeigt werden.

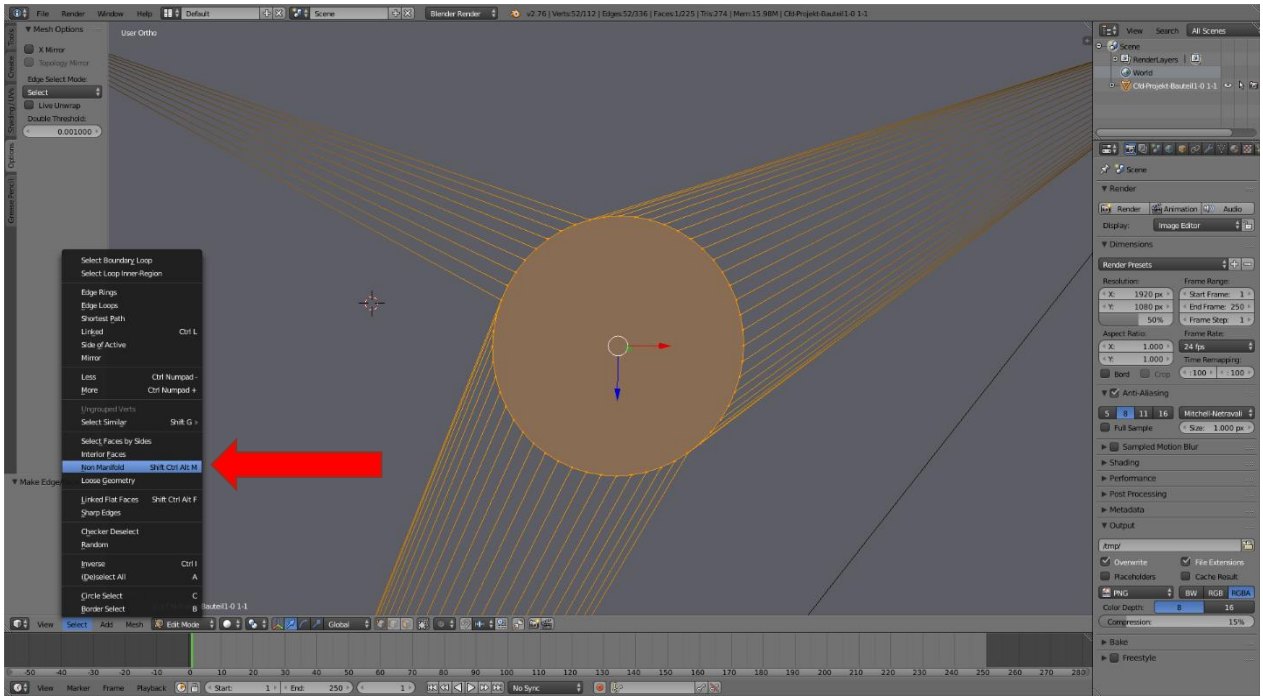


Abbildung 10: Menüpunkt „non manifold“

Nachdem sichergestellt wurde, dass alle Geometrien geschlossen sind, müssen die einzelnen Flächen definiert werden, damit diesen später in OpenFoam® bestimmte Eigenschaften, zum Beispiel Einlass und Auslass des Kühlmediums, zugewiesen werden können. Das heißt, jede Fläche, die eine eigene Eigenschaft besitzen soll, muss separat als .stl-Datei exportiert werden. Dazu zählen der Einlass, der Auslass, das Kühlrohr und der Stahlblock. Im dies durchzuführen, muss auf den Modus der Flächenauswahl gewechselt werden und die zu definierende Fläche ausgewählt werden. Anschließend ist die

p-Taste

zu betätigen und „section“ auszuwählen. Danach erscheint die ausgewählte Fläche separat im rechten Strukturbaum. Nun kann jeder Fläche ein Name zugeordnet werden. Für OpenFoam® sind folgende Definitionen wichtig:

- „inlet“
- „outlet“
- „tube“
- „wall“

Im 6. Kapitel sind die zu exportierenden Flächen bzw. Regionen dargestellt und ausführlich beschrieben. Werden beim Auswählen der Flächen einzelne Teilflächen übersehen, können diese im Nachhinein hinzugefügt werden. Dabei müssen die übersehenen Flächen ausgewählt und die p-Taste gedrückt werden. Danach sind die übersehenen Flächen sowie die Fläche, zu der diese hinzugefügt werden sollen, auszuwählen. Schließlich ist links im Menübaum

Tools > Edit > Join

der „join“-Befehl auszuführen. Dadurch werden die beiden ausgewählten Flächen zusammengeführt. Damit das Modell ordnungsgemäß simuliert werden kann, müssen für den Import in OpenFoam® folgenden .stl-Files vorhanden sein:

„inlet“ (einzeln)
„outlet“ (einzeln)
„tube“ (einzeln)
„wall“
(alle, die nicht zu „inlet“, „outlet“ oder „wall“ zählen)
„solid“
(„snappyHexMesh wall“ - alle Flächen, die später die gleiche „solid“-Randbedingung besitzen sollen)
„fluid“
(„inlet“, „outlet“ und „wall“ – alle Flächen, die später die „fluid“-Randbedingung besitzen sollen)
Damit mehrere Flächen als eine einzelne .stl-Datei exportiert werden können (z.B. „fluid“), müssen diese rechts im Strukturbaum ausgewählt werden. Die Auswahl mehrere Dateien erfolgt mit

Shift + linker Maustaste

Nach Auswahl der einzelnen Flächen können diese unter File > Exportieren > Stl (.stl)
exportiert werden. Beim Exportieren ist darauf zu achten, dass im Exportmenü der Haken bei ASCII gesetzt wird.

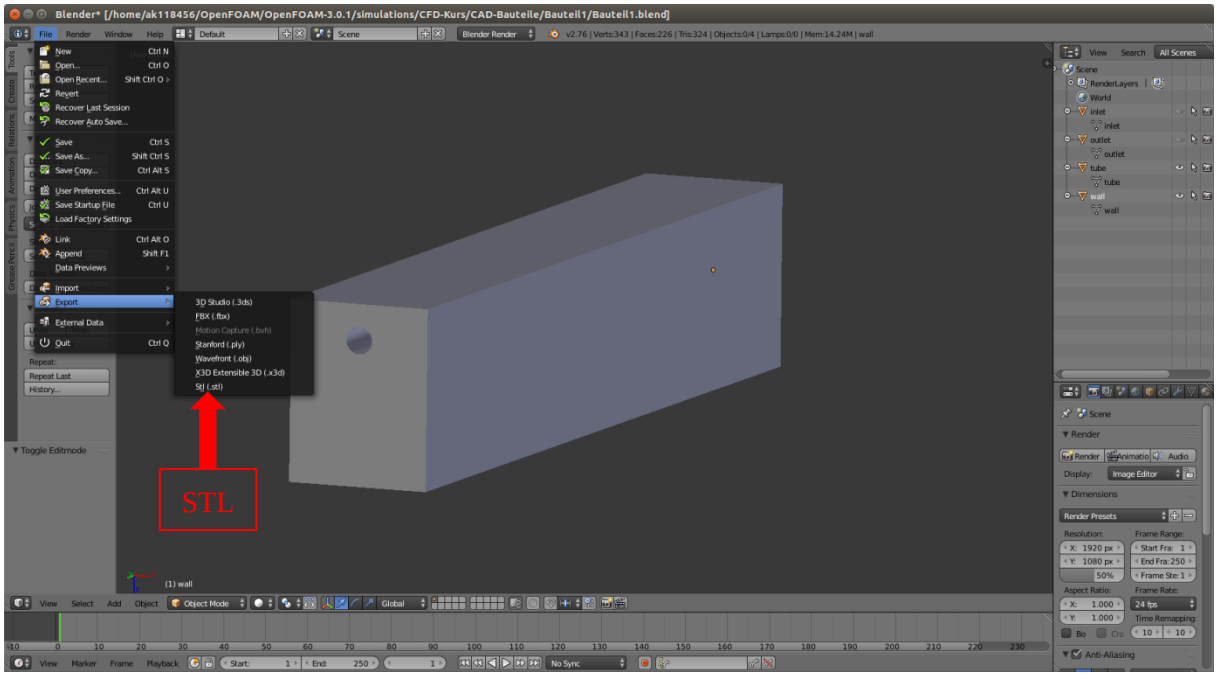



Abbildung 11: Blender Menüpunkt des stl-Exports

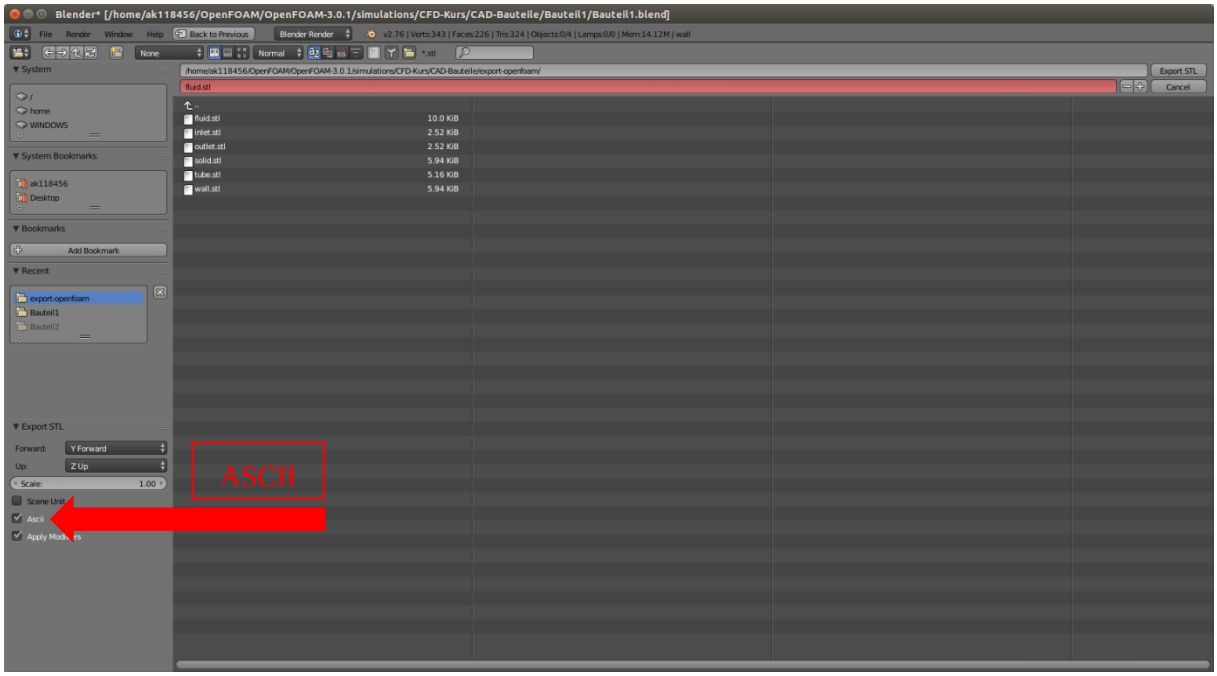



Abbildung 12: Blender stl-Exporteinstellungen, ASCII

# 3. Pre-Processing mit OpenFoam®

Um mit dem Pre-Processing in OpenFoam® beginnen zu können, müssen alle Geometrien separat vorhanden sein, d.h. es müssen „inlet.stl“, „outlet.stl“, „tube.stl“, „wall.stl“, „fluid.stl“ und „solid.stl“ vorliegen.
Wenn alle Geometriedateien vorhanden sind, kann mit dem Aufbau der Simulation begonnen werden. Der in diesem Simulationsbeispiel beschriebene Fall soll den Aufheiz- und Abkühlvorgang eines Stahlblocks zeigen, der von einem Medium durchströmt wird. Hierfür wird zunächst mittels „blockMesh“ und „snappyHexMesh“ die Vernetzung der Geometrie durchgeführt. Danach soll mithilfe des Lösers „chtMultiRegionFoam“ die Temperaturverteilung dargestellt werden.

Simulationen, die mit OpenFoam® durchgeführt werden, besitzen oftmals folgende Standardstruktur:

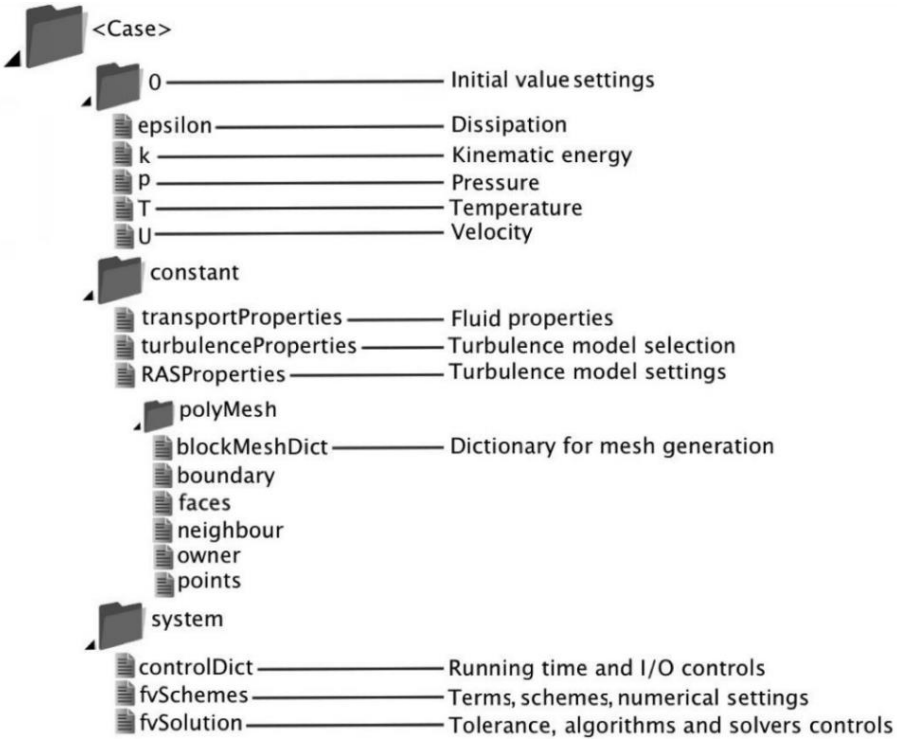


Abbildung 13: Ordnerstruktur der meisten OpenFoam® Simulationen

Wird mit dem Aufsetzen einer Simulation in OpenFoam® begonnen, empfiehlt es sich, einen bereits vorhandenen Simulationsfall zu kopieren und diesen anzupassen. Für die meisten Löser sind in der Standardinstallation von OpenFoam® Beispiele vorhanden, welche im Ordner „tutorials“ zu finden sind.
Das Simulationsbeispiel, das in den nachfolgenden Schritten genauer beschrieben wird, ist folgendermaßen gegliedert:

Name des Simulationsbeispiels: cfdkurs
Speicherort des Simulationsbeispiels: ~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$

Im Speicherort des Simulationsbeispiels, auch Home-Verzeichnis genannt, werden in weiterer Folge alle Befehle im Terminal (Shell) ausgeführt. Muss also zum Beispiel das erste Rechengitter mittels „blockMesh“ erstellt werden, muss zuerst der Ordner „cfdkurs“ geöffnet und danach der Befehl

>blockMesh eingegeben werden. Im Verzeichnis „cfdkurs“ befinden sich die Ordner „0“, „constant“ und „system“.

## 3.1. Gittererstellung allgemein

In OpenFoam® kann das Gitter mithilfe der „snappyHexMesh“-Funktion erstellt werden. Der Ablauf einer solchen Erstellung soll anhand eines einfachen Beispiels erklärt werden [9], [10].

Schritt 1: Für den ersten Schritt wird eine .stl-Datei der Geometrie benötigt. In dieser mithilfe des CAD-Programms erstellten Datei wird die Oberfläche der Geometrie anhand von Dreiecken beschrieben.

Schritt 2: Über diese Geometrie wird mittels „blockMesh“ ein erstes Hintergrundgitter gelegt. Mithilfe der „blockMesh“-Funktion kann ein Quader erstellt werden, welcher in mehrere kubische oder quaderförmige Zellen eingeteilt wird. In „blockMeshDict“, dem zum Ausführen von „blockMesh“ verwendeten Textfile, werden die Koordinaten der Eckpunkte des Quaders und die Anzahl der sich dazwischen befindenden Zellen festgelegt.

Schritt 3: Die „snappyHexMesh“-Funktion verfeinert nun sämtliche die .stl-Oberfläche umgebenden Zellen.
Wie oft diese Zellen verfeinert werden sollen, wird in „snappyHexMeshDict“ festgelegt.

Schritt 4: Anschließend werden entweder alle innerhalb oder alle außerhalb des Betrachtungsraums liegenden Zellen entfernt und die restlichen Zellen weiter verfeinert.

Schritt 5: In einem letzten Schritt werden die Kanten und Flächen geglättet. Außerdem können zusätzliche Schichten (engl. „layers“) eingeführt werden, um den Übergang zwischen den Zellen sowie der Geometrieoberfläche zu verfeinern und so die Auflösung zu erhöhen.

Die Schritte 3 bis 5 werden alle in einem Vorgang während der Ausführung von „snappyHexMesh“ durchgeführt und im Ordner „system“ in „snappyHexMeshDict“ genau definiert.

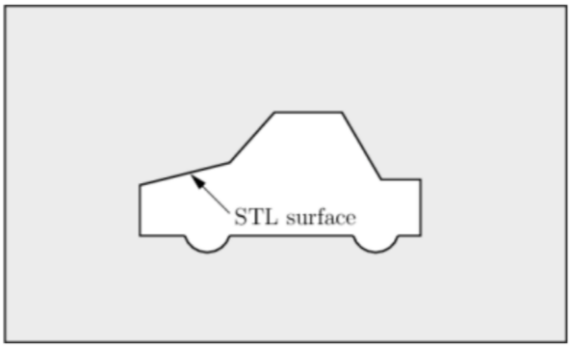



Abbildung 14: Schritt 1, vorhandene stl-Datei

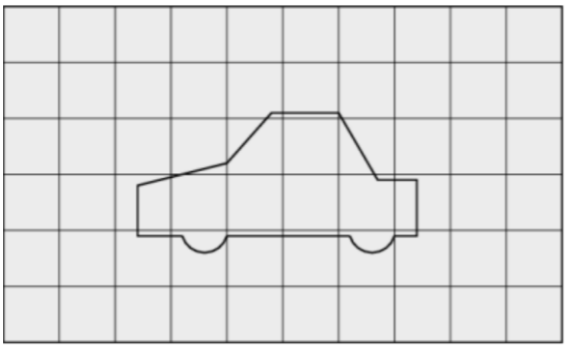


Abbildung 15: Schritt 2, Erzeugung eines „blockMesh“-Blocks (Hintergrundgitter)

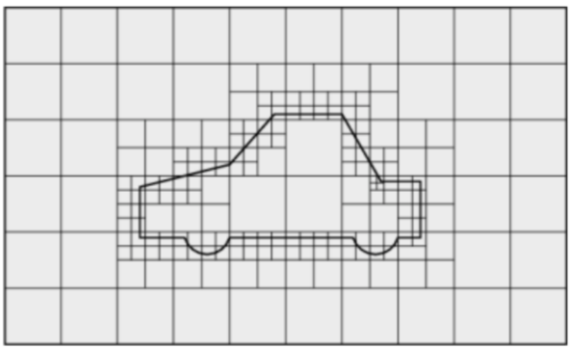


Abbildung 16: Schritt 3, Verfeinerung des Gitternetzes mithilfe der „snappyHexMesh“-Funktion

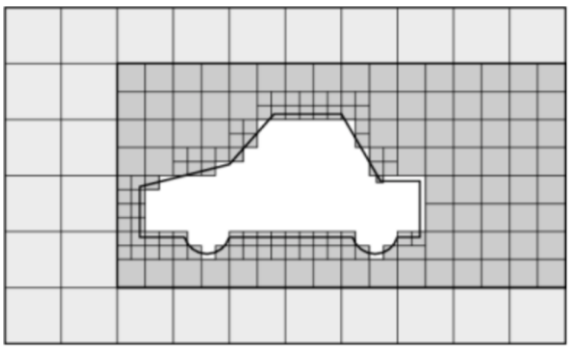


Abbildung 17: Schritt 4, Entfernung von innerhalb oder außerhalb des Betrachtungsraum liegenden Zellen

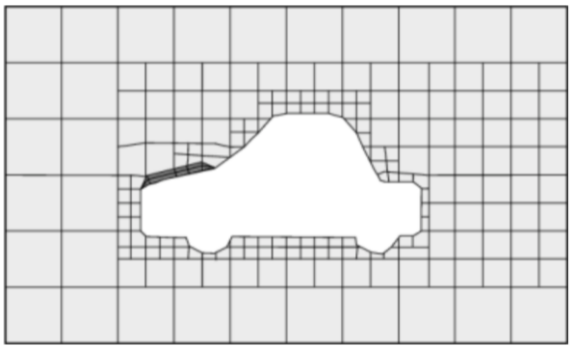


Abbildung 18: Schritt 5, Glättung und Verfeinerung des Gitternetzes

In den nachfolgenden Abschnitten wird das an dieser Stelle übersichtsmäßig dargestellte Vorgehen genauer beschrieben.

# 4. Erstes Rechengitter – „blockMesh“

Bevor mit dem Erstellen eines ersten Rechengitters begonnen wird, muss überprüft werden, ob die Geometrie die richtige Skalierung besitzt. Beim Erstellen der .stl-Datei bzw. beim Exportieren sollte darauf geachtet werden, mit welchen Einheiten die Geometrie exportiert wird. OpenFoam® arbeitet mit SI-Einheiten, daher werden alle Geometrieabmessungen in Metern übernommen. Sollte die Geometrie beispielsweise in Millimetern exportiert worden sein, kann eine Transformation in OpenFoam® vorgenommen werden. Dies kann mit dem Befehl >transformPoints durchgeführt werden:

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

Mithilfe von „blockMesh“ wird wie zuvor beschrieben ein Hintergrundgitter über die Geometrie gelegt. Zum Ausführen des Befehls „blockMesh“ werden das dazugehörige Textfile „blockMeshDict“ sowie das Textfile „controlDict“ benötigt. Beide werden im Ordner „system “ gespeichert und üblicherweise von anderen Simulationsvorlagen übernommen sowie anschließend an den jeweiligen neuen „case“ angepasst. Mit folgendem Befehl kann ein Textfile in einen neuen Ordner kopiert werden:

>cp Ort-wo-sich-die-Datei-befindet Zielordner/

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

Im „blockMeshDict“-Textfile werden die Koordinaten der Eckpunkte des Quaders und die Anzahl der sich dazwischen befindenden Zellen festgelegt. Nachfolgende Abbildung 19 zeigt das „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
(
);

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

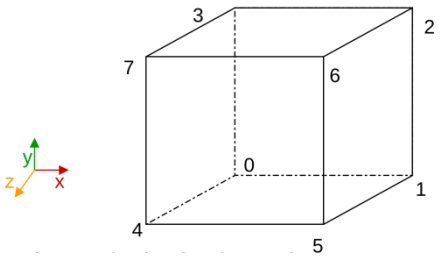



Abbildung 19: Textfile „blockMeshDict“ (links), Beschreibung der Koordinatenpunkte (rechts) [11]

Ab Zeile 19 bis Zeile 28 sind die Eckpunkte („vertices“) des Rechtecks, im Format X, Y, Z angegeben. In Zeile 33 sind folgende Informationen enthalten:

- hex (0 1 2 3 4 5 6 7) Definition des hexaedrischen Blocks. Auf die Reihenfolge achten!
- (30 10 10) Anzahl der Zellen zwischen den Eckpunkten
- „simpleGrading“ (1 1 1) Skalierung der Zellen zwischen den Eckpunkten

Es gibt zwei Möglichkeiten, einen passenden Block über die Geometrie zu legen. Entweder sind die Eckkoordinaten bekannt und diese werden im „blockMeshDict“-File eingetragen oder die Koordinaten sind unbekannt und müssen ermittelt werden. Falls die Koordinaten unbekannt sind, können sie mittels ParaView ermittelt werden. Dabei werden die .stl-Bauteilgeometrie und der erzeugte „blockMesh“-Block in ParaView visualisiert und die jeweiligen Werte abgelesen. Um den „blockMesh“-Block anzeigen lassen zu können, müssen folgende Punkte erfüllt sein:

- „blockMeshDict“-Textfile im Ordner „system“ vorhanden
- „controlDict“-Textfile im Ordner „system“ vorhanden
- „blockMesh“ im Terminal ausgeführt
  - >blockMesh
- Dummyfile „foam.foam“ im „case“-Ordner vorhanden

Um ein Dummyfile im „case“-Ordner anzulegen, muss folgender Befehl ausgeführt werden:

>nano foam.foam

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

Daraufhin öffnet sich der Texteditor und ein leeres Textfile erscheint. Mit
Strg + o
wird das File gespeichert und mit
Strg + x
wird der Editor wieder geschlossen. Daraufhin kann ParaView mit folgendem Befehl geöffnet werden:
>paraview
In ParaView müssen danach die stl-Geometriedaten und das Dummyfile „foam.foam“ geöffnet werden:
File > Open > foam.foam
Falls die Datei nicht sofort angezeigt wird, muss das Öffnen noch mit „apply“ bestätig werden.

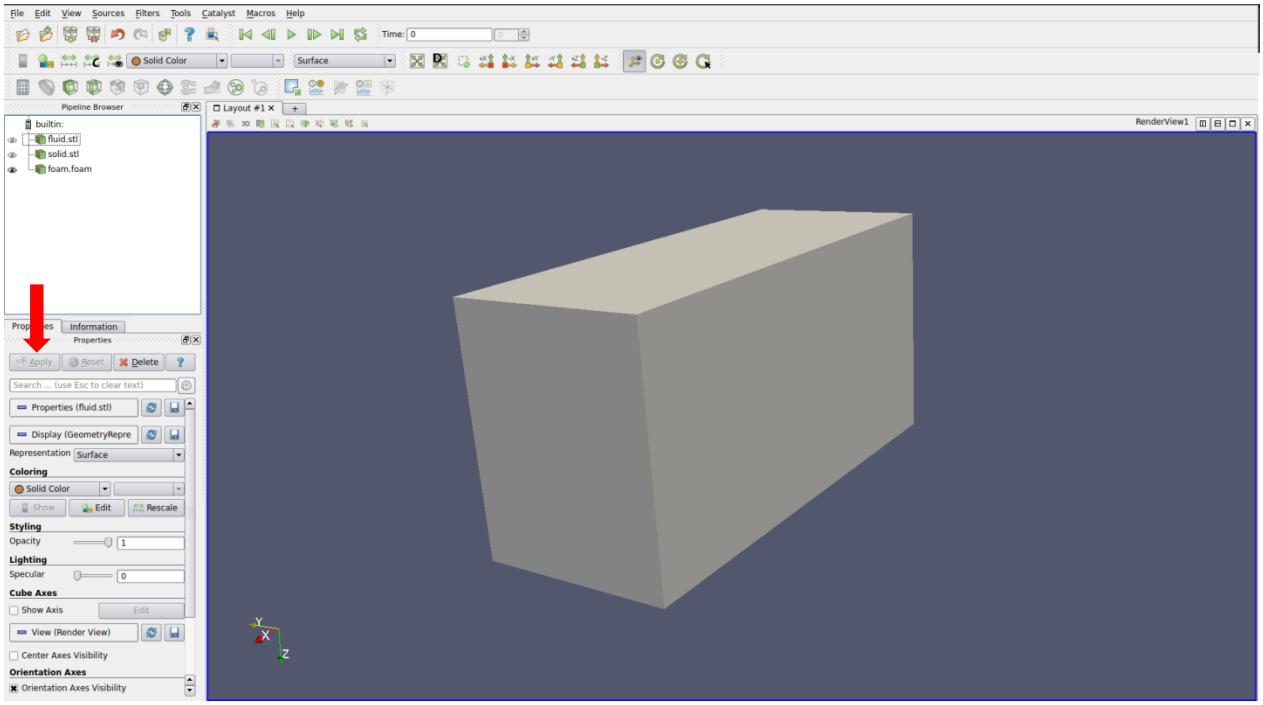


Abbildung 20: Erzeugter Block in ParaView

Im linken, unteren Menübereich werden unter „Informationen“ die Eckpunktkoordinaten der ausgewählten Geometrie angezeigt. In Abbildung 21 sind die Eckpunktkoordinaten der „solid.stl“-Geometrie veranschaulicht.

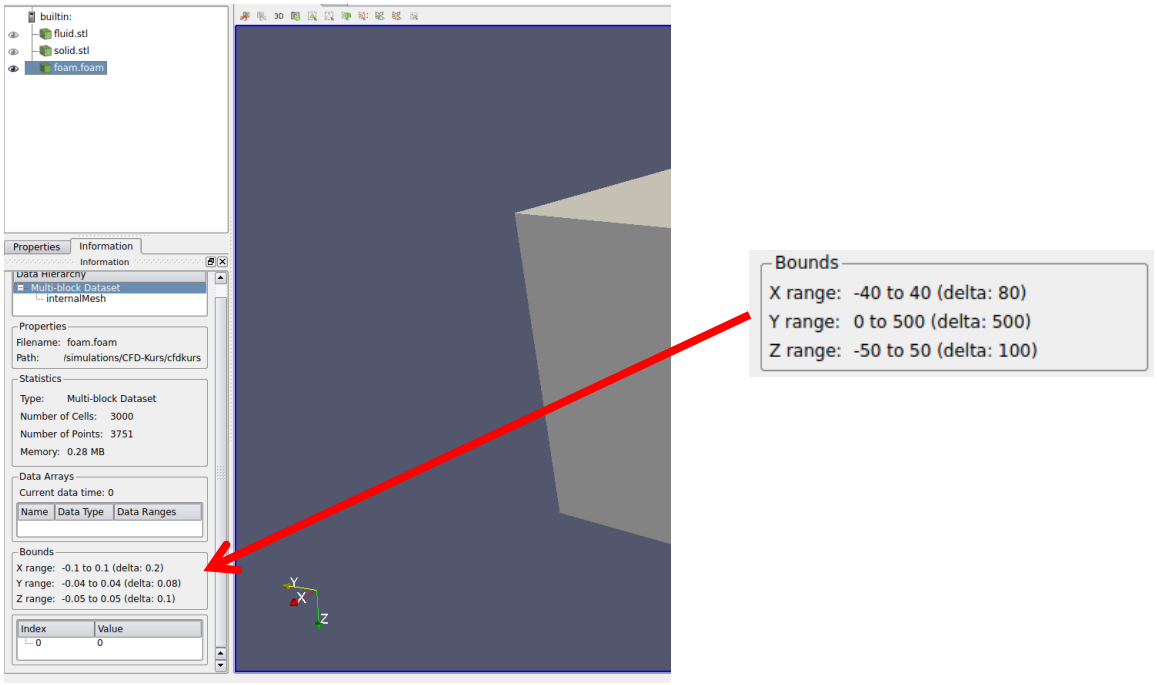



Abbildung 21: Eckpunktkoordinaten der „solid.stl“-Geometrie

Nun müssen die Koordinaten der Simulationsgeometrie, welche aus ParaView abgelesen werden, in das Textfile „blockMeshDict“ eingetragen werden. Dabei sollte der „blockMesh“-Block immer etwas größer als die Simulationsgeometrie gewählt werden. Im gezeigten Beispiel ist der blockMesh-Block um 5 Einheiten größer als die Simulationsgeometrie. Der erzeugte Block besitzt somit folgende Abmessungen:

```
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 );
```

Abbildung 22: Eckpunktkoordinaten des erzeugten „blockMesh“-Blocks, eingetragen im „blockMeshDict“-File

Nachdem die Eckpunktkoordinaten in das „blockMeshDict“-File eingetragen wurden, muss die Anzeige in ParaView mit „refresh“ noch aktualisiert werden. Daraufhin erscheint der zuvor definierte Block im Ansichtsbereich (Abbildung 23).

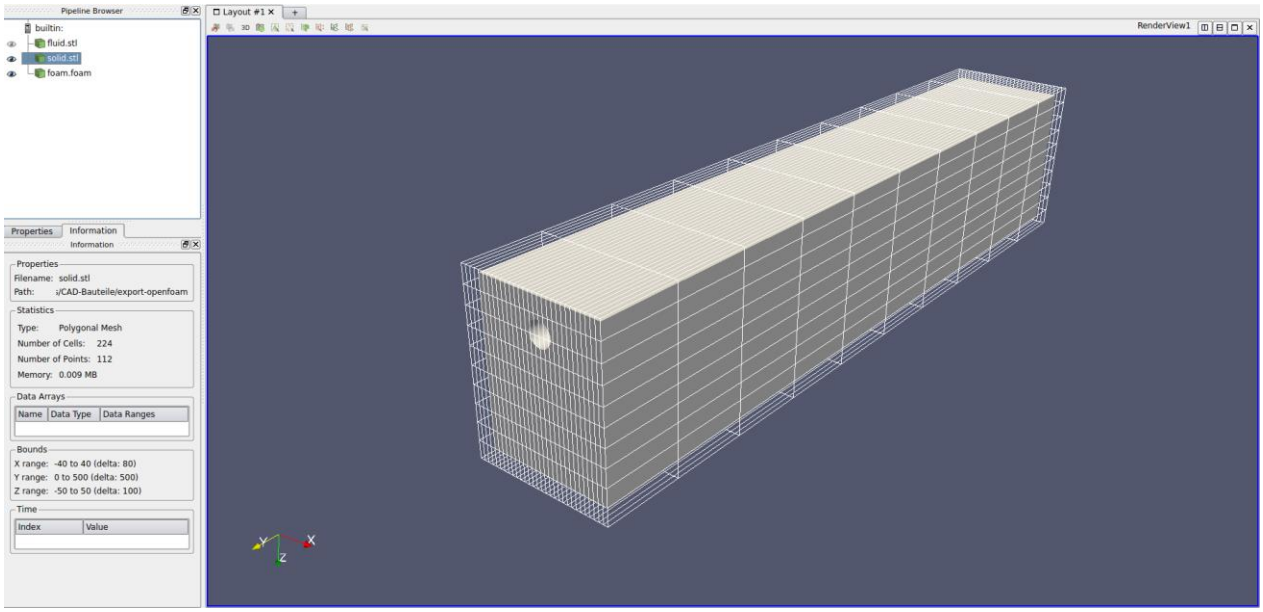



Abbildung 23: Simulationsgeometrie mit dem BlockMesh-Block (Gitter) darüber

Die Form der Rechtecke sollte optimalerweise würfelförmig sein. Dies kann durch Anpassen der Aufteilung im „blockMeshDict“-Textfile vorgenommen werden. Dazu muss das Textfile „blockMeshDict“ geöffnet werden und die Zeile 33 „(30 10 10) – Anzahl der Zellen zwischen den Eckpunkten“ geändert werden. Durch erneutes Ausführen des „blockMesh“-Befehls werden die eingestellten Daten übernommen:

>blockMesh

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

# 5. Zweites Rechengitter – „snappyHexMesh“

Mit dem Programm „snappyHexMesh“ werden anschließend die Zellen verfeinert, welche die stl-Geometrie umschließen. Damit die Verfeinerung mittels „snappyHexMesh“ durchgeführt werden kann, müssen zunächst folgende Dateien in den Ordner „system“ kopiert werden:

- „system“
  - „snappyHexMeshDict“
  - „meshQualityDict“
  - „surfaceFeatureExtractDict“
  - „fvSchemes“
  - „fvSolution“

Des Weiteren müssen sich die stl.-Dateien im Ordner „constant“ befinden:

- „constant“
  - Ordner „triSurface“ (muss gegebenenfalls erstellt werden)
    - „solid.stl“
    - „fluid.stl“

Falls der Ordner „triSurface“ nicht vorhanden ist, kann dieser ohne Weiteres erstellt werden. Möglich ist dies über folgenden Befehl:

```
>mkdir triSurface
```

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

Anschließend kann mit der Anpassung der Textdateien begonnen werden. Der Inhalt der Textdatei „snappyHexMeshDict“ ist in den folgenden Abbildungen dargestellt:

```
/*--------------------------------*- 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.
```

Abbildung 24: Inhalt snappyHexMeshDict-Datei, Zeile 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 // ************************************************************************* //
```

Abbildung 25: Inhalt „snappyHexMeshDict“-Datei, Zeile 233-336

Darin müssen zunächst die Geometriedateien angepasst werden. Hierbei müssen unter „geometry“ (Zeile 30) die verwendeten .stl-Geometrien angegeben werden. Im durchgeführten Beispiel zählen dazu folgende .stl-Dateien:

- „solid.stl“
- „fluid.stl“

Die restlichen Einträge unter „geometry“ können gelöscht werden. Des Weiteren müssen die Einträge bei „features“ (Zeile 103) analog zur Änderung bei „geometry“ angepasst werden.

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

    fluid.stl
    {
        type triSurfaceMesh;
        name fluid;
    }
};
```

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

Im Abschnitt „refinementSurfaces“ (Zeile 138) müssen ebenfalls Anpassungen vorgenommen werden. Dabei sind die Namen wiederum in „solid“ und „fluid“ umzubenennen. In den Einträgen „faceZone“ und „cellZone“ sind die Namen der stl-Dateien einzutragen. Darüber hinaus müssen Koordinaten eines Punktes angegeben werden, der sich im Inneren der zu vernetzenden Geometrie befindet. Dies ist für „solid“ und „fluid“ einzutragen. In der Zeile 201 befindet sich der Eintrag „locationInMesh“, wo wiederum ein Punkt im Inneren der zu vernetzenden Geometrie anzugeben ist. Die Angabe eines Punktes

im Inneren einer Geometrie, dient der Definition des Vernetzungsstarts bzw. ob die innere oder die äußere Geometrie vernetzt werden soll.
In der nachfolgenden Abbildung 26 ist der Inhalt der „meshQualityDict“-Datei dargestellt. Diese wird ebenfalls für die Simulation auch benötig, es sind jedoch keine Änderungen erforderlich.

```
/*--------------------------------*- 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      meshQualityDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Include defaults parameters from master dictionary
#include "$WM_PROJECT_DIR/etc/caseDicts/meshQualityDict"

//#includeEtc "caseDicts/meshQualityDict"


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

Abbildung 26: Inhalt „meshQualityDict“-Datei

Im Textfile „surfaceFeatureExtractDict“ (Abbildung 27) sind wiederum die Namen der stl-Dateien einzutragen (Zeile 17 und 33).

```
/*--------------------------------*- 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      surfaceFeatureExtractDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solid.stl
{
    extractionMethod    extractFromSurface;

    extractFromSurfaceCoeffs
    {
        // Mark edges whose adjacent surface normals are at an angle less
        // than includedAngle as features
        // - 0  : selects no edges
        // - 180: selects all edges
        includedAngle   150;
    }

    // Write options
    writeFeatureEdgeMesh    yes;
}
fluid.stl
{
    extractionMethod    extractFromSurface;

    extractFromSurfaceCoeffs
    {
        // Mark edges whose adjacent surface normals are at an angle less
        // than includedAngle as features
        // - 0  : selects no edges
        // - 180: selects all edges
        includedAngle   150;
    }

    // Write options
    writeFeatureEdgeMesh    yes;
}
// ************************************************************************* //
```

Abbildung 27: Inhalt „surfaceFeatureExtractDict“-File

Nachdem alle Textdateien angepasst wurden, muss noch der Befehl

>surfaceFeatureExtract

Beispiel:

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

ausgeführt werden. Damit werden die Dateien

- „solid.eMesh“
- „fluid.eMesh“

im Ordner „triSurface“ generiert. Danach kann der Befehl

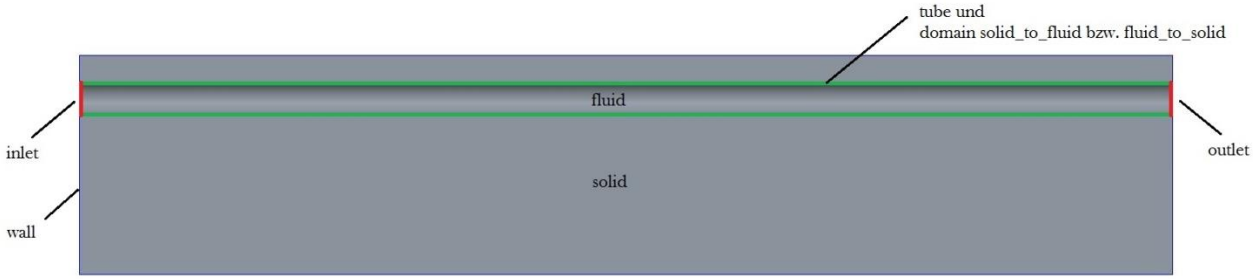
>snappyHexMesh -overwrite

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

ausgeführt werden. Der Befehl „overwrite“ wird verwendet, da ansonsten ein neuer Ordner erstellt wird, falls bereits ein Gitter („mesh“) existiert.

# 6. Erstellung und Zuordnung der Regionen

Da in diesem Beispiel der Wärmeübergang von einem flüssigen zu einem festen Stoff simuliert wird, müssen die einzelnen Geometrien eingeteilt werden. Dies dient dazu, dem Programm mitzuteilen, um welche Geometrie bzw. um welchen Volumenkörper es sich bei einem Feststoff oder einer Flüssigkeit handelt. Zudem sollen den jeweiligen Stoffen im Anschluss angepasste Materialparameter zugeteilt




Mit Blender wurden in einem vorherigen Schritt die Geometrien einzeln als .stl-Files exportiert. Dabei setzt sich die Geometrie „solid.stl“ aus „wall.stl“ und „tube.stl“ zusammen. Die Geometrie „fluid.stl“ setzt sich aus „tube.stl“, „inlet.stl“ und „outlet.stl“ zusammen. Somit müssen insgesamt 7 einzelne .stl-Dateien vorhanden sein. Je nach Simulationssetup kann es sein, dass die Geometrie „tube.stl“ nicht benötigt wird. In Abbildung 28 sind die einzelnen stl-Geometrien bildlich dargestellt.

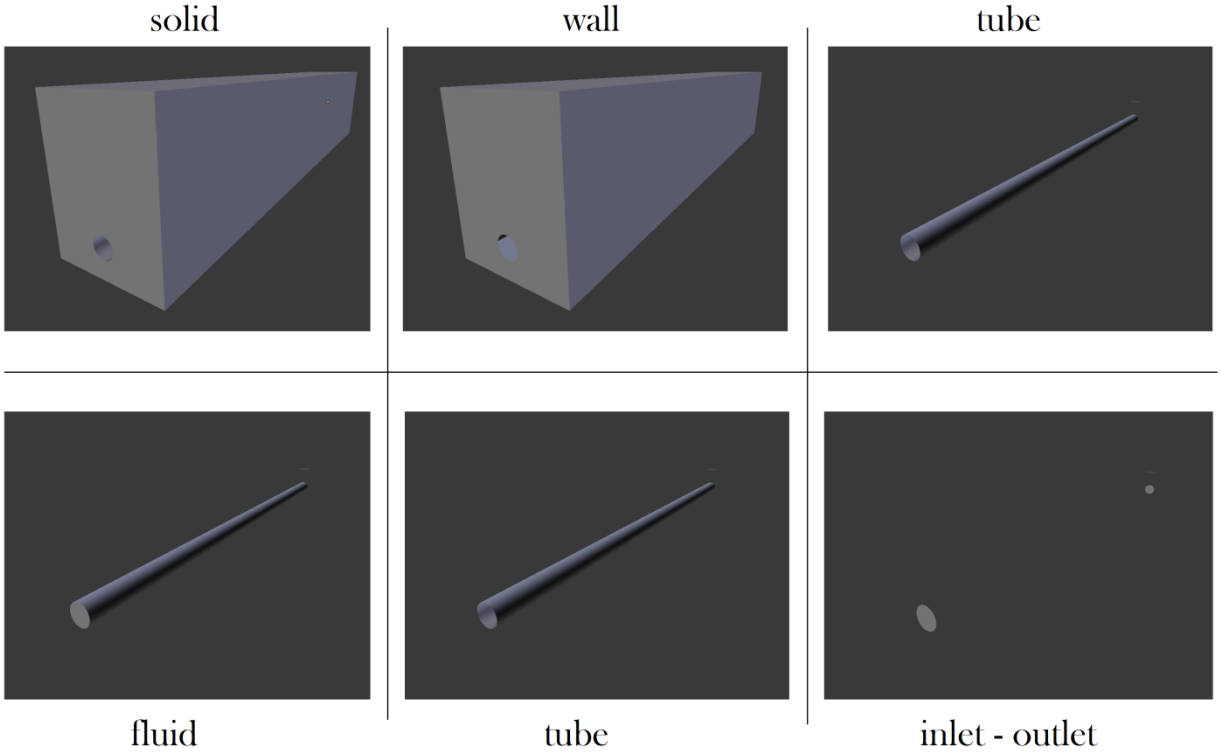



Abbildung 28: Veranschaulichung der einzelnen stl-Dateien

Die einzelnen stl-Dateien müssen bestimmten Regionen zugeordnet werden. Eine Auflistung der stl-Dateien mit den dazugehörigen Regionen ist in Tabelle 2 dargestellt.

Tabelle 2: Einteilung der stl-Dateien in Regionen

| stl-Datei | Region |
|---|---|
| „tube.stl“ | „fluid“ |
| „inlet.stl“ | „fluid“ |
| „outlet.stl“ | „fluid“ |
| „wall.stl“ | „solid“ |

Bevor diese zugeteilt werden, müssen zunächst die Zonen („fluid“ und „solid“) aufgeteilt werden. Dies erfolgt mit folgendem Befehl:

>splitMeshRegions -cellZones -overwrite

Beispiel:
~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$ splitMeshRegions -cellZones -overwrite

Danach müssen die einzelnen stl-Dateien bzw. Flächen den Regionen zugeordnet werden. Dies erfolgt mit dem Befehl:

>surfaceToPatch „Ort der stl-Datei“ -region „fluid oder solid“

Beispiel:
~/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

Dabei kann es vorkommen, dass folgende Fehlermeldung erscheint (Abbildung 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
```

Abbildung 29: Fehlermeldung bei Eingabe des Befehls „surfaceToPatch constant/triSurface/inlet.stl -region fluid“

In diesem Fall muss ein zusätzlicher Schritt durchgeführt werden bzw. sind die Regionen separat zuzuordnen. Dies bedeutet, dass der „surfaceToPatch“-Befehl in einem separat angelegten Ordner ausgeführt werden muss. Dies dient dazu, die Regionen separat voneinander zuordnen („patchen“) zu können. Aus diesem Grund werden Dummy-Ordnerstrukturen angelegt („fluid > constant > polyMesh“), da OpenFoam® die Ordnerstruktur ansonsten nicht erkennt. Würde lediglich eine Region vorhanden sein, also nur „solid“ oder „fluid“, könnte der „surfaceToPatch“-Befehl direkt im Hauptordner (im Beispiel „cfdkurs “) ausgeführt werden.
Da im vorliegenden Beispiel jedoch zwei Regionen vorhanden sind, müssen folgende Schritte ausgeführt werden:

1. Zuerst müssen im Hauptordner die Ordner „solid“ und „fluid“ erstellt werden.
2. In diesen Ordnern muss dann jeweils ein „constant“-Ordner angelegt werden. Ein Ordner wird mit dem Befehl >mkdir Ordnername angelegt.
3. In diesen „constant“-Ordner muss danach der „polyMesh“-Ordner vom Hauptordner (~/OpenFoam/OpenFoam-3.0.1/simulations/CFD-Kurs/cfdkurs/constant) kopiert werden.
4. Des Weiteren muss im jeweiligen Ordner („solid“ und „fluid“) der Ordner „system“ angelegt werden und in diesen die Datei „controlDict“ kopiert werden. Diese ist im Hauptordner („cfdkurs“) im „system“-Ordner zu finden.
5. Abschließend müssen noch die dazugehörigen .stl-Dateien in die erstellten Ordner „fluid“ und „solid“ kopiert werden.

Nach dem Anlegen und Kopieren der Dateien muss die Ordnerstruktur wie folgt aussehen:

**„cfdkurs“**

- **„constant“**
  - „polyMesh“
  - „fluid“
    - „polyMesh“
  - „solid“
    - „polyMesh“
- **solid**
  - „constant“
    - „polyMesh“

    - „system“
        - „controlDict“
    - „wall.stl“
- **„fluid"**
    - „constant“
        - „polyMesh“
    - „system“
        - „controlDict“
    - „inlet.stl“
    - „outlet.stl“

Im Verzeichnis „constant > solid“ dürfen nur die .stl-Dateien der äußeren Wände enthalten sein. Das heißt zum Beispiel, die Wand eines inneren Rohres darf nicht enthalten sein. Nachdem die Ordnerstruktur angelegt worden ist, muss zum angelegten Ordner navigiert werden („solid“ oder „fluid“) und für jede enthaltene .stl-Datei der „surfaceToPatch“-Befehl ausgeführt werden:

>surfaceToPatch CAD-Datei.stl

Beispiel:
~/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.

Zwischen dem Ausführen dieses Befehls für jede einzelne .stl-Datei muss der „patch0“-Eintrag in der „polyMesh-boundary“-Datei geändert werden. Nachfolgend sind die einzelnen Vorgänge Schritt für Schritt dargestellt:
Beispiel für „fluid“:

„cfdkurs > fluid > 0.1“

- Ersten Zeitschrittordner (0.1) öffnen (nicht 0)
- „polyMesh“-Ordner öffnen
- „boundary“-File öffnen
- „patch0“ in „inlet“ umbenennen (bzw. mit dem Namen versehen, der bei „surfaceToPatch“ verwendet wurde)
- >surfaceToPatch outlet.stl im Ordner „cfdkurs > fluid“ ausführen
- „cfdkurs > fluid > 0.2“ öffnen
- „polyMesh“-Ordner kopieren und in „cfdkurs > constant > fluid“ einfügen
- Danach kann der gesamte Ordner „cfdkurs > fluid“ als Sicherung abgelegt oder gelöscht werden

Danach müssen noch folgende Einträge im „boundary“-File gelöscht werden und die Zahl in Reihe 18 (Abbildung 30) von 4 auf 3 geändert werden, da sich die Anzahl der angegebenen Patches geändert hat. Wird dies nicht geändert, führt dies zu einer Fehlermeldung beim späteren Anzeigen der Ergebnisse in ParaView. Die Fehlerfreiheit des erzeugten Gitters kann mit folgenden Befehl (beispielsweise für die Region „solid“) überprüft werden:

> checkMesh -region solid

Beispiel:
~/OpenFOAM/OpenFOAM-3.0.1/simulations/CFD-Kurs/cfdkurs$ checkMesh -region 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       polyBoundaryMesh;
    location    "0.02/polyMesh";
    object      boundary;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

4
(
    fluid_to_solid
    {
        type            mappedWall;
        inGroups
2
(
wall
mappedPatch
)
;
        nFaces          4800;
        startFace       16768;
        sampleMode      nearestPatchFace;
        sampleRegion    solid;
        samplePatch     solid_to_fluid;
    }
    fluid_to_domain0
    {
        type            mappedWall;
        inGroups
2
(
wall
mappedPatch
)
;
        nFaces          0;
        startFace       21568;
        sampleMode      nearestPatchFace;
        sampleRegion    domain0;
        samplePatch     domain0_to_fluid;
    }
    inlet
    {
        type            patch;
        nFaces          32;
        startFace       21568;
    }
    outlet
```

Abbildung 30: boundary-File des „fluids“

Um den Regionen nun ihre Eigenschaften und Anfangsbedingungen zuweisen zu können, wird in „constant“ ein „regionProperties“-File benötigt. Darin müssen folgende Einträge vorhanden sein:

```
/*--------------------------------*- 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)
);


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

Abbildung 31: „regionProperties“-File im Ordner „constant“

Wurden diese Schritte durchgeführt, kann mit der Eingabe der Anfangs- und Randbedingungen begonnen werden.

# 7. Solving – Simulation mit „chtMultiRegion“

Bei dem ausgewählten Simulationsfall wird der Aufheiz- und Abkühlvorgang betrachtet. Beim Aufheizvorgang strömt Wasser mit einer Temperatur von 90°C 10 Sekunden lang durch das Rohr. Darauffolgend beginnt der Abkühlvorgang, bei dem Wasser mit einer Temperatur von 30°C 10 Sekunden lang durch das Rohr strömt. Zusätzlich dazu werden die Simulationen mit zwei verschiedenen Reynoldszahlen bzw. Strömungsgeschwindigkeiten durchgeführt, um den Wärmeübergang bei einer laminaren und turbulenten Strömung betrachten zu können.

## 7.1. Theorie Strömungssimulation

Für die Simulation der Strömung im Kanal wurde das Standard-k-ε-Turbulenzmodell verwendet. Es handelt sich um ein Zwei-Gleichungsmodell, das als meistverwendetes Modell zur Beschreibung von Strömungen in CFD-Simulationen gilt. Dabei werden in der Transportgleichung die Produktion und die Diffusion für turbulente kinetische Energie „k“ angenähert.

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

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

Durch verschiedene Annahmen (z.B. Druck-Geschwindigkeitskorrelation) und Approximationen (z.B. Boussinesq-Ansatz) lauten die Gleichungen des Standard-k-ε-Modells schließlich [12]:

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

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

| | |
|---|---|
| $P_k$ | Produktion |
| $D_k$ | Diffusion |
| $\nu_T$ | Wirbelviskosität |
| $\bar{S}_{ij}$ | viskoser Spannungstensor |
| $x_j$ | Ortskoordinate |
| $\bar{u}_j$ | mittlere Geschwindigkeit |
| $k$ | turbulente kinetische Energie |
| $\varepsilon$ | isotrope Dissipationsrate |
| $C_\mu$ | Modellkonstante |
| $c_{\varepsilon 1}$ | Modellkonstante |
| $c_{\varepsilon 2}$ | Modellkonstante |
| $\sigma_k$ | Modellkonstante |
| $\sigma_\varepsilon$ | Modellkonstante |

Typischerweise werden die Modellkonstanten folgendermaßen definiert: $C_\mu$ = 0,09; $c_{\varepsilon 1}$ = 1,44; $c_{\varepsilon 2}$ = 1,92; $\sigma_k$ = 1,0 und $\sigma_\varepsilon$ = 1,3.

Um yPlus berechnen zu können werden turbulente Anfangswerte benötigt, welche wie folgt berechnet werden [13]:

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

dabei ist k die turbulente kinetische Energie und I die turbulente Intensität, welche mit 0,8 angenommen wird. $u_{ref}$ ist die Geschwindigkeit,

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

ε ist die turbulente Dissipationsrate, $C_\mu$ eine Konstante mit dem Wert 0.09 und L die Länge des Kanals.

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

$\nu_t$ ist dabei die turbulente Viskosität.
Aus den veranschaulichten Formeln ergeben sich für das Simulationsbeispiel folgende Werte:

k =0.273
ε =1.563
ν = 0.00324

## 7.2. Input-Parameter

Um die Simulation fertig aufzusetzen, werden nachfolgend die Eingangsparameter für die laminare und die turbulente Simulation angeführt. Für die Geschwindigkeit beim Einlass (Anfangsbedingung) wurde in beiden Fällen folgende Formel verwendet:

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

Dabei ist Re die Reynoldszahl in [1], u die Geschwindigkeit in [m/s], D der Durchmesser des durchströmten Kanals in [m] und ν (nu) die kinematische Viskosität in [m²/s]. Die kinematische Viskosität von Wasser bei verschiedenen Temperaturen wurde folgender Tabelle entnommen:

Tabelle 3: Stoffparameter von Wasser bei verschiedenen Temperaturen [14]

| $\vartheta$ °C | $\rho$ kg m$^{-3}$ | $c_p$ kJ kg$^{-1}$ K$^{-1}$ | $\beta$ $10^{-3}$ K$^{-1}$ | $\eta$ $10^{-6}$ kg m$^{-1}$ s$^{-1}$ | $\nu$ $10^{-6}$ m$^2$ s$^{-1}$ | $\sigma$ $10^{-3}$ 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 |

Der Druck des durchströmenden Wassers beträgt 5 bar ($5 \cdot 10^5$ Pa). Der durchströmte Metallbock besitzt zu Beginn der Simulation eine konstante Temperatur von 30°C.

### 7.2.1. Eingabeparameter Laminar

| | | |
|---|---|---|
| Reynoldszahl | Re = | 500 |
| Kanaldurchmesser | D = | 0,015 m |
| Kinematische Viskosität, Wasser | $\nu$ = | 0,801 *$10^{-6}$ m²/s bei 30°C |
| | | 0,326 *$10^{-6}$ m²/s bei 90°C |

Daraus ergeben sich folgende Geschwindigkeiten:

| | | |
|---|---|---|
| Strömungsgeschwindigkeit | u = | 0,0267 m/s bei 30°C |
| | | 0,011 m/s bei 90°C |

### 7.2.2. Eingabeparameter Turbulent

| | | |
|---|---|---|
| Reynoldszahl | Re = | 10.000 |
| Kanaldurchmesser | D = | 0,015 m |
| Kinematische Viskosität, Wasser | $\nu$ = | 0,801 *$10^{-6}$ m²/s bei 30°C |
| | | 0,326 *$10^{-6}$ m²/s bei 90°C |

Daraus ergeben sich folgende Geschwindigkeiten:

| | | |
|---|---|---|
| Strömungsgeschwindigkeit | u = | 0,533 m/s bei 30°C |
| | | 0,217 m/s bei 90°C |

Diese Eingabeparameter bzw. Anfangsbedingungen werden in Dokumenten definiert, welche sich im Ordner „0>fluid“ befinden. Darin sind die Dateien „alphat“, „epsilon“, „k“, „nut“, „p“, „p_rgh“, „T“ und „U“ vorhanden. Im Ordner „0>solid“ sind „p“ und „T“ vorhanden.

```
/*--------------------------------*- 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;
    }

}

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

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

| | 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 |

Um das k-ε-Modell anwenden zu können, muss der „yPlus“-Wert, welcher eine Aussage über die wandnächste Zellhöhe liefert, einen Wert zwischen 20 und 100 annehmen. Überprüft wird der Wert mit folgendem Befehl:

> yPlus -region fluid

Damit dieser Befehl ausgeführt werden kann, müssen die folgende Ordnerstruktur sowie darin die angeführten Files enthalten sein:

**0**

- „fluid“
  - „alphat“
  - „epsilon“
  - „k“
  - „nut"
  - „p“
  - „p_rgh“

    - „T“
    - „U“
- „solid“
    - „p“
    - „T“

Die Screenshots der Textdateien sind im Anhang in den Abschnitten 10.1. bis 10.2. zu sehen.
Bei den „fluid“-Files müssen bei „epsilon“, „k“, „nut“, „p“, „p_rgh“, „T“ und „U“ unter dem Eintrag „boundaryFields“ die Namen der Geometrien angepasst werden. Im Simulationsbeispiel sind somit „inlet“, „outlet“ und „fluid_to_solid“ einzutragen. Bei den „solid-Files“ „p“ und „T“ sind „wall“ und „solid_to_fluid“ einzutragen. Des Weiteren sind die Anfangsbedingungen anzupassen, zu denen die Geschwindigkeit [m/s], der Druck [Pa] und die Temperatur [K] zählen.

Im Ordner „constant“ müssen folgende Ordner und Files vorhanden sein:

„constant“
- „fluid“
    - „polyMesh“ (Ordner)
    - „fvSchemes“
    - „fvSolution“
    - „g“
    - „thermophysicalProperties“
    - „transportProperties“
    - „turbulenceProperties“
- „solid“
    - „polyMesh“ (Ordner)
    - „fvSchemes“
    - „fvSolution“
    - „thermophysicalProperties“

Die Screenshots der Textdateien sind im Anhang in den Abschnitten 10.3. bis 10.4. zu sehen.

Nachdem alle Anpassungen vorgenommen wurden, kann der „yPlus“-Befehl schließlich für „solid“ und „fluid“ ausgeführt werden:

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

Da der „yPlus“-Wert im veranschaulichten Beispiel über 100 lag, wurden die Zellen am „fluid-solid“-Grenzbereich mit folgendem Befehl verfeinert:

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

(keine „–region“ Option muss deshalb im Ordner ausgeführt werden, in dem „inlet“ und „outlet“ aufgepatcht wurden)

Nach dem Verfeinern muss der erzeugte „polyMesh“-Ordner nach „constant > fluid“ kopiert werden. Durch den refineWallLayer-Befehl wurden zusätzliche Zellreihen eingefügt, woraufhin sich „yPlus“ auf einen Wert von durchschnittlich 54.74426 verringert. In der Abbildung 32 ist das verfeinerte Mesh dargestellt:

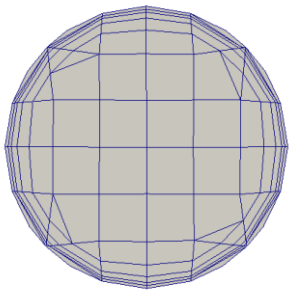


Abbildung 32: Verfeinerte Vernetzung an der Kühlrohr-Geometrie

Für den Festkörper („solid“) wird als Anfangstemperatur ein Wert von 30°C definiert. Des Weiteren sollte die Randbedingung an der äußeren Wand, durch die Änderung des Eintrags unter „externalHeatFluxTemperature“ definiert werden. Dabei wird festgelegt, dass beim Wärmeübergang Festkörper und Umgebung ein konstanter Wärmeübergangskoeffizient von 10 W/m²K verwendet wird. Das heißt, die Umgebung bzw. Luft wird in der Simulation nicht abgebildet, sondern nur mit der genannten, konstanten Randbedingung berücksichtigt.

# 8. Ausführen der Simulation

Um mit der Simulation starten zu können, müssen die Rahmenbedingungen definiert werden. Diese sind im Ordner „constant“ zu finden. Nachfolgende Eingaben sollten definiert werden:

- Materialkenndaten für „fluid“ und „solid“ werden in „thermophysicalProperties“ angegeben. Diese Files befinden sich im Ordner „constant“. Darin enthalten sind Werte wie die Wärmeleitfähigkeit oder -kapazität, welche für Stahl und Wasser gebräuchlich sind.

- Für die turbulente Simulation wird ein File namens „turbulenceProperties“ benötigt, in dem das verwendete Turbulenzmodell definiert ist.

- Die numerischen Einstellungen, welche in „fvSchemes“ und „fvSolution“ zu finden sind, wurden standardmäßig angenommen.

- In „controlDict“ wurde angegeben, dass der erste Part der Simulation (=Aufheizvorgang) 10 Sekunden dauert und dabei alle 0,1 s gespeichert wird.

- Um einen insgesamt 20 Sekunden dauernden Aufheiz- und Abkühlprozess simulieren zu können, wurde nach 10-sekündiger Aufheizphase die Einlasstemperatur und -geschwindigkeit in den Files „T“ und „U“ im Ordner 10 (Simulationszeitschrittordner) angepasst, um schließlich auch die Abkühlphase darstellen zu können.

Die Screenshots der Textdateien sind dem Anhang 10. zu entnehmen. Schlussendlich wird mit

>chtMultiRegionFoam

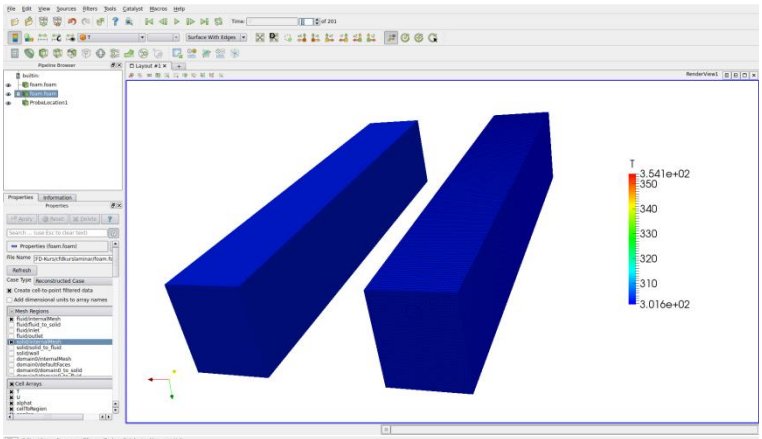
die Simulation gestartet. Nachdem die turbulente Simulation abgeschlossen ist, kann auch die laminare Simulation erstellt werden. Hierfür wird eine Kopie des gesamten turbulenten „case“ („cfdkurs“) erstellt („cfdkurslaminar“). Um mit diesem kopierten „case“ eine laminare Simulation durchführen zu können, müssen folgende Änderungen vorgenommen werden:

- Die Geschwindigkeit im Ordner „0 > U“ wird reduziert, um eine laminare Strömung abzubilden.
- Im File „turbulenceProperties“ im Ordner „constant“ wird die Strömungsart auf laminar geändert.

# 9. Ergebnisse

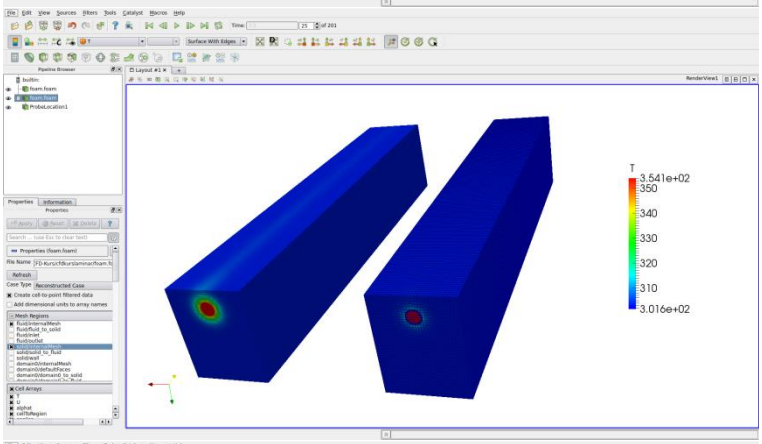
## 9.1. Aufheiz- und Abkühlvorgang

Nach der Simulation wurde der Aufheiz- und Abkühlvorgang in ParaView dargestellt. In den nachfolgenden Abbildungen sind die Ergebnisse in Schritten von 2,5 Sekunden grafisch dargestellt. Dabei ist der turbulente Simulationsfall links und der laminare rechts dargestellt.



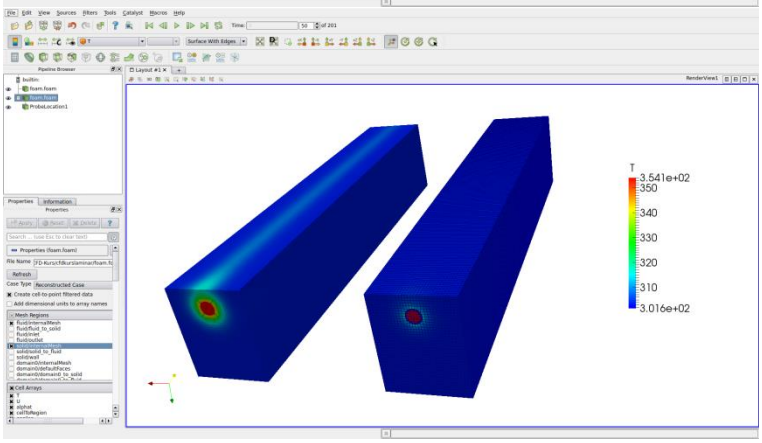

Turbulent (links)
Laminar (rechts)

Ausgangszustand
Konstant 30°C
t = 0 Sekunden



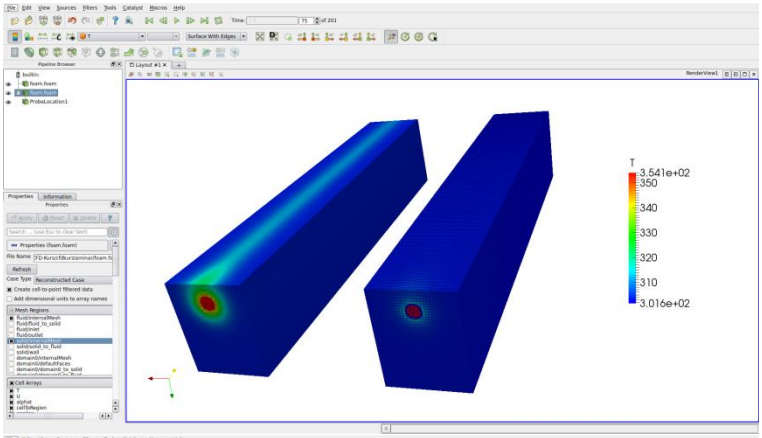

Turbulent (links)
Laminar (rechts)

Aufheizvorgang
t = 2,5 Sekunden



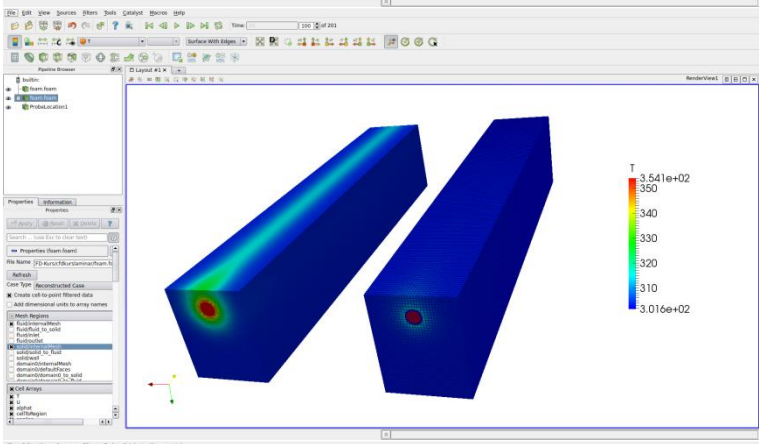

Turbulent (links)
Laminar (rechts)

Aufheizvorgang
t = 5 Sekunden



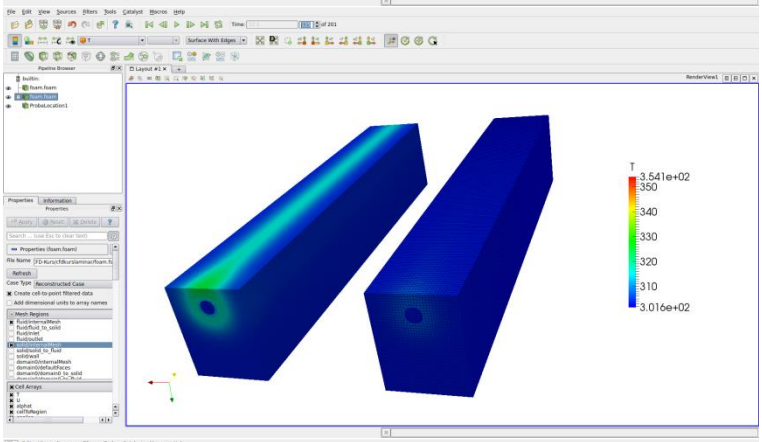

Turbulent (links)
Laminar (rechts)

Aufheizvorgang
t = 7,5 Sekunden



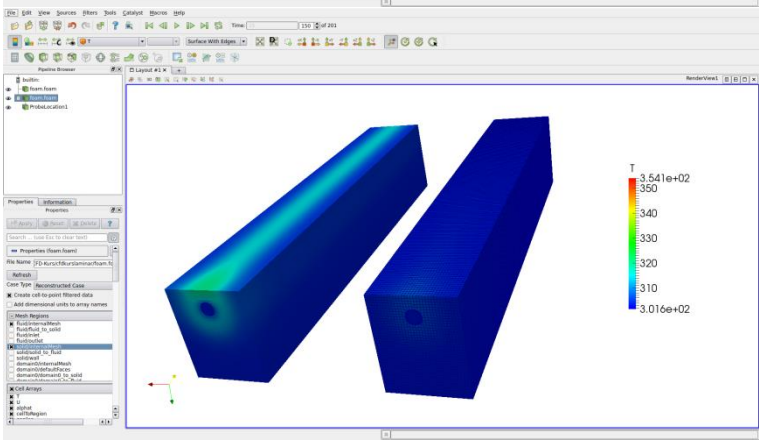

Turbulent (links)
Laminar (rechts)

Ende Aufheizvorgang und
Start der Abkühlung
t = 10 Sekunden



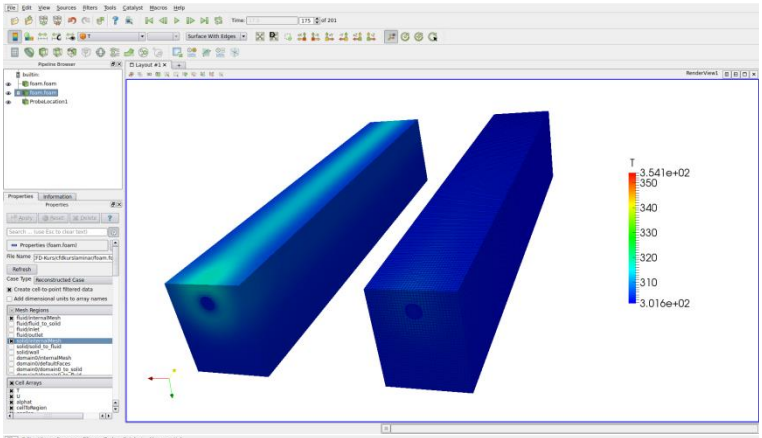

Turbulent (links)
Laminar (rechts)

Abkühlvorgang
t = 12,5 Sekunden



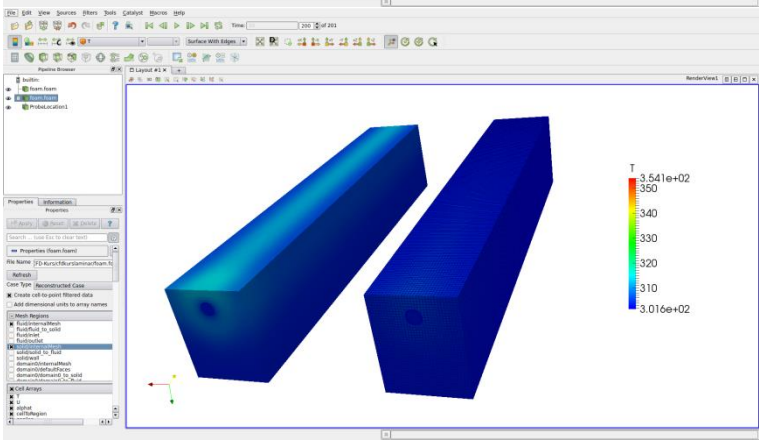

Turbulent (links)
Laminar (rechts)

Abkühlvorgang
t = 15 Sekunden




Turbulent (links)
Laminar (rechts)

Abkühlvorgang
t = 17,5 Sekunden




Turbulent (links)
Laminar (rechts)

Ende Abkühlung
t = 20 Sekunden

Beim Betrachten dieser Abbildungen wird ersichtlich, dass sich die turbulente Variante deutlich dynamischer auf die Temperaturänderung verhält. Um diesen Umstand zu verdeutlichen, können Diagramme von Messpunkten an der Oberfläche erstellt werden. Der Export von Simulationsdaten wird im nächsten Unterabschnitt 0genauer beschrieben. Für den Export wurden zwei Punkte an der Oberfläche definiert (Abbildung 34).

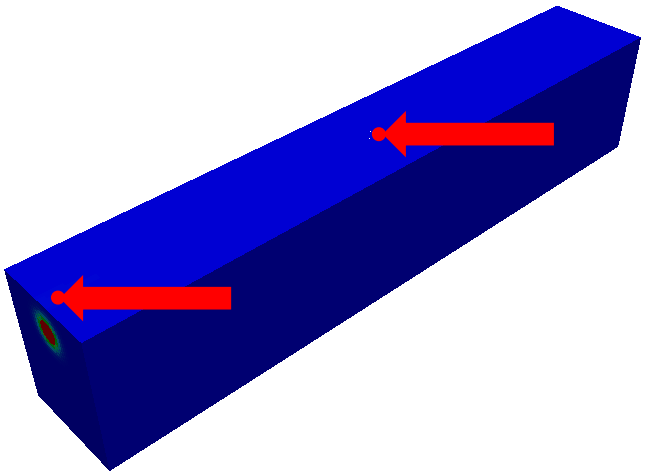


Abbildung 33: Definierte Messpunkte an der Oberfläche der simulierten Geometrie

Der erste Messpunkt befindet sich 10 mm hinter dem Einlass. Der zweite Messpunkt wurde genau in der Mitte des Blocks platziert und befindet sich deshalb 250 mm vom Einlass entfernt. Für diese Punkte werden anschließend die Messwerte generiert und in Diagrammen dargestellt (Abbildung 34 und

Abbildung 35). In der Abbildung 34 ist der Temperaturverlauf bei einer Position von 10 mm hinter dem Einlass für den turbulenten und laminaren Fall dargestellt. Die Strömungsgeschwindigkeiten betragen im turbulenten Fall 0,533 m/s bei 30°C und 0,217 m/s bei 90°C. Beim laminaren Fall beträgt die Strömungsgeschwindigkeit 0,0267 m/s bei 30°C und 0,011 m/s bei 90°C.

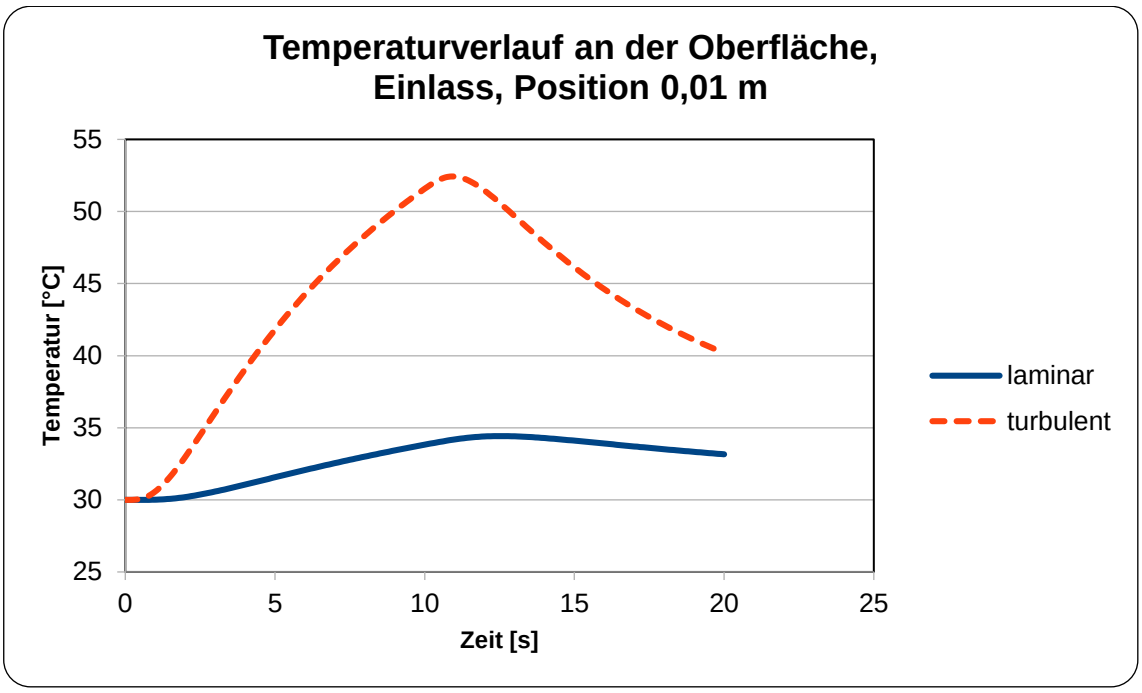



Abbildung 34: Temperaturverlauf an der Oberfläche, Einlass, Position 0,01 m

Erkennbar ist, dass sich der turbulente Fall deutlich dynamischer verhält als der laminare. Dies ist durch die Erhöhung des Wärmeübergangskoeffizienten zwischen Festkörper und Fluid bei hohen Reynolds-Zahlen begründet. In Abbildung 35 ist der Temperaturverlauf bei einer Position von 250 mm hinter dem Einlass dargestellt.

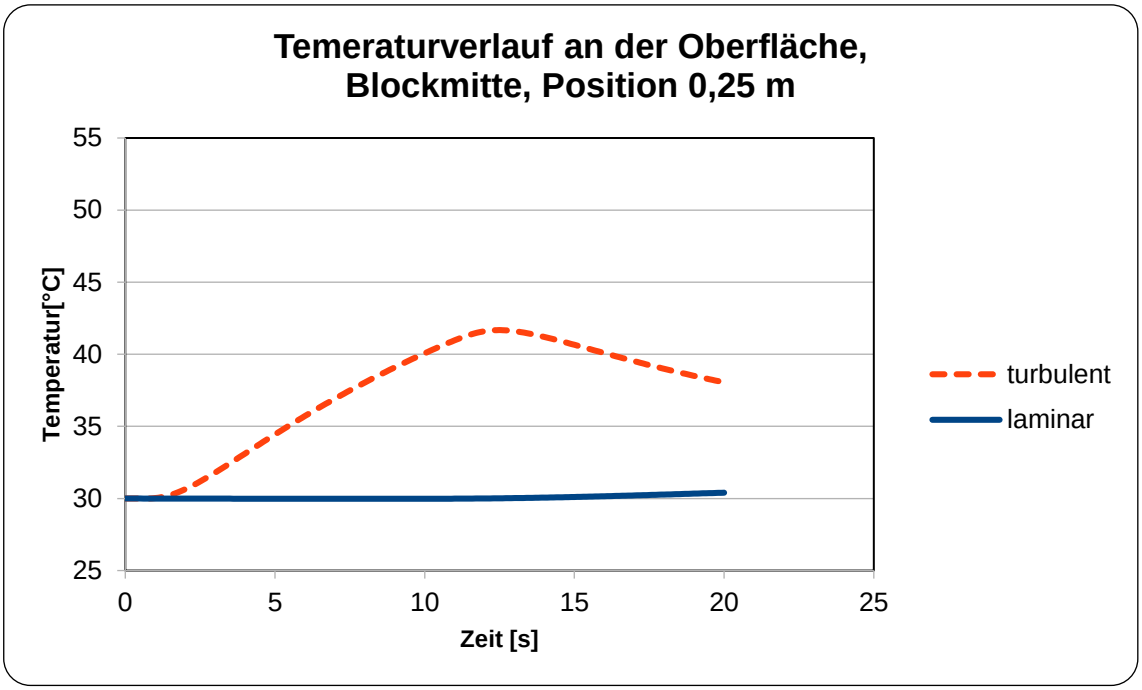



Abbildung 35: Temperaturverlauf an der Oberfläche, Blockmitte, Position 0,25 m

Dabei ist zu erkennen, dass der Temperaturanstieg im Vergleich zum Messpunkt bei 10 mm um ca. 10°C niedriger ausfällt. Beim laminaren Simulationsfall findet kein Temperaturanstieg statt, da die Strömungsgeschwindigkeit so gering ist, dass das warme Wasser erst mitten im Abkühlvorgang beim Messpunkt bei 250 mm ankommt.

## 9.2. Export der Messwerte

Nachdem die Geometrie in ParaView angezeigt wird, kann mit der Funktion „probeLocation“ (Abbildung 36) ein Messpunkt auf der Oberfläche der simulierten Geometrie erzeugt werden.

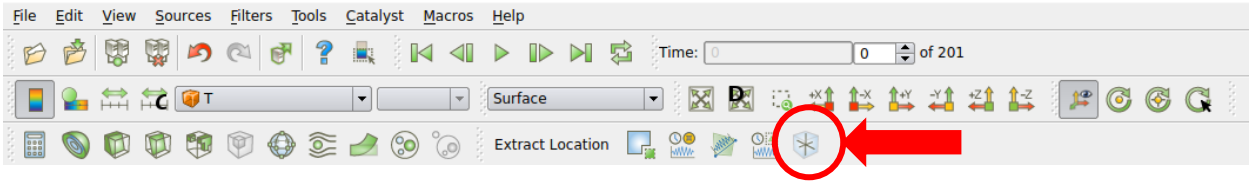



Abbildung 36: ParaView Funktion „probeLocation“

Mit dem Befehl unter „File > Save Data“ können danach ausgewählte Daten für die Zeitschritte ausgegeben werden. Im gezeigten Beispiel werden die Temperaturdaten ausgegeben. Der Nachteil beim Export der Daten auf diese Weise ist, dass für jeden Zeitschritt eine Excel-Datei erstellt wird. Diese können jedoch anschließend im Ordner, in dem die Files abgespeichert sind, mit dem Befehl

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

zu einer einzigen Excel-Datei zusammengeführt werden. Mit diesem Befehl werden jedoch auch die Überschriften der 200 Dateien zusammengeführt, sodass jede Messwertzeile eine überflüssige Überschriftenzeile besitzt. Um diese zu löschen, müssen folgende Befehle ausgeführt werden:

- Öffnen der Datei
  >vim erstelltecsvdatei.csv
- Start der Makroaufzeichnung
  Zweifaches Betätigen der q-Taste
- Zeile löschen
  Zweifaches Betätigen der d-Taste
  Pfeil nach unten, um Zeile zu überspringen
- Makroaufzeichnung beenden
  Einfaches Betätigen der q-Taste
- Befehl mehrfach eine bestimmte Anzahl ausführen
  Anzahl der Ausführungen am Nummernblock eingeben
  Mit @q Ausführung starten
- Speichern und Beenden
  :wq (write and quite)

ParaView besitzt umfangreiche Möglichkeiten zur Auswertung verschiedenster Simulationsfälle. Die hier veranschaulichten Beispiele stellen nur sehr kleine Teile der vorhandenen Möglichkeiten dar. Weitere Informationen zu den Auswertungsmethoden und Tutorials für die Bedienung von ParaView können direkt der Website von ParaView [15] entnommen werden.

# 10. Anhang

## 10.1. Ordner „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. Ordner „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. Ordner „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((nuEff*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. Ordner „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. Literatur


[1] „CAESES Software › CAESES“. .

[2] E. S. I. Group, „Visual-CFD for OpenFOAM“, *ESI Group*, 10-Juni-2016. [Online]. Verfügbar unter: https://www.esi-group.com/software-solutions/virtual-environment/cfd-multiphysics/visual-cfd-openfoam. [Zugegriffen: 03-Dez-2018].

[3] „SimFlow CFD Software - OpenFOAM® GUI“, *simFlow CFD*. [Online]. Verfügbar unter: https://sim-flow.com/. [Zugegriffen: 03-Dez-2018].

[4] „MantiumFlow, OpenFOAM CFD Simulations, download demo and tutorials“, *MantiumFlow, CFD Simulation Software with OpenFOAM*. [Online]. Verfügbar unter: https://mantiumflow.com/. [Zugegriffen: 03-Dez-2018].

[5] „OpenFOAM® Documentation“. [Online]. Verfügbar unter: https://www.openfoam.com/documentation/. [Zugegriffen: 03-Dez-2018].

[6] P. Geisser, *Temperiertechnik - Theorie und Praxis*, 3. Aufl. 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 und 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 Universität Linz, 29-Mai-2015.

[10] OpenCFD, „Mesh generation with the snappyHexMesh utility“. [Online]. Verfügbar unter: http://www.openfoam.com/documentation/user-guide/snappyHexMesh.php. [Zugegriffen: 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-Juni-2016.

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

[13] „OpenFOAM: k-epsilon“. [Online]. Verfügbar unter: http://www.openfoam.com/documentation/cpp-guide/html/guide-turbulence-ras-k-epsilon.html. [Zugegriffen: 27-Sep-2017].

[14] „Stoffwerte von Wasser“. [Online]. Verfügbar unter: http://www.uni-magdeburg.de/isut/LSS/Lehre/Arbeitsheft/IV.pdf. [Zugegriffen: 26-Sep-2017].

[15] „ParaView“. .