Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Untersuchung der Möglichkeiten zum Einsatz von Open Source CFD Software für Strömungssimulationen im Bereich der Gebäudetechnik

Masterarbeit

Zur Erlangung des akademischen Grades
"Master of Engineering (M. Eng.)"

von

Mike Dahncke
Matr. Nr.: 1878648

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät: Technik und Informatik
Department: Maschinenbau und Produktion

Erstprüfer: Prof. Dr.-Ing. Peter Wulf

21. August 2012

Imtech Deutschland GmbH & Co. KG
Forschung und Entwicklung

Betrieblicher Betreuer und Zweitprüfer: Dr.-Ing. Bruno Lüdemann

Mike Dahncke

Untersuchung der Möglichkeiten zum Einsatz von Open Source CFD Software für Strömungssimulationen im Bereich der Gebäudetechnik

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Master of Engineering
der Vertiefungsrichtung Nachhaltige Energiesysteme im Maschinenbau
am Department Maschinenbau und Produktion
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

In Zusammenarbeit mir der Firma Imtech Deutschland GmbH & Co. KG
Abteilung: Forschung und Entwicklung

Betreuender Prüfer: Prof. Dr.-Ing. Peter Wulf
Zweitgutachter: Dr.-Ing. Bruno Lüdemann

21. August 2012

Aufgabenstellung Masterarbeit im Bereich Strömungssimulation bei Imtech Forschung und Entwicklung

Untersuchung der Möglichkeiten zum Einsatz von Open Source CFD Software für Strömungssimulationen im Bereich der Gebäudetechnik

In der Forschung und Entwicklung bei Imtech Deutschland GmbH & Co. KG werden bereits seit dem Jahr 1989 Methoden der CFD (**C**omputational **F**luid **D**ynamics) eingesetzt. Die Anwendungsbereiche der CFD beinhalten Raumströmungen, Außenumströmungen, Rauchausbreitung, Entrauchung, Klimatisierung, Schadstoffbelastung und Kombinationen dieser Anwendungen (HVAC - Heating Cooling Ventilation AirConditioning).
Ziel dieser Masterarbeit ist die Untersuchung der Open Source CFD-Software openFOAM hinsichtlich ihrer Anwendbarkeit in den oben genannten Bereichen der CFD. Dazu sollen mehrere grundlegende Strömungsformen in einfachen Simulationsmodellen dargestellt und mit den Ergebnissen aus ANSYS Fluent verglichen werden. Abschließend soll ein Modell höherer Komplexität mit den Möglichkeiten der untersuchten Tools erstellt und die Ergebnisse mit Ergebnissen von ANSYS Fluent verglichen werden.
Es soll ein Überblick über die in den untersuchten Programmen vorhandenen Modelle und Möglichkeiten erarbeitet, sowie der Aspekt der Bedienbarkeit und der Anwendbarkeit der Programme auf komplexe Fragestellungen mit vielen Randbedingungen und verschiedenen physikalischen Phänomenen herausgearbeitet werden.

Die Masterarbeit teilt sich in folgende Aufgaben auf:

- Einarbeitung in die Open Source CFD Anwendung OpenFOAM, das zugehörige Vernetzungstool snappyHexMesh (bzw. nach Absprache mit Herrn Prof. Dr.-Ing. Wulf) und ParaView zur Datenvisualisierung.
- Simulation und Untersuchung grundlegender Strömungsformen aus Anwendungen des HVAC Bereichs:
    1. Durchströmung eines Kanalbogens, Vergleich der Ergebnisse aus ANSYS Fluent und OpenFoam sowie mit Literaturwerten.
    2. Freie Konvektion, Wärmeübergang an einer senkrechten Platte.
    3. Freie Turbulenz und Rauchschichtung am Beispiel eines thermisch induzierten Auftriebsstrahls. Vergleich mit ANSYS Fluent Simulationsergebnissen und Literaturwerten.
- Zur Untersuchung der Anwendbarkeit der o. g. Software auf komplexe Fragestellungen, sollen am Beispiel eines generischen Treppenhauses der Einfluss des thermischen Auftriebs und der bei aufwärts gerichteter Durchströmung des Treppenhauses entstehende Druckverlust untersucht und mit Ergebnissen aus ANSYS Fluent verglichen werden.

Kurzzusammenfassung

Bei der Imtech Deutschland GmbH & Co KG werden bereits seit 1989 Methoden der Numerischen Strömungssimulation (engl. CFD) eingesetzt. Für die Simulationen wird bei Imtech das kommerzielle CFD-Software-Paket von ANSYS verwendet. Für die Nutzung dieser Software fallen jährliche Lizenzgebühren in erheblichem Umfang an. Diese Lizenzgebühren können durch die Verwendung quelloffener Software (Open Source Software) vermieden werden.
Das Ziel dieser Masterarbeit ist die Untersuchung der Open Source CFD-Software OpenFOAM hinsichtlich ihrer Anwendbarkeit für Strömungssimulationen im Bereich der Gebäudetechnik.
Hierzu wird im ersten Teil eine geeignete Open Source CFD Programmkette ausgewählt. Anschließend werden drei grundlegende Strömungsformen aus dem Bereich der Gebäudetechnik simuliert und die Simulationsergebnisse mittels geeigneter Vergleichsdaten validiert. Weiterhin werden die in den ausgewählten Programmen vorhandenen Modelle und Möglichkeiten herausgearbeitet, sowie deren Bedienbarkeit und Anwendbarkeit bewertet. Um die Anwendbarkeit der ausgewählten Programme auf komplexe Fragestellungen mit vielen Randbedingungen zu untersuchen wird im zweiten Teil am Beispiel eines generischen Treppenhauses der Einfluss des thermischen Auftriebs und der bei aufwärts gerichteter Durchströmung entstehende Druckverlust untersucht.
Abschließend werden die Teil-Ergebnisse zusammengefasst, ein Fazit gezogen und ein Ausblick auf die weitere Verwendung von Open Source CFD Software im Bereich der Gebäudetechnik gegeben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Formelzeichen, Symbole, Indizes und Abkürzungen

Arabische Zeichen

A	m^2	Querschnittsfläche
A_{AB}	m^2	Abluftfläche
A_{ZU}	m^2	Zuluftfläche
AR	$[-]$	Archimedeszahl
D	m	Durchmesser
D_B	m	Brandherddurchmesser
D_h	m	Hydraulischer Durchmesser
H	m	Höhe
I	Ns	Impuls
I_t	$\%$	Turbulenzintensität
L_B	m	Bezugslänge
Nu	$[-]$	Nusseltzahl
P	W	elektrische Leistung
Pr	$[-]$	Prandtlzahl
$\dot{Q}_{ges}$	W	Gesamtwärmestrom
$\dot{Q}_K$	W	konvektiver Wärmestrom
R	m	Radius
Ra	$[-]$	Rayleighzahl
Re	$[-]$	Reynoldszahl
T	K	Temperatur
U_B	m/s	Bezugsgeschwindigkeit
U_B	m/s	Auftriebsgschwindigkeit
b_{AB}	m	Breite der Absaugöffnung
c	m	Geschwindigkeit
cp	$J/(kgK)$	spez. Wärmekapazität
g	m/s^2	Erdbeschleunigung
h	m	Höhe
h	J/kg	spezifische Enthalpie
h_{SG}	m	Höhe der Schichtgrenze
k	m^2/s^2	kinetische Energie der Schwankungsbewegung
k	mm	Rauhigkeitserhebung
l	m	Länge
$\dot{m}_{AB}$	kg/s	Absaugmassenstrom
$\dot{m}_{AS}$	kg/s	Auftriebsstrahlmassenstrom
m	kg	Masse

n	$[-]$	Koordinate normal Stromlinie
p	Pa	Druck
p_rgh	Pa	statischer Druck $p\_rgh = p - \rho gh$
$\dot{q}$	W/m^2	spez. Wärmestrom
t	s	Zeit
u	m/s	Geschwindigkeit
u_x, u_y, u_z	m/s	Geschwindigkeitskomponenten
x, y, z	m	kartesische Koordinaten
y^+	$[-]$	dimensionsloser Wandabstand

Griechische Zeichen

α	m^2/s	Temperaturleitfähigkeit
α	$W/(m^2K)$	Wärmeübergangskoeffizient
β	K^{-1}	thermischer Ausdehnungskoeffizient
δ	m	Dicke der Strömungsgrenzschicht
ϕ	$[-]$	beliebige Strömungsvariable
ε	m^2/s	turbulente Diffusion
η	Pas	dynamische Viskosität
η_t	Pas	(dynamische) turbulente Viskosität
λ	$W/(mK)$	Wärmeleitfähigkeit
λ	$[-]$	Rohrreibungszahl
ν	m^2/s	kinematische Viskosität
ν_t	m^2/s	(kinematische) turbulente Viskosität
ϑ	$^\circ C$	Temperatur
ϑ_∞	$^\circ C$	Umgebungstemperatur
$\Delta\vartheta$	K	Temperaturdifferenz
ρ	kg/m^3	Dichte
τ	N/m^2	Spannung
ζ	$[-]$	Druckverlustbeiwert
ω	$1/s$	charakteristische turbulente Frequenz

Indizes

A	Absaugung
AB	Abluft

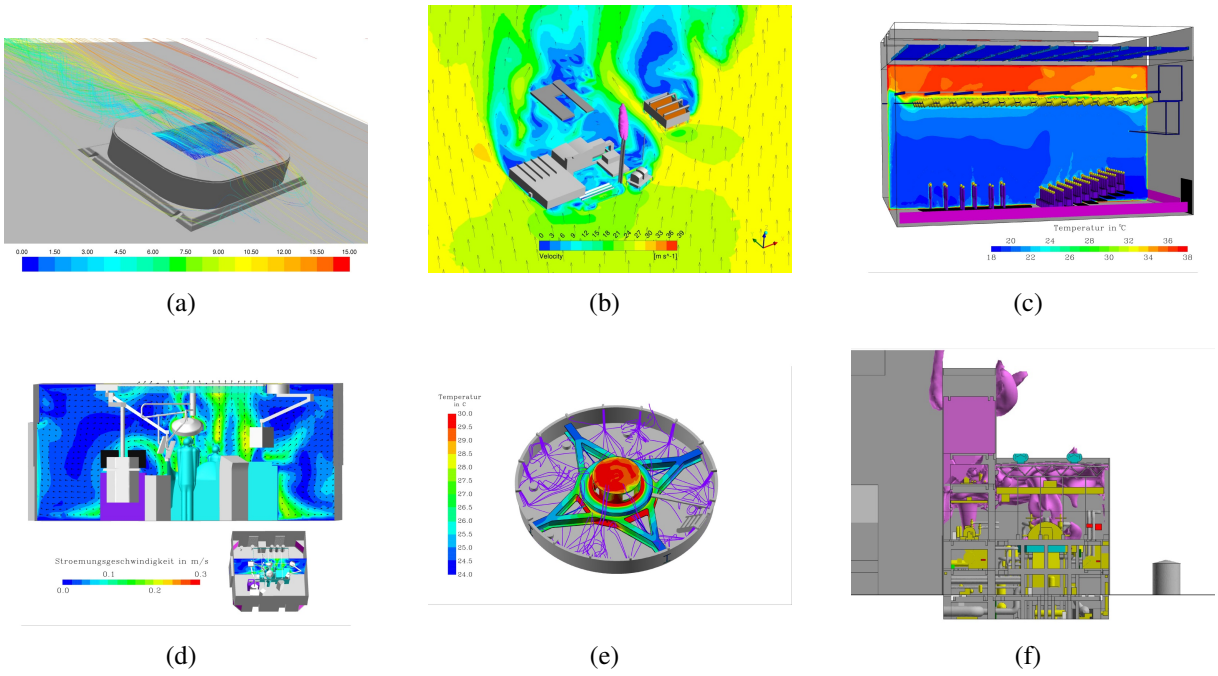
eff	effektive
fr	Reibung (engl. friction)
k	Krümmung
$krit$	kritische
loc	lokal
m	mittlere
max	maximale
t	turbulent
v	Verlust
x,y,z	Koordinatenrichtung vektorieller Größen
∞	Umgebung

Abkürzungen

$CFL-Zahl$	Courant-Friedrichs-Lewy-Zahl
FE	Finite Elemente
GUI	Graphical User Interface (Grafische Benutzeroberfläche)
FV	Finite Volumen
$NS-Gleichungen$	Navier Stokes Gleichungen
$RANSE$	Reynolds gemittelte Navier Stokes Gleichungen
STL	SurfaceTesselationLanguage (Beschreibung der Oberfläche durch Dreiecke)

1 Einleitung

In der Forschung und Entwicklung bei der Imtech Deutschland GmbH & Co KG werden bereits seit 1989 Methoden der Numerischen Strömungssimulation (engl. Computational Fluid Dynamics = CFD) eingesetzt. Die Anwendungsbereiche umfassen globale Simulationen externer Luftströmungen (Abb. 1.1 a) und b)), Berechnung von Außenluftinfiltrationen, Detailuntersuchungen von Raumluftströmungen (Abb. 1.1 c) und d)), Schadstofftransport in Räumen und in der Außenluftumgebung, numerische Untersuchungen von Heizungs- Lüftungs- und Klima-Komponenten (Abb. 1.1 e)) sowie die numerische Optimierung brandschutztechnischer Konzepte (Abb. 1.1 f)). Für die Strömungssimulationen stehen bei Imtech insgesamt 40 Prozessoren



(a) (b) (c)

(d) (e) (f)

Abb. 1.1: Beispiele für die Anwendung der Strömungssimulation bei Imtech: a) Anströmung einer Sportarena, b) Umströmung eines Müllheizkraftwerks, c) Temperaturverteilung in einer Studiobühne, d) Raumströmung in einem OP-Saal, e) Kühlung der Turbinenlagerstützen eines Pumpspeicherkraftwerks, f) Rauchschichtung mit Entrauchung im Maschinenhaus eines Steinkohlekraftwerks [Imtech]

und 320 GB Hauptspeicher zur Verfügung. Mit dieser Ausstattung ist es bisher gelungen Modellgrößen bis zu 410 Millionen Kontrollvolumina zu berechnen. Für die Simulationen wird bei Imtech das kommerzielle CFD-Software-Paket von ANSYS verwendet und für das Postprozessing kommt zusätzlich FieldView zum Einsatz. Für die Nutzung dieser kommerziellen Software fallen jährliche Lizenzgebühren in erheblichem Umfang an.
Diese Lizenzgebühren können durch die Verwendung quelloffener Software (Open Source Software) vermieden werden. Das ambitionierteste Projekt im CFD-Bereich stellt die OpenSource CFD-Software OpenFOAM (Open Field Operation and Manipulation) dar. Im Rahmen der vorliegenden Arbeit wird diese Software für die genannten Anwendungen der Strömungssimulation bei Imtech überprüft.

Dafür werden anfangs drei grundlegende Strömungsformen steigender Komplexität untersucht und mit experimentellen Daten, sowie mit ANSYS Fluent Ergebnissen verglichen. Untersucht werden die turbulente Durchströmung eines Kanalbogens, der freie Wärmeübergang an einer ebenen Wand und die freie Turbulenz und Rauchschichtung am Beispiel eines thermischen Auftriebstrahls. Für jeden Testfall werden die genutzten Modelle vorgestellt und die Bedienbarkeit sowie die Möglichkeiten und Grenzen der genutzten Programme aufgezeigt.
Um den Aspekt der Anwendbarkeit der Software auf komplexe Fragestellungen mit vielen Randbedingungen zu untersuchen wird abschließend das Modell eines generischen Treppenhauses aus der aktuellen Forschung bei Imtech untersucht und mit vorliegenden ANSYS Fluent Ergebnissen verglichen.
In Kapitel Zwei werden die Grundideen von Opensource Software und im Speziellen von OpenFOAM vorgestellt. Weiterhin werden die benötigten Grundlagen zur numerischen Strömungsmechanik zusammengefasst. In Kapitel Drei erfolgt eine Auswahl der in dieser Arbeit zu verwendenden Open Source CFD Programme. Kapitel Vier beschreibt die Simulation der grundlegenden Testfälle. In den Unterkapiteln werden die Grundlagen der Strömungsformen, die Untersuchungsmerkmale, die Vernetzung des Berechnungsgebietes, die Strömungsmodellerstellung und die Auswertung der Simulationen zusammenfassend behandelt. Die genutzten Modelle werden anschließend für die Simulation des Druckverlustes und des Einflusses des thermischen Auftriebs bei der aufwärts gerichteten Durchströmung eines generischen Treppenhauses im fünften Kapitel untersucht. Im sechsten Kapitel werden die Erkenntnisse zusammengefasst und ein Ausblick auf mögliche nachfolgende Arbeiten gegeben.

2 Grundlagen und Stand der Technik

2.1 Open-Source-CFD-Software

Im Rahmen dieser Masterarbeit wird das Open-Source-CFD Programm OpenFOAM verwendet. In diesem Kapitel wird die Software einführend vorgestellt. Dafür wird zunächst erklärt, was Open-Source-Software ist, wie sich diese entwickelt hat und was sie auszeichnet. Aber auch auf Stärken und Schwächen soll eingegangen werden. Anschließend wird OpenFOAM vorgestellt. Begonnen wird auch hier mit einem kurzen historischen Überblick und es werden charakteristische Merkmale, die Struktur und Möglichkeiten, sowie Vor- und Nachteile aufgezeigt.

2.1.1 Geschichte der Open-Source-Software [6]

Bis etwa zur Mitte des Jahres 1960 war Software grundsätzlich „Frei“ und wurde als kostenlose Beigabe zum neuen Rechner ausgeliefert. Die Hersteller verdienten dabei ausschließlich an der Computer-Hardware und die den Programmen zugrunde liegenden Quellcodes waren für jeden interessierten Nutzer der Welt frei zugänglich. Ab 1965 stellten dann etwa IBM dieses Verfahren ein. Die Hilfe außenstehender Entwickler wurde nicht mehr benötigt, da inzwischen genügend eigene Programmierer beschäftigt wurden. Spätestens zu Beginn der siebziger Jahre stellten viele Programmierer fest, dass sich mit der von ihnen entwickelten Software erhebliche Gewinne erzielen ließen. Zur Sicherung ihrer Einnahmequellen wurden Lizenzverträgen geschaffen, die die Weitergabe von Software von einem Nutzer an einen anderen einschränkten oder komplett verboten. Die Quellcodes wurden zu den bestgehütetsten Geheimnissen der Unternehmer auf dem IT-Markt.
Dadurch bedingt waren Computer-Anwender bei Programmfehlern oder Sonderwünschen seither auf das Entgegenkommen der Software-Produzenten angewiesen. Aus Unzufriedenheit mit dieser Entwicklung beschloss Richard Stallman[1] vom Massachusetts Institute of Technology (MIT) 1984, ein wieder freies Programmpaket namens GNU[2] zu entwickeln. Ziel des Computer-Spezialisten vom MIT war es, die offene Zusammenarbeit der Software-Entwickler, wie er sie selbst zu Beginn der Siebziger Jahre noch erlebt hatte, erneut zum Nutzen aller Computer-Anwender zu ermöglichen. Nach Ansicht von Stallman müssen alle Quellcodes vervielfältigt, verändert und weitergegeben werden können, denn frei ist nach Auffassung von Stallman eine Software nur dann, wenn sie für jeden uneingeschränkt nutzbar ist. Diese Überzeugung vertreten Stallman und seine Mitstreiter in der 1985 von ihm gegründeten Free Software Foundation (FSF) bis heute. Dazu hat Richard Stallman die GNU-General Public License (GPL) geschaffen, die die Freiheit der Software schützt.
Allerdings zögerten viele Unternehmen ein Betriebssystem einzuführen, das an jeden freizügig

[1] Richard Stallman, Gründer der Free Software Foundation.
[2] "GNU’s Not Unix!", ausgewählt weil das GNU-Design Unix ähnlich ist, sich aber com Unix-System unterscheidet, da es freie Software und kein Unix Code ist.

verschenkt wird. Angesichts dieser Skepsis aus den Reihen der Wirtschaft schlug der Software-Experte Eric S. Raymond[3] 1998 vor, Software mit offenem Quellcode künftig als Open-Source-Software zu bezeichnen. Die Open-Source-Definition lässt die Verwendung von Open-Source-Software in kommerzieller Software offen. Die GPL schränkt diese Nutzung stark ein, da für Richard Stallman nicht nachvollziehbar ist, dass ein Unternehmen Teile seiner Software freigibt und andere geheim hält.

2.1.2 Stärken und Schwächen von Open-Source-Software [6]

Die Stärken basieren auf den drei Grundprinzipien, die Open-Source-Software auszeichnet:

- Verfügbarkeit des Quellcodes und das Recht, ihn ändern zu dürfen: Da der Quelltext vorliegt, kann jeder interessierte Entwickler das Programm beliebig erweitern, verbessern und den individuellen Bedürfnissen anpassen. Fehler können durch die Mitarbeit von Programmierern in aller Welt schnell aufgespürt und behoben werden. Kein kommerziell orientiertes Unternehmen könnte eine vergleichbar große Zahl von Entwicklern bezahlen und so schnell reagieren,
- Das Recht, die Open-Source-Software sowie alle Änderungen und Verbesserungen am Quellcode weiterzugeben. Jeder Anwender kann Änderungen am Quellcode vornehmen und diese weitergeben. Dadurch wird die Qualität der Software ständig verbessert,
- Keine Exklusivrechte an der Software: Open-Source-Software steht allen offen. Dadurch kann weder ein einzelner Programmierer, noch ein Unternehmen die Richtung der Entwicklungen vorgeben. Auch die Probleme, die bei Anbietern kommerzieller Software entstehen, wenn diese ihre Geschäftstätigkeit aufgeben oder von einer anderen Firma übernommen werden, gibt es bei Open-Source-Software nicht, weil ihre Entwicklung und ihr Fortbestehen nicht von einzelnen Firmen abhängt. Stellt eine Entwicklergruppe ihre Arbeit ein, kann diese von anderen aufgenommen werden.

Neben den Stärken gibt es natürlich auch Schwächen. So weist die Hardware-Unterstützung in manchen Fällen, z.B. bei Hardware-beschleunigten Grafikkarten oder bei Multimedia-Equipment, wie Scannern, Mängel auf. Weiterhin setzt der Umgang mit Open-Source-Programmen im allgemeinen höhere Anforderungen an die Kenntnisse des Nutzers über die Funktionsweise und den Aufbau des Systems voraus als etwa im Microsoft-Umfeld üblich. Für Neueinsteiger im Open-Source-Bereich kann auch die Beschaffung von Informationen zum Problem werden.

2.1.3 OpenFOAM

OpenFOAM (Open Source Field Operation and Manipulation) ist ein in C++ geschriebenes, numerisches, freies Simulationssoftwarepaket mit dem Hauptaugenmerk auf dem Lösen von

[3] Eric S. Raymond. Software-Entwickler und Open-Source-Aktivist

CFD-Problemen. Es exisitieren viele Standardlöser für verschiedene physikalische Problemstellungen. Weitere Löser können in der OpenFOAM-eigenen Syntax hinzugefügt werden. Die Vorbereitung einer Simulation (Preprocessing) erfolgt in der aktuellen OpenFOAM-Version mit Hilfe beigelegter Tools auf der Kommandozeile ohne GUI. Es ist aber auch möglich, externe Programme zu benutzen und die erzeugten Netze anschließend in OpenFOAM zu importieren. Das Einstellen der Simulationsparameter erfolgt ohne GUI (Graphical User Interface = Grafische Benutzeroberfläche) über die Erstellung von Skriptdateien. Die graphische Aufbereitung der Simulationsergebnisse (Postprocessing) geschieht standardmäßig mit Hilfe von ParaView. Alternativ können die Ergebnisse in Formate einiger weit verbreiteter, kommerzieller Visualisierungsprogramme wie etwa Fieldview exportiert werden.

Historisches [26]

Die Anfänge der Entwicklung von OpenFOAM wurden in den späten 80ziger Jahren am Imperial College London unternommen. Dabei sollte eine leistungsstärkere und flexibler Simulationsplattform entwickelt werden als der damaligen Standard Fortran. Dieses führte zu der Wahl von C++, aufgrund der hohen Modularität und der objektorientierten Programmierung, als Programmiersprache. Der OpenFOAM-Vorgänger FOAM wurde im Jahr 2000 an die Firma „Nabla Ltd“ verkauft, die die Software kommerziell vertrieb. Die Software war zur damaligen Zeit sehr Leistungsstark, aber zu kompliziert für die Anwender. Daher konnte sich FOAM nur im akademischen Bereich durchsetzen. Ende 2004 wurde die Firma aufgelöst und im Jahr 2005 wurde FOAM unter dem Namen OpenFOAM Version 1.2 unter der Public Domain License veröffentlicht. Dabei stieg das Interesse an der Software neben den Universitäten auch bei kommerziellen Unternehmen, wie z.B. Volkswagen. 2011 wurde OpenCFD von der Firma Silicon Graphics International (SGI) übernommen. Aktuell ist die OpenFOAM Version 2.1.1 die am 31. Mai 2012 veröffentlicht wurde.

Charakteristisches Merkmal

Ein charakteristisches Merkmal von OpenFOAM ist die Schreibweise für Tensor Operationen und partielle Differenzialgleichungen, die den zu lösenden Gleichungen entspricht [23]. Beispielsweise gilt für die Impulsgleichung:

$$\frac{\partial \rho u}{\partial t} + \nabla \cdot \phi u - \nabla \cdot \eta \nabla u = -\nabla p \tag{2.1}$$

in der ρ die Dichte, ∇ der Nabla-Operator, t die Zeit, p der Druck, η die dynamische Viskosität, u die Strömungsgeschwindigkeit und ϕ eine beliebige Strömungsvariable ist, der Lösungs-Code:

```
1 solve
2 (
3       fvm::ddt(rho,u)
```

```
4     + fvm::div(phi,u)
5     - fvm::laplacian(eta,U)
6       ==
7     - fvc::grad(p)
8 );
```

Diese Schreibweise ermöglicht es Nutzern eigene Solver zu entwickeln. Begründet durch fehlende Dokumentationen werden Anpassungen mit dem fortschreitenden Ausbau der OpenFOAM-Bibliotheken allerdings schwieriger.

Struktur

OpenFOAM ist mit einer umfangreichen Basis-Bibliothek ausgestattet, die die Kernfähigkeiten des Codes bereitstellen [23] (z.B. Tensor- und Feldoperationen, Diskretisierung der partiellen Differenzialgleichungen auf Grundlage einer lesbaren Schreibweise, Lösung linearer Systeme, usw.)
Die Möglichkeiten der Basis-Bibliothek werden genutzt um Anwendungen (engl.: Applications) zu entwickeln. Die Anwendungen werden in der von OpenFOAM vorgestellten höheren Programmiersprache geschrieben, die auf eine Darstellung der konventionellen mathematischen Notationen zielt (vgl. Gleichung (2.1)). Es existieren zwei Arten von Anwendungen:

- Solver (Gleichungslöser): Führen die Berechnungen durch um ein kontinuummechanisches Problem zu lösen.
- Utilities (Hilfsmittel): Werden genutzt um das Berechnungsnetz vorzubereiten, das Simulations-Case vorzubereiten, die Ergebnisse zu bearbeiten, usw..

Jede Anwendung liefert bestimmte Möglichkeiten: Die Anwendung BlockMesh bespielsweise wird genutzt um ein Netz zu erstellen, dass der Nutzer in einer Input-Datei definiert, wohingegen die Anwendung icoFoam die Navier-Stokes-Gleichungen für eine inkompressible laminare Strömung löst.
OpenFOAM nutzt weiterhin Pakete von Drittanbietern um die parallele Funktionalität (z.B. OpenMPI) und das grafische Postprocessing (ParaView) zu ermöglichen.

Möglichkeiten

Standardmäßig beinhaltet OpenFOAM folgenden Solver für [23]:

- Basis CFD Probleme,
- Inkompressible Strömungen mit RANS- und LES-Turbulenzmodellierung,
- Kompressible Strömungen mit RANS- und LES-Turbulenzmodellierung,
- Auftriebsgetriebene Strömungen,

- Mehrphasenströmungen,
- Partikel-Tracking,
- Verbrennungsprobleme,
- Wärmeaustausch,
- und weitere.

Diese Basis-Solver können durch den Nutzer beliebig angepasst werden.
Weiterhin können die Utilities in OpenFOAM folgendermaßen eingeteilt werden [23]:

- Netzerzeugung: Sie dienen zur Erzeugung von Netzen, entweder durch eine Inputdatei (blockMesh) oder durch eine STL[4]-Datei, durch die automatische Hexaedervernetzung (snappyHexMesh).
- Netzumwandlung: Zur Umwandlung von Netzen anderer Programme zur Nutzung mit OpenFOAM
- Netzmanipulierung: Zur Manipulation des Netzes, wie bespielsweise lokalen Verfeinerungen, Definition von Regionen, usw..
- Parallelbearbeitung: Zur Zerlegung und dem Wiederzusammenführen von Berechnungsgebieten für parallele Berechnungen auf mehreren Prozessoren
- Pre-Processing: Werkzeuge um das Simulations-Case vorzubereiten
- Post-Processing: Werkzeuge zur Verarbeitung der Simulationsergebnisse

Vor- und Nachteile

OpenFOAM liefert in der Basis-Version viele Solver und Anwendungen, mit denen sich Strömungsprobleme lösen lassen. Mit Hilfe der mitgelieferten Tutorials wird der Einstieg in die Simulation mit OpenFOAM erleichtert. Reichen die Basis-Funktionen nicht aus, so bietet OpenFOAM durch die bereits erwähnte nutzerfreundliche Schreibweise der partiellen Differenzialgleichungen und die Anpassbarkeit des Codes die Möglichkeit, die Solver für das vorliegende Strömungsproblem anzupassen oder sogar gänzlich neue Solver zu schreiben. Vorteilhaft ist weiterhin, dass OpenFOAM vielgestaltige unstrukturierte Netze verarbeiten kann [23], kommerzieller Support und Trainingsmöglichkeiten seitens der Entwickler bestehen und keine Lizensgebühren für das Programm anfallen (GPL, siehe Kapitel 2.1).
Es existieren allerdings auch Nachteile. So gibt es keine integrierte grafische Oberfläche, wie

[4]Bei der STL-Schnittstelle (SurfaceTesselationLanguage (Beschreibung der Oberfläche durch Dreiecke)) handelt es sich um eine Standardschnittstelle vieler CAD-Systeme. Das STL-Format beinhaltet die Beschreibung der Oberfläche von 3D-Körpern mit Hilfe von Dreiecksfacetten.

man sie von kommerziellen CFD-Programmen (z.B. CFX, Fluent) kennt. Die Ordnerstrukturbasierte Arbeitsweise kann daher schnell unübersichtlich werden. Der größte Nachteil ist aber die nur bedingt vorhandene Dokumentation zu OpenFOAM, wodurch der Einstieg für neue Nutzer erschwert wird und auch der Lernprozess beeinträchtigt wird. Auch das Programmierhandbuch [22] liefert zu wenige Details.

2.2 Beschreibung von Strömungsfeldern

Strömungsvorgänge werden in der Strömungsmechanik im Wesentlichen anhand der Erhaltungsgleichungen für Masse, Impuls und Energie beschrieben. Für die theoretischen Betrachtungen der grundlegenden Testfälle werden diese Erhaltungsgleichungen zunächst kurz erläutert (nach [30], [15], [24], [21], [11]).

2.2.1 Massenerhaltung

Die Kontinuitätsgleichung besagt, dass für ein Volumenelement der Kantenlänge $dx\ dy\ dz$ die Differenz der in das Volumen ein- und austretenden Massenströme gleich der zeitlichen Massen- bzw. Dichteänderung des Elements sein muss.

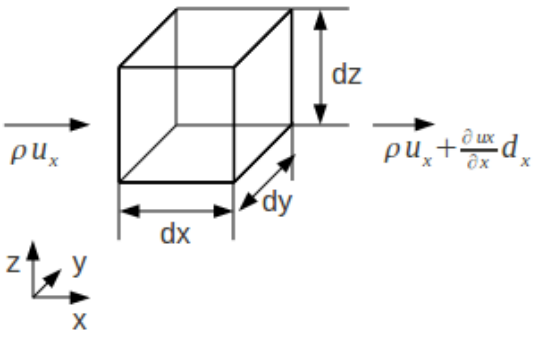


Abb. 2.1: Exemplarische Darstellung zur Definition der Massenerhaltung für die x-Richtung (modifiziert nach [30])

In das Volumenelement strömt in x-Richtung der Massenstrom $\dot{m}_{x1} = \rho\ u_x\ dy\ dz$ durch die Fläche $dy\ dz$ auf der linken Seite ein. Auf der gegenüberliegenden Seite tritt der Massenstrom

$$\dot{m}_{x2} = \left(\rho\ u_x + \frac{\partial(\rho\ u_x)}{\partial x}\ d_x\right)\ dy\ dz \tag{2.2}$$

aus dem Volumenelement aus (vgl. Abbildung 2.1). Die Differenz ergibt

$$d\dot{m}_x = \frac{\partial(\rho\ u_x)}{\partial x}\ dx\ dy\ dz. \tag{2.3}$$

Für die anderen Koordinatenrichtungen gilt analog

$$d\dot{m}_y = \frac{\partial(\rho\, u_y)}{\partial y}\, dy\, dx\, dz, \tag{2.4}$$

$$d\dot{m}_z = \frac{\partial(\rho\, u_z)}{\partial z}\, dz\, dx\, dy. \tag{2.5}$$

Durch Summenbildung folgt für die Massenänderung im Volumenelement

$$\left(\frac{\partial(\rho\, u_x)}{\partial x} + \frac{\partial(\rho\, u_y)}{\partial y} + \frac{\partial(\rho\, u_z)}{\partial z}\right) dx\, dy\, dz = -\frac{\partial \rho}{\partial t}\, dx\, dy\, dz \tag{2.6}$$

bzw.

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho\, u_x)}{\partial x} + \frac{\partial(\rho\, u_y)}{\partial y} + \frac{\partial(\rho\, u_z)}{\partial z} = 0. \tag{2.7}$$

2.2.2 Impulserhaltung (Navier-Stokes-Gleichungen)

Der Impuls I ist eine durch das Produkt aus Masse und Geschwindigkeit gebildete Bewegungsgröße. Die zeitliche Änderung dieser Größe durch die Summe aller an der Masse angreifenden Kräfte entspricht der resultierenden Trägheitskraft. Es wird wiederum ein Volumenelement mit den Abmessungen $dx\, dy\, dz$ betrachtet. Für dieses Element mit der Fluiddichte ρ ergibt sich die Trägheitskraft zu

$$\vec{F} = \frac{dI}{dt} = \frac{d(m\,\vec{u})}{dt}. \tag{2.8}$$

Durch die Anwendung der Produktregel auf diese Gleichung folgt

$$\frac{d(m\vec{u})}{dt} = \vec{u}\frac{dm}{dt} + m\frac{d\vec{u}}{dt}. \tag{2.9}$$

Die Trägheitskraft $\vec{F}$ und die Geschwindigkeit $\vec{u}$ sind vektorielle Größen. Verwendet man ein kartesisches Koordinatensystem, kann für das betrachtete Volumenelement, das in x-Richtung von einer Strömung mit der Geschwindigkeit u_x mitbewegt wird, geschrieben werden[5]:

$$dF_x = \rho\, dx\, dy\, dz\, \frac{du_x}{dt}. \tag{2.10}$$

Die Kraft dFx, die sich aus der Druckkraft, Reibungskraft und äußerer Feldkräfte zusammensetzt, bewirkt die Verschiebung des Volumenelementes:

$$dF_x = \rho\, dx\, dy\, dz\, \frac{du_x}{dt} = dF_{p,x} + dF_{R,x} + dF_{G,x}. \tag{2.11}$$

Mit dem allgemeinen Zusammenhang zwischen Druck und Kraft $dF = p\, dA$, folgt am betrach-

[5] Da sich die Masse im mitbewegten Volumenelement zeitlich nicht ändert gilt: $\vec{u}\frac{dm}{dt}$=0

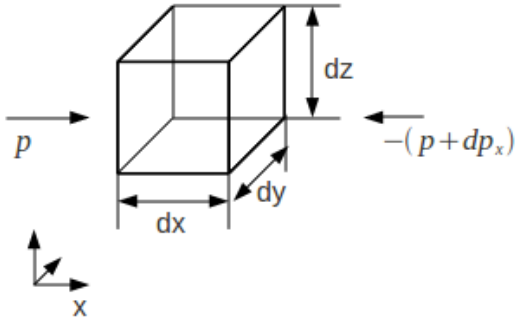


Abb. 2.2: Exemplarische Darstellung zur Definition der Druckkraft für die x-Richtung (modifiziert nach [30])

teten Volumenelement in x-Richtung auf der linken Seite (vgl. Abbildung 2.2):

$$dF_{p,x1} = p_x \, dy \, dz \tag{2.12}$$

und auf der rechten Seite

$$dF_{p,x2} = -\left(p_x + \left(\frac{\partial p}{\partial x} dx\right)\right) dy \, dz. \tag{2.13}$$

Somit ergibt sich für die Summe der Druckkräfte mit Gleichung (2.12) und (2.13) in x-Richtung

$$dF_{p,x} = -\frac{\partial p}{\partial x} dx \, dy \, dz. \tag{2.14}$$

Bei newtonschen Fluiden treten vom Geschwindigkeitsgradienten proportional abhängige (Schub-) Spannungen auf. Die Fläche an der die Spannungen wirken steht senkrecht auf der Richtung, in der sich die Strömungsgeschwindigkeit ändert (vgl. Abbildung 2.3).

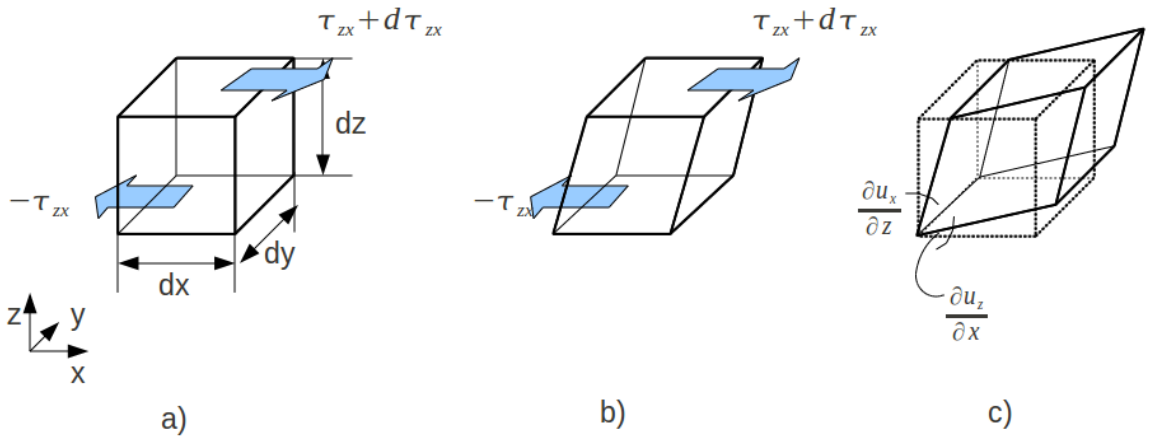


Abb. 2.3: Exemplarische Darstellung zur Definition der a) Reibkraft, b) Verformung durch Schubspannungen, c) Verformung durch Scherspannungen (modifiziert nach [30])

Für die Schubspannungen in der x-y-Ebene folgt nach Newton

$$\tau_{zx} = \eta \frac{\partial u_x}{\partial z}. \tag{2.15}$$

Bei den an den Spannungen angegebenen Indizes kennzeichnet der erste Index das normal zur angegebenen Richtung stehende Flächenelement und der zweite Index legt die Wirkrichtung der Spannungskomponente fest.
Die Schubspannungen deformieren das Volumenelement in die in Abbildung 2.3b) gezeigte Lage. Je nach Strömungzustand kommt es neben der translatorischen auch zu einer rotatorischen Bewegung des Fluidelementes. Bei der betrachteten Bewegung in die x-Richtung würde dabei eine Drehung um die y-Achse erfolgen, wodurch zusätzlich Scherspannungen in der y-z-Ebene auftreten, so dass das Volumenelement weiter deformiert wird (Abbildung 2.3c)) . Bei drehungsbehafteten Strömungen erhöht sich damit die Schubspannung in der x-y-Ebene um den Term $\eta \frac{\partial u_z}{\partial u_x}$ und Gleichung (2.15) erweitert sich auf

$$\tau_{zx} = \eta \left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x} \right). \tag{2.16}$$

Die Reibkraft ergibt sich als Produkt aus Reibspannung und Reibfläche auf der Unter- bzw. Oberseite:

$$dF_{R,u,zx} = -\tau \, dx \, dy, \tag{2.17}$$

$$dF_{R,o,zx} = -\tau \, dx \, dy + \left(\frac{\partial \tau_{zx}}{\partial z} dz \right) \, dx \, dy \tag{2.18}$$

Das Zusammenführen der Gleichungen (2.16) bis (2.18) ergibt für die Reibkräfte in der x-y-Ebene

$$dF_{R,zx} = -\frac{\partial \tau_{zx}}{\partial z} \, dz \, dx \, dy = \eta \frac{\partial}{\partial z} \left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x} \right) \, dz \, dx \, dy. \tag{2.19}$$

Neben den Schubspannungen treten auch Spannungen in der x-z-Ebene und in der y-z-Ebene[6] auf, so dass für die Kraftkomponenten vergleichbar zu Gleichung (2.19) gilt:

$$dF_{R,yx} = -\frac{\partial \tau_{yx}}{\partial y} \, dz \, dx \, dy = \eta \frac{\partial}{\partial y} \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \right) \, dy \, dx \, dz, \tag{2.20}$$

$$dF_{R,xx} = -\frac{\partial \tau_{xx}}{\partial x} \, dz \, dx \, dy = \eta \frac{\partial}{\partial x} \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_x}{\partial x} \right) \, dx \, dy \, dz. \tag{2.21}$$

Bei der Betrachtung einer kompressiblen Strömung existiert eine weitere Normalspannungskomponente, die auf die Geschwindigkeitsgradienten infolge der dichtebedingen Volumenänderung des Fluidelements zurückzuführen sind. Die Berücksichtigung dieser Spannungskom-

[6] Im betrachteten Fall einer Strömung in die x-Richtung wirken in der y-z-Ebene Normalspannungen τ_{xx}

ponente führt exemplarisch für die Normalspannung in x-Richtung auf[7]:

$$\tau_{xx} = 2\eta \frac{\partial u_x}{\partial x} - \frac{2}{3}\eta \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right). \tag{2.22}$$

Die Summe der Reibungskräfte in x-Richtung ergibt somit

$$\begin{aligned} dF_{R,x} = & \left(\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} \right) dxdydz \\ & \eta \left[\frac{\partial}{\partial x} \left(2\frac{\partial u_x}{\partial x} - \frac{2}{3} \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right) \right) + \right. \\ & \left. \frac{\partial}{\partial y} \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \right) + \frac{\partial}{\partial z} \left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x} \right) \right] dxdydz. \end{aligned} \tag{2.23}$$

Weiterhin wirkt auf das Volumenelement die Schwerkraft

$$d\vec{F}_G = \vec{g}\ \rho\ dx\ dy\ dz. \tag{2.24}$$

Durch Einsetzen der Gleichungen (2.14), (2.19) und (2.24) in die Gleichung (2.11) und nach dem Kürzen von dx dx dz ergibt sich:

$$\rho \frac{du_x}{dt} = -\frac{\partial p}{\partial x} + \left(\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} \right) + g_x\, \rho. \tag{2.25}$$

Die bisherige Herleitung der Impulsgleichung erfolgte anhand eines „materielles Teilchen“, das sich mit der Strömung mitbewegt (Lagrange’sche Betrachtungsweise). Um die Eigenschaften der Strömung an festen Raumpunkten (x,y,z) (Eulersche Betrachtungsweise) zu bestimmen, wird die Gleichung (2.25) umgeformt. Dafür wird das totale Differenzial du_x

$$\frac{du_x}{dt} = \frac{\partial u_x}{\partial x} u_x + \frac{\partial u_x}{\partial y} u_y + \frac{\partial u_x}{\partial z} u_z + \frac{\partial u_x}{\partial t} \tag{2.26}$$

gebildet. Damit folgt für die drei Komponenten der Impulsgleichung (Navier-Stokes-Gleichungen):

$$\begin{aligned} \rho \frac{\partial u_x}{\partial t} + \rho \left(\frac{\partial u_x}{\partial x} u_x + \frac{\partial u_x}{\partial y} u_y + \frac{\partial u_x}{\partial z} u_z \right) &= -\frac{\partial p}{\partial x} + \left(\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} \right) + g_x \rho \\ \rho \frac{\partial u_y}{\partial t} + \rho \left(\frac{\partial u_y}{\partial x} u_x + \frac{\partial u_y}{\partial y} u_y + \frac{\partial u_y}{\partial z} u_z \right) &= -\frac{\partial p}{\partial y} + \left(\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} \right) + g_y \rho. \\ \rho \frac{\partial u_z}{\partial t} + \rho \left(\frac{\partial u_z}{\partial x} u_x + \frac{\partial u_z}{\partial y} u_y + \frac{\partial u_z}{\partial z} u_z \right) &= -\frac{\partial p}{\partial z} + \left(\frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} \right) + g_y \rho \end{aligned} \tag{2.27}$$

[7] Der Faktor $-\frac{2}{3}\eta$ begründet sich auf die Stokessche Hypothese, vgl. hierzu z.B. [24]

Die viskosen Spannungen ergeben sich dabei aus dem deviatorischen Spannungstensor[8] (inkompressibel):

$$\begin{pmatrix} \tau_{xx} & \tau_{yx} & \tau_{zx} \\ \tau_{xy} & \tau_{yy} & \tau_{zy} \\ \tau_{xz} & \tau_{yz} & \tau_{zz} \end{pmatrix} = \eta \begin{pmatrix} 2\frac{\partial u_x}{\partial x} & \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} & \frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x} \\ \frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y} & 2\frac{\partial u_y}{\partial y} & \frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y} \\ \frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z} & \frac{\partial u_z}{\partial y} + \frac{\partial u_y}{\partial z} & 2\frac{\partial u_z}{\partial z} \end{pmatrix}. \tag{2.28}$$

2.2.3 Energieerhaltung (Wärmetransportgleichung)

Die Energiegleichung begründet sich auf die von Fourier[9] gefundene Gesetzmäßigkeit zur Wärmeleitung in festen Körpern

$$\dot{q} = -\lambda \frac{\partial \vartheta}{\partial n}. \tag{2.29}$$

Sie besagt, dass die Wärmestromdichte $\dot{q} = \frac{\dot{Q}}{A}$ in einer Querschnittsfläche senkrecht zur Richtung des Wärmeflusses proportional ist zum Temperaturgefälle $\frac{\partial \vartheta}{\partial n}$ über diese Fläche hinweg. Die Proportionalitätskonstante ist die Wärmeleitfähigkeit λ. Begonnen wird mit der Aufstellung der Wärmebilanz für ein mit der Strömung mit bewegtes Volumenelement $dx\,dy\,dz$ (Abbildung 2.4).

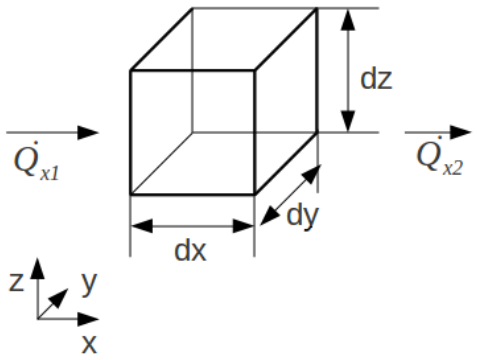


Abb. 2.4: Exemplarische Darstellung zur Definition der Wärmeleitung für die x-Richtung (modifiziert nach [30])

Auf der linken Seite tritt in das Volumenelement der Wärmestrom

$$\dot{Q}_{x1} = -\lambda \frac{\partial \vartheta}{\partial n} dydz \tag{2.30}$$

ein. Auf der gegenüberliegenden Seite tritt der Wärmestrom

$$\dot{Q}_{x2} = \left(-\lambda \frac{\partial \vartheta}{\partial n} + \frac{\partial}{\partial x} \left(-\lambda \frac{\partial \vartheta}{\partial x} dx \right) \right) dy\,dz \tag{2.31}$$

[8] Die Komponenten des deviatorischen Spannungstensors beschreiben die zähigkeitsbedingten Spannungen, d.h. die Abweichungen vom statischen Druckzustand.

[9] Jean-Baptiste-Joseph Fourier (* 21. März 1768 bei Auxerre; †16. Mai 1830 in Paris) war ein französischer Mathematiker und Physiker.

aus dem Volumenelement aus. Hieraus ergibt sich die Differenz

$$\dot{Q}_{x1} - \dot{Q}_{x2} = d\dot{Q}_x = \frac{\partial}{\partial x}\left(\lambda \frac{\partial \vartheta}{\partial x}\right) dx\, dy\, dz \tag{2.32}$$

und entsprechend in die anderen Koordinatenrichtungen

$$\begin{aligned} d\dot{Q}_y &= \frac{\partial}{\partial y}\left(\lambda \frac{\partial \vartheta}{\partial y}\right) dx\, dy\, dz, \\ d\dot{Q}_z &= \frac{\partial}{\partial z}\left(\lambda \frac{\partial \vartheta}{\partial z}\right) dx\, dy\, dz. \end{aligned} \tag{2.33}$$

Die Summe der Gleichungen (2.32) und (2.33) ergibt die zeitliche Änderung der inneren Energie des Volumenelements, also die Energie, die zu einer Temperaturerhöhung des Volumenelements führt:

$$\begin{aligned} \frac{dQ}{dt} = cp\, \rho \frac{d\vartheta}{dt} dx\, dy\, dz = \\ \left(\frac{\partial}{\partial x}\left(\lambda \frac{\partial \vartheta}{\partial x}\right) + \frac{\partial}{\partial y}\left(\lambda \frac{\partial \vartheta}{\partial y}\right) + \frac{\partial}{\partial z}\left(\lambda \frac{\partial \vartheta}{\partial z}\right)\right) dx\, dy\, dz. \end{aligned} \tag{2.34}$$

Dabei ist cp die Wärmekapazität des Fluides. Die Energiegleichung soll wie die Impulsgleichung für ein ortsfestes Volumenelement umformuliert werden. Dafür ist für die Temperaturänderung das totale Differenzial

$$\frac{d\vartheta}{dt} = \frac{\partial \vartheta}{\partial x} u_x + \frac{\partial \vartheta}{\partial y} u_y + \frac{\partial \vartheta}{\partial z} u_z + \frac{\partial \vartheta}{\partial t} \tag{2.35}$$

einzusetzen. Damit folgt für die Energiegleichung

$$\begin{aligned} cp\, \rho \left(\frac{\partial \vartheta}{\partial x} u_x + \frac{\partial \vartheta}{\partial y} u_y + \frac{\partial \vartheta}{\partial z} u_z + \frac{\partial \vartheta}{\partial t}\right) = \\ \left(\frac{\partial}{\partial x}\left(\lambda \frac{\partial \vartheta}{\partial x}\right) + \frac{\partial}{\partial y}\left(\lambda \frac{\partial \vartheta}{\partial y}\right) + \frac{\partial}{\partial z}\left(\lambda \frac{\partial \vartheta}{\partial z}\right)\right). \end{aligned} \tag{2.36}$$

Änderungen der inneren Energie durch Druck-, Normal- und Schubspannungskräfte (Reibungswärme), sowie Volumenkräfte und eine mögliche Energiezufuhr von außen bleiben an dieser Stelle unberücksichtigt, da sie für die in dieser Arbeit betrachteten Strömungsvorgänge im Vergleich zu den konvektiven Termen nur einen sehr geringen Einfluss haben.

2.2.4 Reynolds-Gleichungen

Die Navier-Stokes-Gleichungen (2.23) beschreiben allgemein die Bewegung Newtonscher Fluide. Bei turbulenten Strömungen, die die Mehrzahl aller zu betrachtenden Strömungen darstellen, sind der Hauptströmung unregelmäßig schwankende Querkomponenten in alle Raumrich-

tungen überlagert. Dieses führt dazu, dass eine mit den bisher betrachteten Erhaltungsgleichungen zu beschreibende Strömung stets instationären Charakter besitzt und eine Lösung nur noch numerisch mit erheblichem Aufwand möglich ist. Auf Reynolds geht der Ansatz zurück, die momentane Verteilung der physikalischen Feldgrößen (Geschwindigkeit, Druck, Temperatur, usw.) in die Navier-Stokes-Gleichungen (vgl. Gl. 2.27) einzuarbeiten:

$$\vec{u} = \bar{\vec{u}} + \vec{u}', \;\; p = \bar{p} + u', \;\; \vartheta = \bar{\vartheta} + \vartheta'. \tag{2.37}$$

Die erste Größe stellt dabei den Hauptanteil der Feldgröße und die zweite Größe den fluktuierenden Anteil dar. Bei einem gekoppelten Geschwindigkeits- und Temperaturfeld, hervorgerufen durch temperaturabhängige Stoffgrößen, ist es vorteilhaft massengemittelte Größen in der Form

$$\tilde{u}_x = \frac{\overline{\rho u_x}}{\bar{\rho}}, \;\; \tilde{u}_y = \frac{\overline{\rho u_y}}{\bar{\rho}}, \;\; \tilde{u}_z = \frac{\overline{\rho u_z}}{\bar{\rho}}, \;\; \tilde{\vartheta} = \frac{\overline{\rho \vartheta}}{\bar{\rho}}. \tag{2.38}$$

einzuführen und die folgenden Größen zu definieren:

$$\begin{aligned} u_x &= \tilde{u_x} + u_x'', \;\; p = \bar{p} + u', \\ u_y &= \tilde{u_y} + u_y'', \;\; \rho = \bar{\rho} + \rho', \\ u_z &= \tilde{u_z} + u_z'', \;\; p = \bar{\vartheta} + \vartheta''. \end{aligned} \tag{2.39}$$

Der Vorteil ist, dass die Größen $\overline{\phi''}$ bei einer zeitlichen Mittelung nicht Null sind. Mit den Rechenregeln für Mittelwerte lassen sich die Fluktuationsanteile in die Erhaltungsgleichungen einfügen. Für die Kontinuitätsgleichung ergibt sich:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \left(\bar{\rho}\tilde{u}_x\right)}{\partial u_x} + \frac{\partial \left(\bar{\rho}\tilde{u}_y\right)}{\partial u_y} + \frac{\partial \left(\bar{\rho}\tilde{u}_z\right)}{\partial u_z} = 0, \tag{2.40}$$

für die Impulsgleichung (nur für die x-Richtung angegeben):

$$\begin{aligned} \frac{\partial \left(\bar{\rho}\tilde{u}_x\right)}{\partial t} + \frac{\partial \left(\bar{\rho}\tilde{u}_x^2\right)}{\partial x} + \frac{\partial \left(\bar{\rho}\tilde{u}_x\tilde{u}_y\right)}{\partial y} + \frac{\partial \left(\bar{\rho}\tilde{u}_x\tilde{u}_z\right)}{\partial z} = \\ - \frac{\partial \bar{p}}{\partial x} + \left(\frac{\partial \bar{\tau}_{xx}}{\partial x} + \frac{\partial \bar{\tau}_{yx}}{\partial y} + \frac{\partial \bar{\tau}_{zx}}{\partial z} \right) + g_x\bar{\rho} \\ - \left(\frac{\partial \left(\overline{\rho u_x''^2}\right)}{\partial x} + \frac{\partial \left(\overline{\rho u_x'' u_y''}\right)}{\partial y} + \frac{\partial \left(\overline{\rho u_x'' u_z''}\right)}{\partial z} \right), \end{aligned} \tag{2.41}$$

und für die Energiegleichung:

$$\frac{\partial\left(\bar{\rho}c_p\tilde{\vartheta}\right)}{\partial t}+\frac{\partial\left(\bar{\rho}cp\tilde{\vartheta}\tilde{u}_x\right)}{\partial x}+\frac{\partial\left(\bar{\rho}cp\tilde{\vartheta}\tilde{u}_y\right)}{\partial y}+\frac{\partial\left(\bar{\rho}c_p\tilde{\vartheta}\tilde{u}_z\right)}{\partial z}= \\ \frac{\partial}{\partial x}\left(\lambda\frac{\partial\tilde{\vartheta}}{\partial x}-c_p\overline{\rho\vartheta''u''_x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial\tilde{\vartheta}}{\partial y}-c_p\overline{\rho\vartheta''u''_y}\right)+ \\ \frac{\partial}{\partial z}\left(\lambda\frac{\partial\tilde{\vartheta}}{\partial z}-c_p\overline{\rho\vartheta''u''_z}\right). \tag{2.42}$$

Bei der RANS-Impulsgleichung (Reynolds gemittelte Impulsgleichung) kommen im Wesentlichen Terme auf der rechten Seite der Gleichung hinzu, die die Fluktuationsanteile der Geschwindigkeit berücksichtigen. Die Schwankungsgrößen sorgen für einen Impulsaustausch benachbarter Fluidschichten und erhöhen scheinbar die Zähigkeit des Fluides. Sie werden deshalb als zusätzliche Reibungsglieder interpretiert. Zur Bestimmung der zusätzlichen Terme werden sogenannte Turbulenzmodelle herangezogen, die eine Beziehung zwischen den Schwankungen und der Hauptströmung herstellen. Die meisten Turbulenzmodelle für Scherströmungen basieren auf der Boussinesq-Annahme, das Reynoldsspannungen (wie die viskosen Spannungen) parallel zum Schergradient sind. Die berücksichtigte scheinbare turbulente Zähigkeit η_t ist dabei kein Stoffwert, sondern eine Ortsfunktion $\eta_t = f(x,y,z,t)$. Mit dieser Annahme wird die gesamte Information über die Wirkung der Turbulenz in einem Strömungsfeld auf die skalare Größe der scheinbaren Zähigkeit verlagert. Hieraus folgt die einschränkende Annahme, dass eine isotrope Turbulenzstruktur vorausgesetzt wird.
Die Turbulenzmodellierung der Schwankungsgrößen der Energiegleichung lässt sich entsprechend zur Impulsgleichung beschreiben. Die in diesem Zusammenhang verwendete turbulente Leitfähigkeit λ_t ist wie die Wirbelviskosität als eine Ortsfunktion innerhalb des Strömungsfeldes zu interpretieren.

2.3 Grundlagen der numerischen Strömungsmechanik

Die analytische Beschreibung von Strömungsfeldern ist nur für relativ einfache Strömungsformen möglich. Die Motivation der numerischen Strömungsmechanik besteht daher in der Absicht komplexe Strömungsformen innerhalb eines Kontrollgebietes möglichst vollständig berechnen zu können. Grundlagen hierfür sind die im vorherigen Abschnitt vorgestellten Erhaltungsgleichungen, sowie weitere Gleichungen für Turbulenz und/oder Strahlung.
Nützlich bei der Behandlung der numerischen Erhaltungsgleichungen ist deren gleichartige Struktur, die auf dem gemeinsamen Transportmechanismen der Konvektion und der Diffusi-

on beruht. In allgemeiner Form lautet diese Transportgleichung:

$$\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{\substack{\text{lokale}\\ \text{zeitliche}\\ \text{Änderung}}} + \underbrace{\frac{\partial(\rho u_x\phi)}{\partial x} + \frac{\partial(\rho u_y\phi)}{\partial y} + \frac{\partial(\rho u_z\phi)}{\partial z}}_{\text{Konvektion}} = \underbrace{\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right) + \frac{\partial}{\partial z}\left(\Gamma\frac{\partial\phi}{\partial z}\right)}_{\text{Diffusion}} + \underbrace{S_\phi}_{\text{Quellterm}} . \tag{2.43}$$

Dabei stellt die abhängige Größe ϕ die jeweils zu betrachtende Strömungsgröße dar und Γ bezeichnet den zugehörigen Diffusionskoeffizienten[10]. Aufgrund der Gleichartigkeit der Erhaltungsgleichungen können diese numerisch mit dem selben Gleichungslöser bearbeitet werden.

2.3.1 Diskretisierung der Transportgleichungen

Der erste Schritt, das System partieller Differenzialgleichungen zu lösen, ist die Diskretisierung. Zur Diskretisierung stehen die Verfahren der Finiten Differenzen (FD), der Finiten Elemente (FE) und der Finiten Volumen (FV) zur Verfügung. Das Verfahren der finiten Volumen hat bei den heute verfügbaren CFD-Codes eine dominierende Stellung und wird daher nachfolgend näher betrachtet. Bei dieser Methode wird das betrachtete Strömungsgebiet in einzelne Kontrollvolumina (Gitterzellen) zerlegt, über die die Transportgleichungen (2.43) integriert werden:

$$\begin{aligned}\int_V \frac{\partial(\rho\phi)}{\partial t}dV + \int_V \left[\frac{\partial(\rho u_x\phi)}{\partial x} + \frac{\partial(\rho u_y\phi)}{\partial y} + \frac{\partial(\rho u_z\phi)}{\partial z}\right]dV = \\ \int_V \left[\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right) + \frac{\partial}{\partial z}\left(\Gamma\frac{\partial\phi}{\partial z}\right)\right]dV + \int_V S_\phi dV.\end{aligned} \tag{2.44}$$

Vorteilhaft an dieser Form ist die Tatsache, dass sich Volumenintegrale der in den Gleichungen enthaltenen Divergenzterme mit Hilfe des Gaußschen Integralsatzes in Oberflächenintegrale umschreiben lassen. Hierdurch entfallen Ableitungen und die Ableitungen der Diffusionsterme (Reibungsterm, Wärmeleitung) werden zu Ableitungen erster Ordnung.
Die Zeitableitung erfolgt in der Regel mit

$$\int_V \frac{\partial(\rho\phi)}{\partial t}dV = \frac{\partial}{\partial t}\int_V \rho\phi dV \approx \frac{\partial}{\partial t}(\rho\phi)V \approx V\frac{(\rho\phi)-(\rho\phi)^0}{\Delta t}. \tag{2.45}$$

Dabei wird angenommen, dass sich das einzelne Kontrollvolumen zeitlich nicht ändert und die Werte innerhalb des Kontrollvolumens konstant sind. $(\rho\phi)^0$ sind dabei die Werte zum Zeitpunkt t und $(\rho\phi)$ die Werte zum Zeitpunkt $t+\Delta t$.

[10] Die Bedeutung der Größen ϕ und Γ in den Erhaltungsgleichungen: Kontnuitätsgl.: $\phi = 1, \Gamma = 0$; x-Impuls-Gl.: $\phi = u_x, \Gamma = \eta$; y-Impuls-Gl.: $\phi = u_y, \Gamma = \eta$; z-Impuls-Gl.: $\phi = u_z, \Gamma = \eta$; Energiegleichung: $\phi = \vartheta, \Gamma = \lambda$.

Ähnlich wird der Quellterm S_Q mit

$$\int_V S_Q dV = V\bar{S_Q} \tag{2.46}$$

diskretisiert, wobei $\bar{S_Q}$ den Mittelwert im Volumen darstellt.

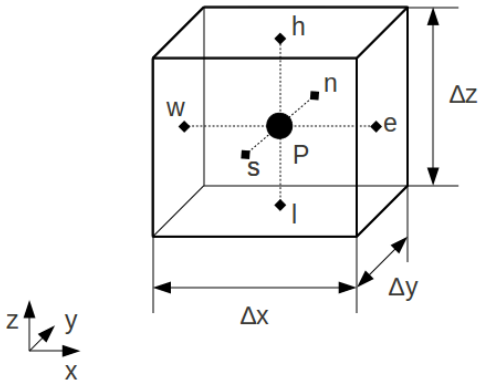


Abb. 2.5: Kartesisches Kontrollvolumen mit den üblichen Bezeichnungen (n (north), e (east), s (south), w (west), h (high), l (low)). Abbildung modifiziert nach [30]

Die Diskretisierung des Konvektionsterms einer Hexaederzelle gemäß Bild 2.5 ergibt das Volumenintegral unter Anwendung des Gaußschen Integralsatzes und der Annahme, dass über die einzelnen Flächen des Kontrollvolumens jeweils homogene Verteilungen der Größen ρ, ϕ und $\vec{u}$ vorliegen:

$$\begin{aligned}
&\int_V \left[\frac{\partial(\rho u_x \phi)}{\partial x} + \frac{\partial(\rho u_y \phi)}{\partial y} + \frac{\partial(\rho u_z \phi)}{\partial z} \right] dV = \underbrace{\oint_A \rho \phi \vec{u} \cdot \vec{n} dA}_{\text{Gaußscher Integralsatz}} = \\
&\int_{A_e} \rho \phi u_x A_e - \int_{A_w} \rho \phi u_x A_w + \int_{A_n} \rho \phi u_y A_n - \int_{A_s} \rho \phi u_y A_s + \int_{A_h} \rho \phi u_z A_h - \int_{A_l} \rho \phi u_z A_l = \\
&\underbrace{(\rho \phi u_x A)_e - (\rho \phi u_x A)_w + (\rho \phi u_y A)_n - (\rho \phi u_y A)_s + (\rho \phi u_z A)_h - (\rho \phi u_z A)_l}_{\text{Annahme homogen verteilter Größen}},
\end{aligned} \tag{2.47}$$

wobei für die Flächenelemente $A_e = \Delta y \Delta z$, $A_n = \Delta x \Delta z$ usw. gilt.
Die Diskretisierung des Diffusionsterms erfolgt auf ähnliche Weise zum Konvektionsterm. Anwendung findet dabei der Gaußsche Integralsatz und die Annäherung der Ableitungen auf den Rändern durch sogenannte Zentraldifferenzen zu den Nachbarzellen. Das Prinzip der Zentraldifferenzen ergibt Beispielsweise für die e-Seite:

$$\left(\Gamma \frac{\partial \phi}{\partial x} \right)_e = \Gamma_e \frac{\phi_E - \phi_P}{\Delta x_E}. \tag{2.48}$$

Die verwendeten Indizes stehen dabei für die Position der Zelle im Rechennetz (vgl. Abbildung 2.6).

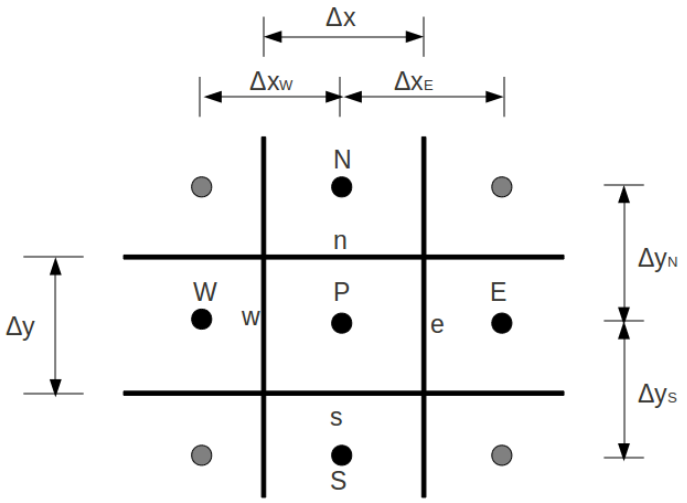


Abb. 2.6: Zweidimensionale Volumenzellengeometrie (modifiziert nach [30])

Aus dieser Vorgehensweise folgt für den diskretisierten Diffusionsterm:

$$\begin{aligned}
&\int_V \left[\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right)+\frac{\partial}{\partial z}\left(\Gamma\frac{\partial\phi}{\partial z}\right)\right]dV=\oint_A \vec{F}_\phi\cdot\vec{n}dA= \\
&\int_{A_e}\Gamma\frac{\partial\phi}{\partial x}dA_e-\int_{A_w}\Gamma\frac{\partial\phi}{\partial x}dA_w+\int_{A_n}\Gamma\frac{\partial\phi}{\partial y}dA_n-\int_{A_s}\Gamma\frac{\partial\phi}{\partial y}dA_s+\int_{A_h}\Gamma\frac{\partial\phi}{\partial z}dA_h \\
&-\int_{A_l}\Gamma\frac{\partial\phi}{\partial z}dA_l= \\
&\Gamma_e\frac{\phi_E-\phi_P}{\Delta x_E}A_e-\Gamma_w\frac{\phi_P-\phi_W}{\Delta x_W}A_w+\Gamma_n\frac{\phi_N-\phi_P}{\Delta x_N}A_n-\Gamma_s\frac{\phi_P-\phi_S}{\Delta x_S}A_S+\Gamma_h\frac{\phi_H-\phi_P}{\Delta x_H}A_h \\
&-\Gamma_l\frac{\phi_P-\phi_L}{\Delta x_L}A_l.
\end{aligned} \tag{2.49}$$

$\vec{F}_\phi$ ist dabei der Vektor, den der mit Γ multiplizierte Gradient des Skalarfeldes ϕ bildet, d.h. $\vec{F}=\left(\Gamma\frac{\partial\phi}{\partial x}\ \ \Gamma\frac{\partial\phi}{\partial y}\ \ \Gamma\frac{\partial\phi}{\partial z}\right)^T$.
Die numerische Auswertung der Transportgleichungen bedient sich der Annahme, dass die für das Kontrollvolumen berechneten Strömungsvariablen über dieses Volumen konstant sind, d.h. innerhalb des Volumens den selben Wert besitzen. Diese Werte werden im Punkt P in der Mitte des Kontrollvolumens gespeichert. Die zur Approximation der Divergenzterme benötigten Werte auf den Rändern werden aus einer Interpolation zwischen den einzelnen Zellmittelpunkten gewonnen. Zur Interpolation werden verschiedene „Diskretisierungsverfahren“ eingesetzt, von denen das vorgestellte zentrale Differenzenverfahren eine Möglichkeit darstellt, wobei von der Annahme eines stückweise linearen Verlaufes der Größen zwischen den benachbarten Zellmittelpunkten ausgegangen wird. Darüber hinaus existieren weitere Verfahren, z.B. UPWIND,

QUICK, Power law, etc. die zur Verbesserung der zu interpolierenden Werte andere Verläufe zwischen benachbarten Zellmittelpunkten ansetzen. Einen Überblick über die verschiedenen Diskretisierungsansätze gibt z.B. [11].
Zur Behandlung der diskretisierten Gleichungen mit einem Gleichungslöser ist es üblich diese in der Form

$$a_P \phi_P = \sum_{nb} (a_{nb} \phi_{nb}) + b \tag{2.50}$$

zu schreiben. P steht dabei für den KV-Mittelpunkt, a und b sind Koeffizienten, die sich aus dem Diskretisierungsansatz ergeben, und der Index nb läuft über alle Nachbarpunkte, die nach der jeweiligen Diskretisierungsmethode in die Berechnung mit einbezogen werden. Diese Gleichungen werden iterativ gelöst. Stellvertretend für die Klasse der iterativen Methoden soll an dieser Stelle das Gauss-Seidel, das Jacobi- und das ILU-Verfahren genannt werden. Eine Übersicht über die Lösungsverfahren gibt [11].
Eine besondere Schwierigkeit bei der numerischen Auswertung der Transportgleichungen tritt in Zusammenhang mit der Impulsgleichung und den in den Quellterm hineingezogenen Drucktermen auf. Für die Bestimmung des Druckfeldes existiert keine unmittelbare Gleichung. Dieses Feld ist nur indirekt durch die Kopplung von Impuls- und Kontinuitätsgleichung möglich, da unter Verwendung des korrekten Druckverlaufes in den Bewegungsgleichungen ein Geschwindigkeitsfeld resultiert, das auch die Kontinuitätsgleichung erfüllt. Bei der Diskretisierung des Quellterms der Impulsgleichungen und z.B. der Anwendung des zentralen Differenzenverfahrens verschwindet allerdings der Druck im Punkt P. D.h. das Druckfeld wird auf einem gröberen Gitter berechnet, als die Geschwindigkeiten. Das Druck- und das Geschwindigkeitsfeld sind dadurch entkoppelt, was bei der Berechnung u.U. zu physikalisch widersinnigen Ergebnissen führen kann.
Die Konsistens von Druck- und Geschwindigkeitsfeldern wird üblicherweise iterativ mit einem Druckkorrekturverfahren erreicht. Typische Druckkorrekturverfahren sind das SIMPLE-SIMPLEC-, PISO- und das PIMPLE-Verfahren (für mehr Details siehe [11]). Der beispielhafte Ablauf des SIMPLE-Verfahren ist [11]:

1. Schätzen des Druckfeldes p^0 (z.B. homogene Druckverteilung im Strömungsfeld),
2. Lösen der Impulsgleichungen für ${u_x}^0$, ${u_y}^0$, ${u_z}^0$,
3. Lösen der Druckkorrekturgleichung für Δp^0, Bestimmung der Geschwindigkeitskorrekturen $\Delta {u_x}^0$, $\Delta {u_y}^0$, $\Delta {u_z}^0$,
4. Korrektur des Druck- und Geschwindigkeitsfeldes,
5. Berechnen von Feldvariablen wie z.B. Temperatur, Dichte, Viskosität, etc., die das Strömungsfeld beeinflussen können,
6. Prüfen, ob aus dem geschätzten Druckfeld ein Geschwindigkeitsfeld resultiert, das sowohl die Impulsgleichung als auch die Kontinuitätsgleichung erfüllt und keine weitere

Druckkorrektur erforderlich ist. Wenn dieser Punkt nicht erfüllt ist, werden die Schritte 2 bis 6 wiederholt.

Die Konvergenzrate hängt dabei stark von der Zeit-Schrittgröße oder - bei stationären Strömungen - vom Wert des Unterrelaxationsparameters in den Impulsgleichungen ab [11]. SIMPLEC-, PISO-, PIMPLE-Verfahren sind Weiterentwicklungen des SIMPLE-Verfahrens. Diese Verfahren benötigen keine Unterrelaxationsfaktoren[11] für den Druck, wodurch die Konvergenz sich erhöht.

2.3.2 Turbulenzmodelle

Die Lösung der im vorherigen Kapitel vorgestellte diskretisierten Transportgleichungen bei Strömungen mit turbulentem Charakter ist nur mit einem hohen numerischen Aufwand möglich, weil die Strömung infolge der Schwankungsbewegungen lokal instationär ist. Um diese Schwankungsbewegungen in der Berechnung eines Strömungsfeldes erfassen zu können, ist eine sehr feine räumliche und zeitliche Auflösung des zu betrachtenden Gebiets erforderlich. Die sogenannte direkte numerische Simulation (DNS) ist daher z.Z. auf akademische Anwendungen beschränkt [30].
Neben den DNS-Verfahren existieren sogenannte Grobstrukturverfahren (z.B. Large Eddy Simulation (LES)), die kleinstskaligen Turbulenzbewegungen mit einfacher Modellierung beschreiben, während „großskalige Wirbel“ in der Berechnung aufgelöst werden.
Der große Teil der numerischen Strömungsberechnungen wird derzeit unter Verwendung von Wirbelviskositäts-Turbulenzmodellen mit der scheinbaren turbulenten Viskosität η_t durchgeführt. Die Größe η_t fließt in den Diffusionskoeffizienten Γ der Impulstransportgleichung (vgl. Gl. (2.43)) in der Form $\Gamma = \eta + \eta_t$ ein.
Es wird zwischen Null-, Ein-, und Zweigleichungs-Modellen unterschieden, die charakterisiert sind durch die Anzahl der Differenzialgleichungen, die zur Modellierung der Größe η_t zu lösen sind.
Nullgleichungsmodelle stellen einen rein algebraischen Zusammenhang zwischen der Wirbelviskosität und dem Geschwindigkeitsfeld her (z.B. Prandtl'sches Mischungswegkonzept, vgl. [11]). Dabei basieren viele Ein-Gleichungs-Modelle auf der Formulierung einer Transportgleichung für die kinetische turbulente Schwankungsbewegung k. Bei Zweigleichungsmodellen wird neben der k-Gleichung eine weitere Transportgleichung zur Schließung der Gleichungen verwendet. Weit verbreitet sind in diesem Zusammenhang das von Jones und Lauder [11] entwickelte k-ε-Modell sowie das k-ω-Modell und das k-ω-SST-Modell. Das k-ε-Modell liefert gute Ergebnisse für wandferne Strömungen, hat aber Probleme bei Strömungen, die an der Wand ablösen. Das k-ω-Modell hat gegenteilige Vor- und Nachteile. Aktuell verdrängt das k-

[11] Unterrelaxationsfaktoren beeinflussen die Stabilität des Lösungsverfahrens. Bei iterativen Lösungsalgorithmen können Instabilitäten auftreten, wenn die Änderung der Größe von einem zum andren Iterationsschritt nicht beschränkt wird. Dieses Limitieren der Größenänderung zwischen zwei Iterationsschritten wird als Unterrelaxation bezeichnet.

ω-SST-Model das k-ω-Modell als Standard in der Industrie, da es die guten Eigenschaften des k-ω-Modell im inneren des Strömungsfeldes und die höhere Genauigkeit des k-ω-Modells in Wandnähe kombiniert.
Die Modellgleichungen der Turbulenzmodelle haben isotropen Charakter und gelten wegen der bei ihrer Formulierung getroffenen Voraussetzungen für Strömungen hoher Reynoldszahlen. Diese Bedingung beschränkt die Anwendung auf Strömungsgebiete mit ausreichender Entfernung zu Wänden. Zur Wand hin nimmt der Turbulenzeinfluss, aufgrund der geringer werdenden Schwankungsbewegungen, bis auf Null (Haftung des Fluids an der Wand) ab. In diesem Bereich überwiegt der Einfluss der molekularen Viskosität η. Die Wandgrenzschicht wird daher bei den Turbulenzmodellen durch Wandfunktionen überbrückt. Diese beschreiben die Strömungsgrößen im wandnahen Bereich durch analytische Funktionen, die auf dem sogenannten Universellen Wandgesetz beruhen (vgl. Abbildung 2.7).

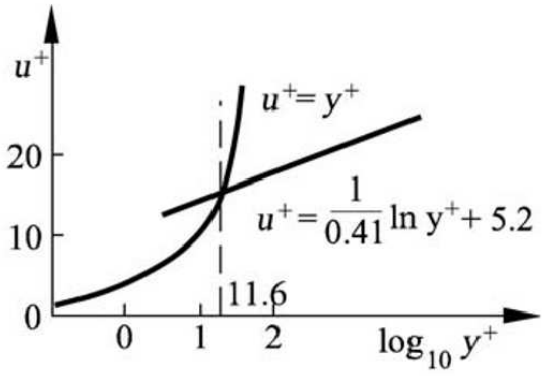


Abb. 2.7: Halblogarithmische Darstellung Geschwindigkeitsprofile in der viskosen Unterschicht und der wandnahen Schicht nach [21]: u+=Dimensionslose Geschwindigkeit, y+=Dimensionsloser Wandabstand

Aus dem isotropen Charakter der Wirbelviskositäts-Turbulenzmodelle ergeben sich Defizite derartiger Turbulenzmodelle bei Strömungsformen, bei denen mehr als eine Spannungskomponente des Spannungstensors Relevanz besitzt, wie z.B. bei dem in dieser Arbeit untersuchten Kanalbogen bei Strömungen auf gekrümmten Stromlinien, bei Systemrotationen oder bei ablösenden Grenzschichten. Darüber hinaus sind die Modelle nicht in der Lage Strömungen zu beschreiben, bei denen eine turbulente Spannungsgröße und der zugehörige Geschwindigkeitsgradient unterschiedliche Vorzeichen haben, weil η_t stets positiv definiert ist [30].
Zur Modellierung anisotroper Turbulenz finden Reynolds-Spannungs-Modelle Anwendung. Dabei werden aus den Transportgleichungen der einzelnen turbulenten Spannungen jeweils separate Modellgleichungen konstruiert. Daraus resultieren sechs Transportgleichungen für die turbulenten Spannungen und eine Transportgleichung für die Dissipation ε.
Obwohl Turbulenzeigenschaften durch eine Modellierung von Reynolds-Spannungs-Modellen physikalisch plausibel beschreibbar sind, werden im Rahmen von numerischen Strömungsfeldberechnungen derzeit vorwiegend Wirbelviskositätsmodelle eingesetzt und weiterentwickelt.

Zurückzuführen ist dieses auf [30]:

- Die höhere Anzahl der zu lösenden Gleichungen und den damit verbundenen höheren numerischen Aufwand
- Häufigere Konvergenzprobleme
- Wirbelviskositätsmodelle sind numerisch robuster
- Wirbelviskositätsmodelle liefern akzeptable Ergebnisse für einen weiten Bereich industrieller Anwendungen, Reynolds-Spannungs-Modelle zeigen hier oft keine klare Überlegenheit
- Nicht zuletzt liegen zahlreiche Vergleiche von Berechnungen auf Basis von Wirbelviskositäts-Modellen und experimentellen Untersuchungen vor

2.3.3 Randbedingungen, Konsistenz und Konvergenz

Zur Berechnung der RANS-Gleichungen müssen neben dem Turbulenzmodell Randbedingungen für das Strömungsgebiet definiert werden. Nur durch die richtige Wahl der Randbedingungen kann eine Strömung realitätsnah abgebildet werden. Die vorgegebenen Größen am Rand sind aus Messungen oder theoretischen Herleitungen bekannt (Drücke, Geschwindigkeiten, Massenströme, usw.). Die Anzahl der physikalischen Randbedingungen hängen davon ab, wie die Strömung den Rand überschreitet (Zuström-, Abström-, Festkörperrand). Allerdings dürfen nicht alle Größen am Rand vorgegeben werden, da sonst durch Rundungsfehler die Erhaltungsgleichungen verletzt werden könnten.
Für eine physikalische Lösung ist es weiterhin mindestens erforderlich, dass die Diskretisierung (vgl. 2.3.1) konsistent, das Lösungsverfahren stabil und die Lösung konvergent ist. Die Differenzengleichungen sind konsistent, wenn sie für $\Delta t, \Delta x, \Delta y, \Delta z \rightarrow 0$ in die Differenzialgleichungen übergehen bzw. ihre Abbruchfehler zu Null werden. Stabil ist ein numerisches Lösungsverfahren dann, wenn die Abbruchfehler immer kleiner werden. Zur Überprüfung dient das sogenannten Residuum, das seinen Ursprung im Abbruch der Taylorreihen hat. Nach [20] wird eine Lösung als konvergent bezeichnet, wenn das Residuum um 4-5 Größenordnungen gesunken ist (Abb. 2.8).
Darüber hinaus bezeichnet man eine numerische Methode als konvergent, wenn die Lösung der diskretisierten Gleichung bei unendlich klein werdenden Gitterabständen zur exakten - allerdings unbekannten - Lösung der Differenzialgleichung tendiert. In diesem Fall wird die Lösung als gitterunabhängige Lösung bezeichnet (vgl. [20]). Zur Kontrolle der Gitterabhängigkeit der Lösung werden Gitterunabhängigkeitsstudien bei verschiedenen Netzfeinheiten durchgeführt.

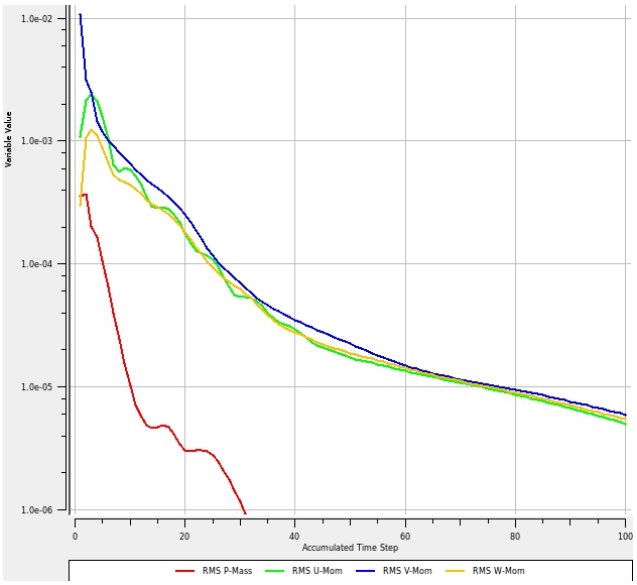


Abb. 2.8: Beispiel eines konvergenten Lösungsverlaufs

2.3.4 Ablauf einer CFD-Berechnung

Nachfolgend wird der Ablauf einer CFD-Berechnung kurz vorgestellt. Den Ablauf zeigt Abbildung 2.9.

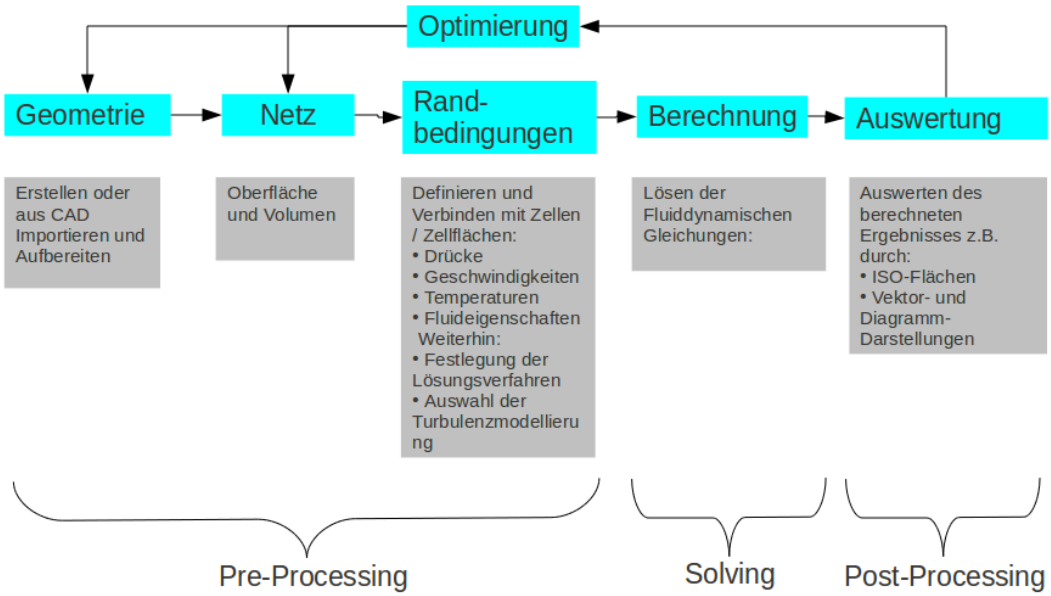


Abb. 2.9: Ablaufschemata einer CFD-Berechnung

Der Teilschritt des Pre-Processing umfasst die geometrische Aufbereitung des Strömungsfeldes, die Diskretisierung des Strömungsfeldes (Gittergenerierung), die Definition von Rand- und Anfangsbedingungen, die Festlegung der Fluideigenschaften und die Wahl der Lösungs- und Interpolationsverfahren.
Im nachfolgenden Teilschritt, dem Solving, werden die zur Beschreibung des Strömungsfeldes herangezogenen partiellen Differentialgleichungen (vgl. Kapitel 2.2) mit Finite-Volumen-,

Finite-Elemente- oder Finite-Differenzen-Methoden gelöst. Zur Beschreibung bestimmter physikalischer Phänomene und Eigenschaften der Strömung finden zusätzliche Modelle Anwendung. So werden beispielsweise zur Modellierung der Turbulenz verschiedene Turbulenzmodelle (vgl. Kapitel 2.3.2) eingesetzt oder in Abhängigkeit von der Art einer Mehrphasen-Strömung unterschiedliche Methoden zur Beschreibung von mehrphasigen Strömungsfeldern verwendet.
Nach erfolgter Simulation wird im Teilschritt des Post-Processing das berechnete Ergebnis, das in der Regel aus mehreren Millionen Daten besteht, grafisch aufbereitet. Auf der Grundlage grafischer Ergebnisdarstellungen in Form von Iso-Flächen-, Vektor- und Diagramm-Darstellungen der verschiedenen Strömungsfeldgrößen und deren integraler Größen erfolgt die Untersuchung sowie Bewertung des Strömungsfeldes. Auf diese Weise wird der direkte Vergleich von numerischen und experimentellen bzw. analytischen Daten vereinfacht und sehr bildhaft.
Zu beachten gilt, dass das Berechnungsnetz das Ergebnis beeinflusst. Daher sind Studien mit verschieden feinen Netzen durchzuführen und eine netzunabhängige Lösung sollte angestrebt werden. Aus einer CFD-Studie können beispielsweise Änderungen der Geometrie für eine optimale Strömungsführung abgeleitet werden. In diesem Fall sind mehrere Berechnungsdurchgänge mit verschiedenen Geometrien iterativ durchzuführen.

3 Auswahl der zu verwendenden CFD Werkzeugkette

Der im vorherigen Kapitel vorgestellte Ablauf einer CFD-Berechnung erfordert verschiedene Programme für die Vernetzung, das Pre-Processing, das Solving und das Post-Processing. Dabei kann Open Source Software oder kommerzielle Software verwendet werden. Die Voraussetzung ist, dass entsprechende Schnittstellen zwischen den Programmen vorhanden sind. Abbildung 3.1 zeigt mögliche CFD-Programmketten bei der Arbeit mit OpenFOAM, für die Schnittstellen vorhanden sind. Es wird dabei aber kein Anspruch auf Vollständigkeit erhoben.

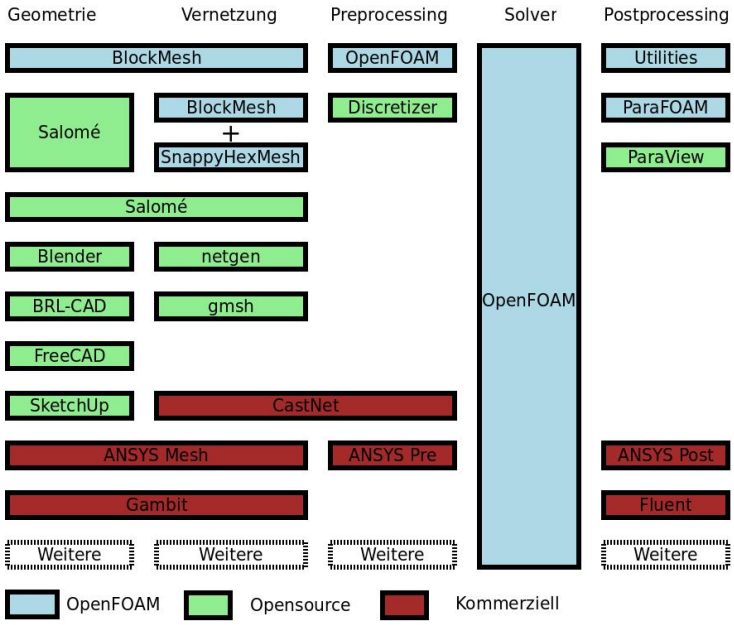


Abb. 3.1: Übersicht möglicher CFD-Programmketten für den Einsatz der OpenSource CFD Software OpenFoam (modifiziert nach [18])

Ziel dieser Arbeit soll es sein möglichst die Tools aus OpenFOAM (Blau markiert), oder, falls nicht möglich, weitere Open Source Software (Grün markiert), zu verwenden. Einfache Geometrien und das entsprechende Berechnungsnetz mit Randflächen für die Zuweisung von Randbedingungen lassen sich in Open Foam mittels des Tools „Blockmesh“ erstellen. Für die Erstellung von Berechnungsnetzen mittels einer vorhandenen einfachen oder komplexen Geometrie kann das OpenFOAM Tool „SnappyHexMesh“ verwendet werden. Die Voraussetzung dafür ist allerdings, dass das Modell im STL-Format vorliegt und die Flächen, die mit einer Randbedingung belegt werden sollen, in der STL-Datei einzeln benannt sind. OpenFOAM liefert kein Tool, dass diese Bedingungen erfüllt. Daher wird im Rahmen dieser Arbeit auf die Open Source Software Salomé zurückgegriffen, welche einen Export einzelner STL-Oberflächen ermöglicht, die anschließend benannt und wieder zusammengefasst werden können. Alternativ kann allerdings auch Blender, oder jedes weitere Programm, dass die genannten Bedingungen erfüllt, genutzt werden.

Das Pre-Processing wird, wie in OpenFOAM üblich, in Form von ordnerstrukturieren Skript-Dateien durchgeführt. Für das Post-Processing werden mitgelieferte Utilities aus OpenFOAM

und für die grafische Auswertung ParaFOAM verwendet. ParaFoam ist eine Anpassung der Open Source Software Paraview, wodurch ein direktes Einlesen der OpenFOAM Ergebnisse ermöglicht wird.
Im Rahmen dieser Arbeit wird CAELinux 2011[12] verwendet. CAELinux basiert auf Ubuntu Linux und kann direkt von der Live-CD gestartet oder dauerhaft installiert werden. Der Vorteil ist, dass die ausgewählten Programme Salomé, OpenFOAM, SnappyHexMesh, BlockMesh und ParaFoam schon vorinstalliert und auf die entsprechende Ubuntu-Version abgestimmt sind.

[12]Link: www.caelinux.com

4 Simulation drei grundlegender Testfälle

In diesem Kapitel werden drei grundlegende Strömungsformen, mit steigender Komplexität, aus dem Einsatzgebiet der Gebäudetechnik untersucht. Es soll dabei die Frage geklärt werden, ob die Simulationsergebnisse aus OpenFOAM mit denen des bei Imtech genutzten kommerziellen CFD-Programms ANSYS Fluent vergleichbare und in den bekannten Grenzen physikalisch richtige Ergebnisse liefert. Weiterhin dient dieses Kapitel dazu, einen Überblick über die Möglichkeiten der genutzten Open Source Programme zu geben, Aspekte der Bedienbarkeit herauszuarbeiten und Grenzen der Programme aufzuzeigen. Dafür werden die Schritte der Geometrieerstellung, der Vernetzung und des Pre- und Postprocessings für die Testfälle vergleichend dargestellt und bewertet.
Der erste und einfachste Testfall untersucht die turbulente Durchströmung eines Kanalbogens. Dieser Testfall eignet sich insbesondere für die Überprüfung des verwendeten Turbulenzmodells, der Wandfunktionen und des Einflusses der Vernetzung. Das Strömungsmodell ist relativ einfach, da die Strömung inkompressibel und isotherm gerechnet werden kann. Der zweite Testfall untersucht den konvektiven Wärmeübergang an einer senkrechten Wand. Die Komplexität nimmt bei diesem Strömungsproblem zu, da die Strömung nicht mehr isotherm ist, die Stoffwerte daher veränderlich sind und mehr Randbedingungen vorliegen. Im dritten Testfall wird der thermisch induzierte Auftriebsstrahl über einem Brandherd in einem Raum untersucht.
Nachfolgend werden zunächst die theoretischen Grundlagen zu den Testfällen zusammengefasst.

4.1 Theoretische Grundlagen

4.1.1 Turbulente Durchströmung eines Kanalbogens

Eine turbulenten Rohrströmung aus einem ruhenden Fluid heraus entwickelt sich gemäß Abbildung 4.1.

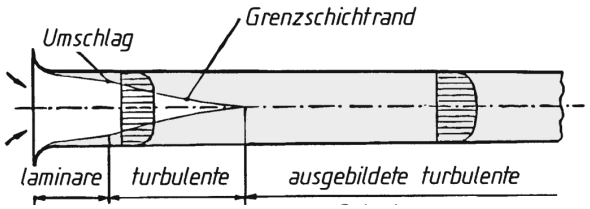


Abb. 4.1: Entwicklung einer turbulenten Rohrströmung aus der Ruhe (modifiziert nach [7])

Die Grenzschicht wird in Strömungsrichtung dicker und erfasst nach einer gewissen Einlaufstrecke den gesamten Rohrquerschnitt. Der Druckabfall in der Einlaufstrecke ist höher als bei voll ausgebildeter Strömung. Er besteht nur teilweise aus Druckverlusten (d.h. dem Verlust an mechanischer Energie), sondern zum Teil auch aus der Beschleunigung des Fluides. Charakteristisch für eine turbulente Rohrströmung ist der scharfe Geschwindigkeitsgradient in Wandnä-

he. Eine Strömung ist turbulent, wenn die kritische Reynoldszahl

$$Re = \frac{u_m\, d}{\nu} \leq Re_{krit} = 2320 \tag{4.1}$$

überschritten wird. In dieser Gleichung ist u_m die mittlere Geschwindigkeit, d ist eine charakteristische Länge, im Fall der Rohrströmung der Rohrdurchmesser, und ν ist die kinematische Viskosität. Die bei der Durchströmung eines Rohrbogens auftretenden Effekte zeigt Abbilung 4.2.

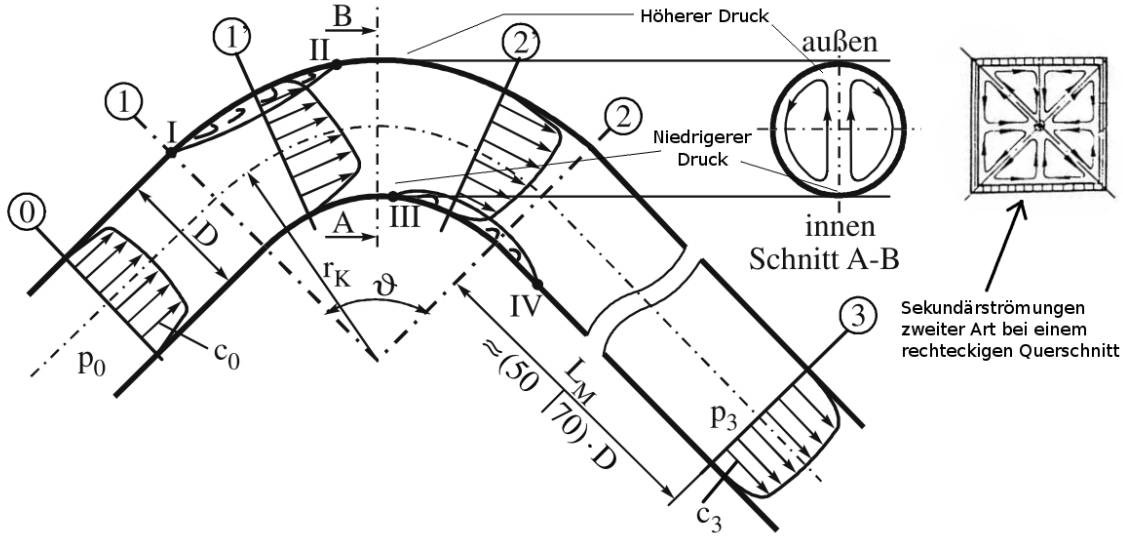


Abb. 4.2: Darstellung der Strömungsverhältnisse in einem Rohrbogen (modifiziert nach [19]) und Sekundärströmung zweiter Art in einem quadratischen Kanal (modifiziert nach [25]).

Im Zulauf (Ebene ⓪) liegt ein vollausgebildetes, turbulentes Strömungsprofil c_0 (vergleiche Abb. 4.1) und der konstante Druck p_0 vor. Im Bereich des Rohrbogens (① ⟶ ②) treten durch die Krümmung der Stromlinien Fliehkräfte auf, die durch einen von innen nach außen steigenden Druck im Gleichgewicht gehalten werden (Energiekonstanz). Dieses verursacht im Krümmeraußenbereich (I ⟶ II) einen Druckanstieg und eine damit verbundene Geschwindigkeitsabnahme, während im Innenbereich der Druck fällt und die Geschwindigkeit steigt (Ebene ①́). Im weiteren Verlauf des Krümmers steigt dann im Innenbereich (III ⟶ IV) der Druck wieder bis etwa auf den Druck p_3 der ausgeglichenen Abströmung an, während die Geschwindigkeit entsprechend abnimmt. Im zugehörigen Außenbereich sinkt der Druck zum Krümmerende hin entsprechend gegen p_3, was zur Beschleunigung der Außenströmung führt (Ebene ②́). Noch im Krümmeraustritt (Ebene ②) liegt außen eine höhere Geschwindigkeit als innen vor. Erst nach einer Auslaufstrecke L_M von 50 bis 60 mal dem Rohrdurchmesser liegt erneut ein vollausgebildetes turbulentes Strömungsprofil c_3 vor [19].
In den Wandbereichen I ⟶ II und III ⟶ IV strömt die Grenzschicht gegen steigenden Druck, was zu Ablösung mit Wirbelbildung und zusätzlichen Druckverlusten führt. Im Schnitt A - B des Krümmers liegt außen ein höherer Druck als innen vor. Die langsam strömenden Wand-

grenzschichten stehen nicht im dynamischen Gleichgewicht mit dem Druckgradienten in Umfangsrichtung. Daher wird das Grenzschichtmaterial längs der Rohrwand nach innen gedrückt, trifft dort auf die entsprechende Strömung der Gegenwand und wird in der Rohrmitte wieder nach außen geführt. Eine solche Strömung, die in einer Ebene normal zur Hauptströmung stattfindet, wird Sekundärströmung genannt und kann in allen gekrümmten Kanälen beobachtet werden. Im Fall des Rohrbogens entstehen zwei schraubenförmige gegenläufige Sekundärwirbel die nach [25] als Sekundärströmungen erster Art bezeichnet werden.
Bei nicht kreisförmigen Querschnitten kommt es zu Sekundärströmungen zweiter Art (vgl. Abbildung 4.2). Diese Sekundärströmung wird auch die „Tote Ecken ausfüllende Strömung“ genannt, da aufgrund der erhöhten Reibung in den Ecken das strömende Fluid in diese Ruhegebiete ausweichen will [25].
Der Druckverlust einer turbulenten Strömung in einem geraden Rohr ist nach Gleichung (4.2) quadratisch von u_m abhängig:

$$\Delta p_v = \lambda \frac{\Delta l}{d} \rho \frac{{u_m}^2}{2}. \tag{4.2}$$

λ ist die Rohrreibungszahl, Δl die betrachtete Rohrlänge, d der Rohrdurchmesser, ρ die Dichte des Fluids und u_m die mittlere Geschwindigkeit. Die direkte Reibung beschränkt sich auf eine dünne Schicht in Wandnähe, die sogenannte laminare Unterschicht. Außerhalb bewirken turbulente Austauschbewegungen weit größere Verluste (vgl. Kapitel 2.2.4). Die laminare Grenzschicht ist von großer Bedeutung für den Turbulenzgrad bzw. das Ausmaß des Druckverlustes, abhängig davon, ob sie die Wandrauhigkeiten, die zusätzliche Turbulenzen hervorrufen ganz abdeckt oder nicht. Beschrieben wird der Einfluss der laminaren Grenzschicht durch die Rohrreibzahl der ausgebildeten Rohrströmung λ. Lambda kann nach Formeln berechnet oder im Colebrook-Diagramm (Abb. 4.3) abgelesen werden.

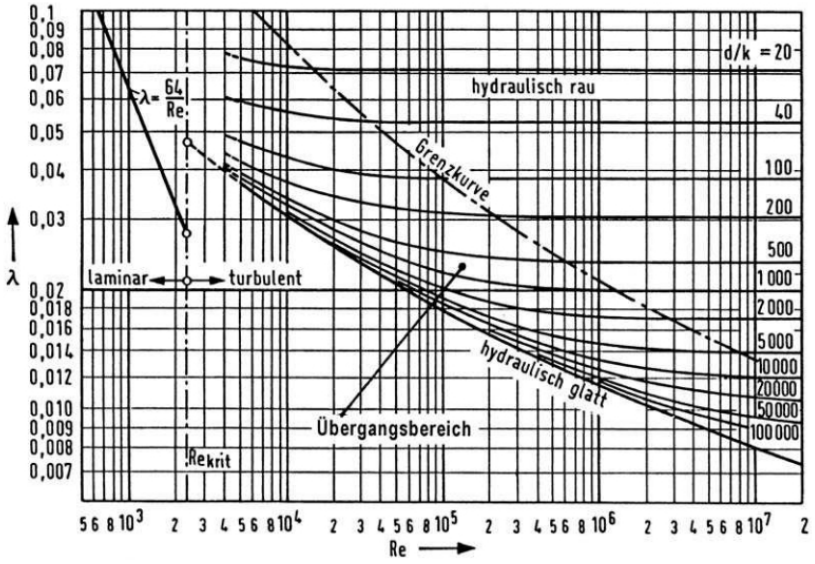


Abb. 4.3: Colebrook-Diagramm für den Rohrwiderstandsbeiwert λ (nach [7])

In diesem Diagramm zeigt sich, dass bei einer kleinen Rauhigkeitserhebung k, bei der alle

Unebenheiten in der laminaren Grenzschicht liegen (k $<\delta_l$), die Rauhigkeit keine Widerstandserhöhung bewirkt und ein rauhes Rohr sich wie ein glattes Rohr verhält [29]. In diesem Fall ist die Rohrreibzahl ausschließlich von der Reynoldszahl Re abhängig ($\lambda = f(Re)$) und setzt sich vor allem aus den Zähigkeitseinflüssen des Mediums zusammen [29].
Die durch den Krümmer hervorgerufenen Verluste entstehen zusammenfassend durch die Wandreibung (λ), die Ablösungen der Grenzschicht und Sekundärströmungen erster und zweiter Art (Vergleiche Abbildung 4.2). Abbildung 4.4 zeigt die Zusammensetzung der Verlustbeiwerte für einen Rohrbogen.

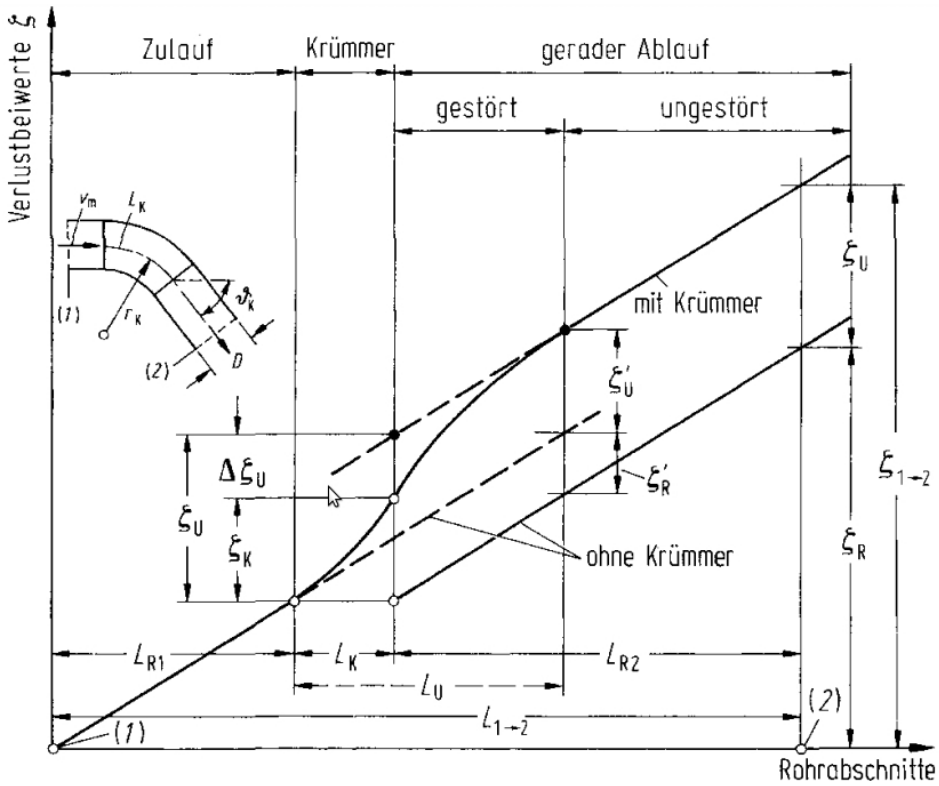


Abb. 4.4: Zusammensetzung der Verlustbeiwerte in Rohrbögen (nach [27])

Demnach beträgt der Verlustbeiwert einer zwischen den Stellen (1) und (2) aus dem geraden Zulaufrohr, dem Rohrbogen und dem geraden Ablaufrohr bestehenden Rohrumlenkung der Gesamtlänge $L_{1\longrightarrow 2} = L_{R1} + L_K + L_{R2} = L_R + L_K$

$$\zeta_{1\longrightarrow 2} = \zeta_R + \zeta_U \tag{4.3}$$

mit $\zeta_R = \lambda \cdot \frac{L_R}{D}$ als Verlustbeiwert der beiden geraden Rohrstücke und ζ_U dem durch die Rohrumlenkung im Rohrbogen und im gestörten Ablaufrohr ($L = L_U$) hervorgerufenen Verlustbeiwert, der sich folgendermaßen zusammen setzt:

$$\zeta_U = \acute{\zeta}_R + \acute{\zeta}_U \tag{4.4}$$

mit $\acute{\zeta}_R = \lambda \frac{L_K}{D}$.

4.1.2 Freie Konvektion und Wärmeübergang an einer ebenen Wand [15]

In vielen wärmetechnischen Problemen im Bereich der Gebäudetechnik treten Strömungen mit konvektivem Wärmeübergang auf. Der zweite Testfall untersucht daher den freien konvektiven Wärmeübergang an einer senkrechten Wand. Während eine erzwungene Strömung durch äußere Kräfte, beispielsweise die durch eine Pumpe oder ein Gebläse aufgeprägten Drücke hervorgerufen wird, entsteht die freie Strömung durch Massenkräfte in einem Fluid, in welchem Dichtegradienten vorhanden sind. An einer beheizten senkrechten Wand (vgl. Abbildung 4.5) erwärmt sich das Fluid und erfährt einen Auftrieb. Es entsteht eine freie Strömung, die zunächst laminar ist und nach einer gewissen Länge turbulent wird.
Der entscheidende Unterschied zur erzwungenen Konvektion ist, dass eine Strömung nur in unmittelbarer Wandnähe auftritt, wo diese durch Auftriebskräfte induziert wird (Grauer Bereich in Abbildung 4.5). Für Prandtlzahlen nahe Eins (wie bei Luft) sind die Schichtdicken für das Temperaturprofil und das Geschwindigkeitsprofil wie in Abbildung 4.5 dargestellt etwa gleich groß. In diesem Fall wird die Reynoldszahl, der charakteristischen Kennzahl bei erzwungener

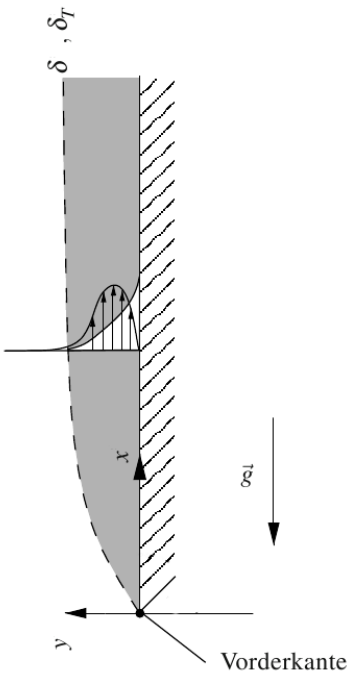


Abb. 4.5: Geschwindigkeits- und Temperaturverteilung in laminaren Grenzschichten bei natürlicher Konvektion (Pr $\approx$ 1) (modifiziert nach [15])

Konvektion, nicht mit der charakteristischen vorgegebenen Geschwindigkeit U_∞, sondern mit einer durch die Temperaturdifferenz hervorgerufene Geschwindigkeit

$$u_B = \sqrt{gL\beta\Delta T} \tag{4.5}$$

gebildet, da diese Temperaturdifferenz die Ursache für die Strömung darstellt. g ist der Betrag des Erdbeschleunigungsvektors, L eine wählbare Bezugslänge, β der isobare thermische Aus-

dehnungskoeffizient und ΔT die charakteristische Temperaturdifferenz.
Setzt man diese Geschwindigkeit in die Reynoldszahlgleichung ein, so erhält man die Grashofzahl als charakteristische Kennzahl für Probleme freier Strömung:

$$Re = \frac{\rho u_B L}{\eta} = \frac{\rho \sqrt{g\beta\Delta T} L^{3/2}}{\eta} = \sqrt{Gr};\ \ Gr = \frac{\rho^2 g\beta\Delta T L^3}{\eta^2} \tag{4.6}$$

und zusammen mit der Prandtl-Zahl die Rayleight-Zahl, die den Charakter der Wärmeübertragung innerhalb des Fluides beschreibt:

$$Ra = Gr\ Pr. \tag{4.7}$$

Die Wärmestromdichte wird mit der Lauflänge stets kleiner, weil die Temperaturgradienten, mit denen insgesamt die konstante Temperaturdifferenz $\Delta T = T_W - T_\infty$ überbrückt wird, in dem Maße abnehmen, in dem der Auftriebsstrahl sich aufweitet. Damit nimmt auch der Temperaturgradient an der Wand in Laufrichtung ab, was zur Verringerung der Wärmestromdichte führt. An jeder Stelle x muss die mit diesem Temperaturprofil gespeicherte innere Energie genau der Energie entsprechen, die bis zu dieser Stelle in Form von Wärme insgesamt über die Wand übertragen worden ist [15].
Im Falle einer turbulenten Strömung kommt es im Vergleich zur laminaren Strömung zu einem deutlich verbesserten Wärmeübergang. Dieses ist durch die Schwankungsbewegungen der Strömung zu begründen, wodurch es zu einer erheblichen Intensivierung des Austausches von Impuls und innerer Energie benachbarter Fluidteilchen kommt. Während bei laminaren wandparallelen Strömungen ein Impulstransport und ein Transport innerer Energie jeweils quer zur Hauptströmungsrichtung ausschließlich aufgrund molekularer Transportmechanismen erfolgt, kommt es bei turbulenten Strömungen zu einer lokalen Vermischung und damit zu einem erheblichen Quertransport von Impuls und Energie [15]. Zu den molekularen Transportkoeffizienten Viskosität η (für den Impuls) und Wärmeleitfähigkeit λ (für die innere Energie) treten deshalb jeweils turbulenzbedingte, zusätzliche Koeffizienten hinzu. Für die effektiven Größen folgt daher

$$\eta_{eff} = \eta + \eta_t, \tag{4.8}$$

$$\lambda_{eff} = \lambda + \lambda_t \tag{4.9}$$

beim Übergang von einer laminaren zu einer turbulenten Strömung. Die Größen η_t und λ_t sind keine Stoffwerte, sondern sind sog. Strömungsgrößen. Für den Wärmeübergang ist entscheidend, dass η_t und λ_t an der Wand selbst Null sind, weil die Wand sehr wandnahe Schwankungsbewegungen stark dämpft und an der Wand selbst keine Strömung und damit auch keine Schwankungsbewegungen vorliegen. Abbildung 4.6 zeigt den prinzipiellen Verlauf der Transportkoeffizienten, dabei wird zwischen einer Wand- und einer Außenschicht unterschieden. In

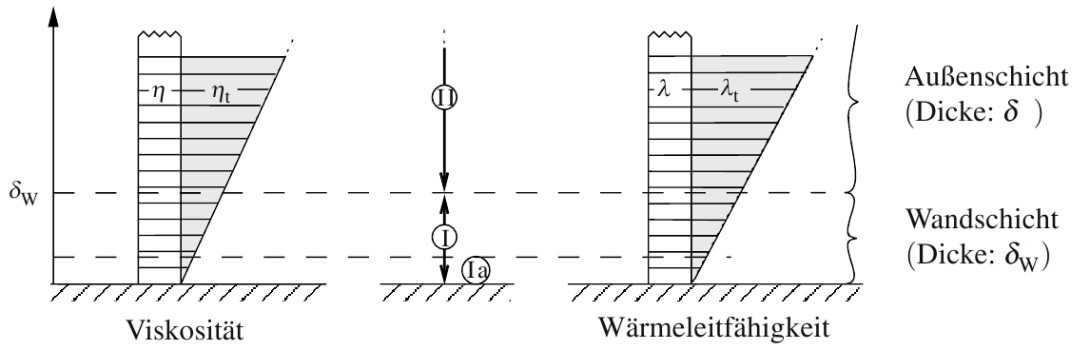


Abb. 4.6: Prinzipieller Verlauf der Transportkoeffizienten nach [15]

der Wandschicht ① spielen sowohl die molekularen (η, λ) als auch die turbulenten (η_t, λ_t) Austauschkoeffizienten bei der Definition des Geschwindigkeits- und Temperaturprofils eine Rolle. Innerhalb dieser Schicht kann noch eine sog. viskose Unterschicht indentifiziert werden, in der ausschließlich der molukulare Austausch wirksam ist (im Bild 4.6 ①a). In der Außenschicht, bei größeren Wandabständen, dominieren die turbulenten Austauschkoeffizienten.
Die Wandschichtdicke δ_w ist erheblich kleiner als die Dicke der Außenschicht, oftmals nur einen Bruchteil eines Millimeters. Trotzdem ist diese Schicht von entscheidender Bedeutung, weil sich in ihr die Übertragungsprozesse an der Wand abspielen.

4.1.3 Thermisch induzierter Auftriebsstrahl und Rauchschichtung [30]

Der dritte Testfall untersucht die Simulations-Möglichkeiten von OpenFOAM hinsichtlich thermisch induzierter Auftriebsstrahlen und dadurch hervorgerufene thermische Schichtenströmungen. Diese Strömungsform spielt in der Gebäudetechnik eine wichtige Rolle, da im Falle eines Gebäudebrandes Personen die Flucht ermöglicht werden soll und daher für die Dauer der Evakuierung der Aufenthalts- bzw. Fluchtwegbereich raucharm zu halten ist. Die Höhe h_{SG} dieser raucharmen Schicht ist gebäude- bzw. nutzungsabhängig. Die Dimensionierung der notwendigen Rauchabzugeinrichtung basiert auf der Berechnung des durch den (Brandrauch-) Auftriebsstrahl in der Höhe $z = h_{SG}$ transportierten Massenstroms $\dot{m}_{AS}$, der durch Induktion von Umgebungsluft mit zunehmender Strahllauflänge wächst.
Die sich bei einem Brand einstellende Raumströmung lässt sich idealisiert durch die Einteilung in drei Strömungsbereiche veranschaulichen. Ausgehend von einer freien Auftriebsströmung, die den ersten Bereich charakterisiert, erreicht der Konvektionsstrahl nach einer Lauflänge $z = H$ die Raumdecke. Hier schließt sich als zweiter Bereich eine walzenförmige Strömung an. Als dritter Bereich ist der untere Bereich zu nennen, der sich vom Boden bis zur Höhe h_{SG} des zu betrachtenden Raumes erstreckt.

Auftriebsstrahl (plume)

Wie im zweiten Testfall ergibt sich die Strömung im Auftriebsstrahl einzig durch Dichteunterschiede des zu betrachtenden Fluids. Die turbulente natürliche Auftriebsströmung lässt sich dabei in zwei Bereiche unterteilen, den Strahlformierungsbreich und den Ähnlichkeitsbereich (Bild 4.7).

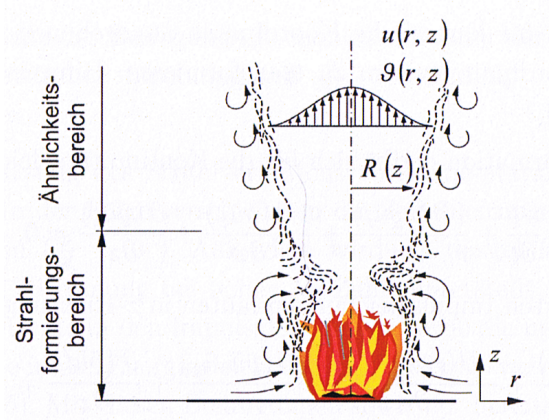


Abb. 4.7: Freier turbulenter Auftriebsstrahl nach [30]

In unmittelbarer Nähe der Wärmequelle wirken im Strahlformierungsbereich Trägheits-, Auftriebs-, Reibungs- und Druckkräfte. Letztere sorgt dabei für das horizontale Nachströmen von Fluidteilchen in Richtung der Wärmequelle. Bei Annäherung des Fluidteilchens an das Zentrum der Wärmequelle wächst durch die fortschreitende Erwärmung die Auftriebskraft gegenüber der Druckkraft stetig an, was zu Strömungsablösungen und schließlich zur Bildung des Auftriebsstrahls führt. Bei flächenmäßig kleinen Wärmequellen formiert sich der Auftriebsstrahl idealerweise mittig. Mit steigender Wärmefreisetzung weichen die realen Strömungszustände von diesem Ideal ab, so dass sich mehrere Auftriebsstrahlen oberhalb der Wärmequelle bilden. Jeder Einzelstrahl baut dabei seine kinetische Energie durch Induktion der Umgebungsluft ab, wodurch es zwischen den Auftriebsstrahlen zu einer instabilen Strömung mit lokal hohen Turbulenzen kommt. Die Strömungsverhältnisse in diesem Bereich sind daher nicht analytisch beschreibbar.
Im oberen Bereich legen sich die Auftriebsstrahlen zusammen und bilden einen stabilen Einzelstrahl. In diesem sogenannten Ähnlichkeitsbereich kann das Strahlverhalten mit Hilfe der im Kapitel 2.2 angegebenen Erhaltungsgleichungen abgeschätzt werden.

Strömungsverhältnisse in der Rauchschicht

Nach einer Lauflänge von $z = h_{SG}$ erreicht ein sich über einer Wärmequelle formierender Konvektionsstrahl die Schichtgrenze. Nach dem Durchströmen dieser Grenze löst sich der Strahl in aller Regel nicht auf, sondern strömt weiter, wobei sich allerdings die Auftriebskraft ändert, da nun in seiner Umgebung die Temperatur der Rauchschicht herrscht.

Im Nahbereich der Raumdecke erfährt der vertikale Auftriebsstrahl eine Umlenkung (Staupunktströmung), wodurch eine radiale Abströmung des Fluides resultiert (Abb. 4.8).

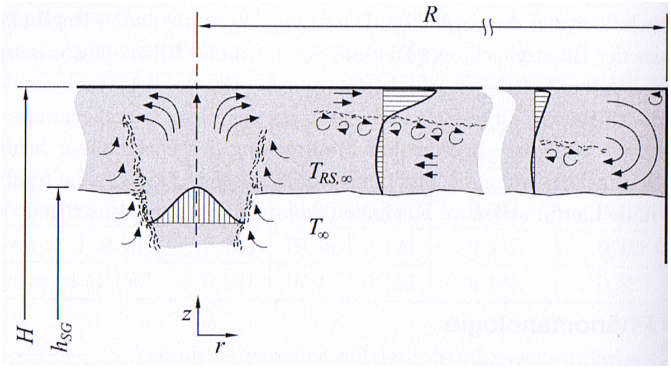


Abb. 4.8: Strömungsverhältnisse im Bereich der deckennahen Schicht nach [30]

An die Strahlumlenkung schließt sich ein radialer Deckenstrahl an. Die mittlere Strahlgeschwindigkeit in diesem Bereich nimmt mit der Entfernung zum Mittelpunkt des Staupunktes ab. Die Ursache hierfür liegt in der in radialer Richtung stetig zunehmenden Strahlaufweitung und an der durch turbulenten Impulsaustausch hervorgerufenen Strahlinduktion am Strahlrand. Analog zum Auftriebsstrahl vergrößert sich der im Deckenstrahl transportierte Massenstrom mit zunehmender Lauflänge.
Wenn sich der Deckenstrahl in einer bereits formierten oberen Schicht ausbreitet, so kann angenommen werden, dass die Temperaturdifferenz zwischen dem Deckenstrahl und der Strahlumgebung klein ist. Unter dieser Voraussetzung besteht im Bezug auf die Strahlausbreitung eine nur schwache Kopplung zwischen Impuls- und Energiegleichung, so dass vereinfacht isotherme Verhältnisse bzw. ein Feld konstanter Dichte in der oberen Schicht unterstellt werden können [30].

4.2 Untersuchungsmerkmale

Auf der Grundlage der theoretischen Betrachtungen der drei Testfälle sind die zu untersuchenden Strömungsmerkmale festzulegen. Zu den gewählten Merkmalen sollten dabei belastbare Messwerte und im Idealfall Ergebnisse aus Fluent-Strömungssimulationen vorliegen. Nachfolgend werden die in den Simulationen zu untersuchenden Merkmale der drei Testfälle zusammengefasst.

4.2.1 Turbulente Durchströmung eines Kanalbogens

Für den ersten Testfall werden als Ausgangspunkt für die Validierung der numerischen Strömungssimulation die auftretenden Sekundärströmungen (Abbildung 4.2) qualitativ untersucht, sowie ebenfalls qualitativ der Verlauf des Totaldrucks über dem Kanalbogen verglichen (Abbildung 4.4). Die Geometrie des Kanalbogens wird dabei so gewählt, dass die Strömungsverhältnisse im Kanalbogen mit denen des dargestellten Rohrbogens in Abbildung 4.4 vergleichbar

sind. Als quantitative Validierungsgrundlage der Simulation werden Druckverlustbeiwerte nach [16] genutzt. Dieses mehrere hundert Seiten starke Werk, mit gemessenen Verlustbeiwerten für verschiedenste Geometrien, kann als Standard für die Bestimmung von Druckverlusten angesehen werden. Daher wird ein Kanalbogen aus diesem Werk, der weiterhin typische Maße eines Strömungskanals der Gebäudetechnik aufweist, ausgewählt (Abb. 4.9).

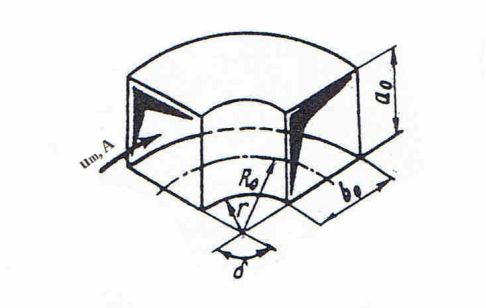


Abb. 4.9: Gewählte Kanalbogengeometrie (modifiziert nach [16])

Die aufgeführten Druckverlustbeiwerte sind in folgenden Grenzen gültig:

$\frac{R_o}{b_0} < 3{,}0$, $0 < \delta \leq 180°$, $\frac{l_0}{D_h} \geq 10$ und $Re \geq 2 \times 10^5$.

l_0/D_h ist die Länge der Einlaufstrecke vor dem Rohrbogen und D_h der hydraulische Durchmesser, der sich für einen rechteckigen Querschnitt durch:

$$D_h = \frac{2\, a_0\, b_0}{a_0 + b_0} \tag{4.10}$$

berechnen lässt. Tabelle 4.1 zeigt die gewählten und berechneten Parameter des Kanalbogens.

Tab. 4.1: Gewählte Parameter des zu Untersuchenden Kanalbogens

a_0 in m	b_0 in m	R_0/b_0	δ	D_h in m	l_0/D_h	Re	Wandrauhigkeit
1	1	2	90°	1	10	3×10^5	hydraulisch glatt

Nach [16] lässt sich der Druckverlust durch

$$\zeta \equiv \frac{\Delta p}{\rho {u_m}^2/2} = \zeta_{loc} + \zeta_{fr} \tag{4.11}$$

berechnen. ζ_{loc} ist die Kennzahl des lokalen Strömungswiderstandes und ζ_{fr} die Kennzahl des Widerstandes aufgrund von Reibungseinflüssen. Dabei setzt sich ζ_{loc} aus den Koeffizienten $A_1 = f(\delta)$, $B_1 = f(R_0/b_0)$ und $C_1 = f(a_0/b_0)$ zusammen:

$$\zeta_{loc} = A_1 \cdot B_1 \cdot C_1. \tag{4.12}$$

Diese Koeffizienten können in [16] nachgeschlagen werden. Für die gewählten Parameter nach Tabelle 4.1 folgen hieraus für die Koeffizienten A1=1, B1=0,15 und C1=1 und somit nach Gleichung 4.12 für $\zeta_{loc} = 0,15$.
ζ_{fr} wird für ein gerades Rohr mit dem hydraulischen Durchmesser D_h und der Länge der Achse des Kanalbogens bestimmt. Für ein gerades Rohr gilt $\zeta_{fr} = \lambda \cdot \frac{l}{D_h}$. Die Länge der Kanalachse lässt sich durch das Verhältnis

$$\frac{l}{D_h} = \pi \frac{\delta^\circ}{180^\circ} \frac{R_o}{D_h} = 0,0175 \frac{R_o}{D_h} \delta \tag{4.13}$$

bestimmen. Daraus folgt:

$$\zeta_{fr} = 0,0175 \frac{R_o}{D_h} \delta \lambda. \tag{4.14}$$

λ kann dem Diagramm 4.3 oder Tabellenwerten nach [16] entnommen werden. Weiterhin lässt sich λ für hydraulisch glatte Rohre und einer Reynolds-Zahl größer 4000 nach der Gleichung

$$\lambda = \frac{1}{(1,8 \cdot lg(Re) - 1,64)^2} \tag{4.15}$$

berechnen [16]. Nach Gleichung (4.15) ergibt sich für die Rohrreibzahl ein Wert von $\lambda = 0,015$ und nach Gleichung (4.14) für ζ_{fr}=0,04725. Damit ergibt sich für die Gesamtwiderstand-Ziffer des Kanalbogens nach Gleichung (4.11) ein Wert von $\zeta = 0,197$. Tabelle 4.2 listet die berechneten theoretischen Widerstandsziffern zusammenfassend auf.

Tab. 4.2: Berechnete Verlustkennziffern des Kanalbogens

ζ_{loc}	ζ_{fr}	ζ
0,15	0,04725	0,197

Beachtet werden muss dabei, dass der lokale Widerstandsbeiwert ζ_{loc} nach [16] nicht nur den lokalen Druckverlust über den Kanalbogen, sondern auch den Druckverlust nach dem Bogen beinhaltet, da sich nach der Störung die Geschwindigkeiten in der geraden Auslaufstrecke (gestörte Länge L_A) erst wieder ausgleichen müssen (vgl. Abb. 4.4).
Weiterhin lässt sich über die gewählte Reynoldszahl mit dem hydraulischen Durchmesser und der kinematischen Viskosität von Luft die mittlere Einströmgeschwindigkeit u_m in den Kanalbogen bestimmen:

$$u_m = \frac{Re \cdot D_h}{\nu}. \tag{4.16}$$

Mit der kinematischen Viskosität der Luft bei 20°C von $\nu = 1,532 \cdot 10^{-5}$ m^2/s (vgl. [17]) und den Werten nach Tabelle 4.1 ergibt sich eine Einströmgeschwindigkeit in den Kanalbogen von 4,596 m/s. Mit dieser Geschwindigkeit werden vergleichende CFD-Simulationen mit Fluent und OpenFOAM durchgeführt.

4.2.2 Freie Konvektion und Wärmeübergang an einer ebenen Wand

Als Untersuchungsgrundlage dient die experimentelle Studie [3], die niedrig turbulente natürliche Konvektion in einem mit Luft gefüllten Raum untersucht. Der untersuchte Raum ist 0,75 m hoch, 0,75 m breit und 1,5 m tief. Durch die Tiefe des Raumes stellt sich in der Mittelebene eine zweidimensionale Strömung ein. Die senkrechten Wände des untersuchten Raumes werden mit konstanten Temperaturen von 10 °C und 50 °C beheizt, bzw. gekühlt. Die Rayleigh Zahl ist mit $1,58\ x\ 10^9$ angegeben und die Prandl-Zahl beträgt 0,71. Gemessen werden die lokalen Geschwindigkeiten und Temperaturen an verschiedenen Stellen im Raum. Aus den Messwerten werden anschließend mittlere und fluktuierende Parameter bestimmt und weiterhin die mittlere und die lokale Nusseltzahl, Wandschubspannungen, die turbulente kinetische Energie und die Dissipation angegeben. Die Geschwindigkeitsmessungen werden mit einem LDA durchgeführt, der eine Unsicherheit von 0,07% aufweist. Die Temperaturmessung wird mit einem Thermoelement mit einer Unsicherheit von 0,1 K durchgeführt. Weitere Genauigkeitsangaben und Einzelheiten zum Testaufbau sind [3] zu entnehmen.
Für die Validierung der OpenFOAM Simulationen werden die Messwerte in einer Höhe von 0,375 m in der Mitte des Raumes genutzt. Es werden die dimensionslose Temperatur und die Geschwindigkeit parallel zur Wand ausgewertet sowie die lokale und mittlere Nusselt Zahl berechnet und verglichen. Im Ausblick auf den Testfall der Rauchschichtung wird weiterhin die Temperaturverteilung über die Höhe des Raumes untersucht.

4.2.3 Thermisch induzierter Auftriebsstrahl und Rauchschichtung

Als Untersuchungsgrundlage für den dritten Testfall dient [30]. Dieser Fortschritts-Bericht des VDI untersucht das Schichtungsverhalten von Brandrauch anhand einer Basisraumgeometrie. Die Basisraumgeometrie bildet im wesentlichen ein zylinderförmiges Strömungsgebiet mit einer bodennahen ringförmigen Zuluftöffnung und einer deckennahen Abluftöffnung. In den Simulationsrechnungen wird als Wärmequelle ein „Volumenmodell“ verwendet, berücksichtigt durch einen Wärmestrom-Quellterm (W/m^3) in der Energiegleichung.
Der Forschungsbericht untersucht mehrere Modellvarianten mit verschiedenen Raumdurchmesser, Absaugmassenströmen, sowie Wärmefreisetzungsraten. Die Untersuchungen in dieser Arbeit beschränkten sich auf die Simulation einer Modellvariante in OpenFOAM und dem Vergleich mit den vorhandenen Fluent Ergebnissen. Verglichen wird dabei zum einen die sich einstellende Schichthöhe h_{SG} in Abhängigkeit vom Absaugmassenstrom $\dot{m}_{AB}$ und die Deckenstrahlausbreitung in Form der Geschwindigkeiten im Deckenstrahl.
Die Bestimmung der Schichtgrenze erfolgt aus dem Temperaturfeld, das sich im Berechnungsmodell einstellt. In der Studie [30] werden dabei zwei Temperaturverläufe über die Raumhöhe an zueinander um 90° versetzten Positionen verwendet. Die Auswertpositionen befinden sich dabei in einem Abstand von 0,5 m und 1 m zur Raumwand. Die Temperaturen an dieser Stelle werden gemittelt und die maximale vertikale Temperaturänderung bestimmt. Im Rahmen des

Beitrages [30] wird diese Position als Schichtgrenze identifiziert. Die Geschwindigkeiten im Deckenstrahl werden an vier radialen Positionen (2 m, 4 m, 6 m, 8 m) untersucht, qualitativ mit Abbildung 4.8 und weiterhin mit den Fluent-Ergebnissen der Studie [30] verglichen.

4.3 Geometrieerstellung und Vernetzung

Im Anschluss an die Vorstellung der Grundlagen und der Definition der Untersuchungsmerkmale wird nachfolgend auf die Geometrieerstellung und Vernetzung der drei grundlegenden Testfälle eingegangen. Der Ablauf wird dabei ausführlich am Beispiel des Kanalbogens beschrieben. Für die weiteren Testfälle werden anschließend nur noch die Besonderheiten dargestellt.

4.3.1 Turbulente Durchströmung eines Kanalbogens

Der Kanalbogen besitzt eine relativ einfache Geometrie. Eine strukturierte Blockvernetzung wäre daher möglich und würde zu einem sehr guten Netz führen. Bei einer komplizierteren Geometrie ist dieses Vorgehen nicht möglich oder sehr zeitaufwendig. Die Vernetzung sollte daher möglichst automatisch erfolgen. Der in dieser Arbeit verfolgte Ansatz ist die Vernetzung des Strömungsgebietes mit dem Vernetzungsprogramm SnappyHexMesh. Für die Vernetzung mit SnappyHexMesh wird eine STL-Geometrie des Bauteils benötigt. Diese Geometrie wird in Salomé erstellt.
Bei der Erstellung der Rohrbogengeometrie ist die CFD-Berechnung bereits mit zu berücksichtigen. Eine Beschränkung der Geometrie auf den Rohrbogen nach Abbildung 4.9 wäre daher nicht zielführend. Vielmehr sind definierte Ein- und Auslaufstrecken für die Simulation zu berücksichtigen. Eine Einlaufstrecke ist erforderlich, da die Strömungsgeschwindigkeit am Einströmrand konstant vorgegeben wird (Kolbenprofil) und sich erst nach einiger Zeit das typische turbulente Strömungsprofil ausgebildet hat (vgl. Abb. 4.1). Nach [16] soll die Einauflänge $L_{ein} \geq 10 \cdot D_h$ sein. Dieses deckt sich auch mit der Angabe nach [7] für die turbulente Einlauflänge von $L_{ein} = (10 \text{ bis } 30) \cdot d$. Nach dem Kanalbogen weist die Strömung eine deutliche Störung auf. Daher wird die Auslaufstrecke deutlich länger als die Einlaufstrecke gewählt. Die Länge orientiert sich an den Angaben nach [29], wonach die Umlenkwiderstandsbeiwerte ζ_U nur bei einer genügend langen Auslaufstrecke von $L_{aus} \geq 50 \cdot d$ gelten.

Geometrieerstellung in Salomé

Die Geometieerstellung in Salomé ist flächenbasiert. Dabei werden Skizzenelemente über verschiedene Vorgaben definiert (Linien, Bögen, Kreise, usw.) und daraus anschließend Flächen erstellt. Um den Flächen des Kanals später Randbedingungen zuweisen zu können, müssen diese in Salomé gruppiert werden. Es werden drei Gruppen - Inlet, Outlet, Walls - erstellt, die entsprechenden Flächen zugewiesen, anschließend einzeln exportiert, mit einem Editor geöffnet und benannt (Inlet, Outlet, Walls).

Die STL-Dateien mit benanntem Solid - Inlet, Outlet, Walls - werden danach zu einer einzigen Datei zusammengefasst. Hier wird ein Linux Befehl (z.B. cat inlet.stl » assembly.stl) verwendet.

Vernetzung

Die Vernetzung der Kanalbogengeometrie soll mit SnappyHexMesh erfolgen. SnappyHexMesh benötigt für die Vernetzung ein Grundnetz. Dieses Netz wird im BlockMesh-Werkzeug erstellt. Im Fall des Kanalbogens muss das BlockMesh die in Salomé erstellte Kanalbogengeometrie vollständig aufnehmen. Grundsätzlich arbeitet SnappyHexMesh so, dass das Grundnetz an den Schnittpunkten mit den Flächen der Geometrie verfeinert und getrennt wird. Der nicht benötigte Netzbereich wird anschließend entfernt (Im Falle des Kanalbogens das Netz außerhalb der Kanalbogengeometrie). Der verbleibende Teil des Netzes wird an die Geometrie angelegt und falls gewünscht Boundary-Layer angebracht.
Nachfolgend ist die BlockMeshDict Datei aufgeführt, die die Erstellung des Grundnetzes steuert. Die Datei wird zum besseren Verständnis in mehrere Teilbereiche aufgeteilt.
Es beginnt jede OpenFoam-Datei mit einem Header, der u.a. die verwendete FOAM-Version benennt (hier nicht dargestellt), im Anschluss erfolgt eine Beschreibung der FOAM-Datei (hier ebenfalls nicht dargestellt, siehe Anhang A.1). Im weiteren Verlauf werden die Eckpunkte des Blockmesh festgelegt:

```
15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17 convertToMeters 1;
18
19 vertices
20 (
21     ( -0.5 -0.5 -0.5 ) //0
22     (  10  -0.5 -0.5 ) //1
23     (  11  -1.5 -0.5 ) //2
24     (  11  -12  -0.5 ) //3
25     (  13  -12  -0.5 ) //4
26     (  13  -1.5 -0.5 ) //5
27     (  10   1.5 -0.5 ) //6
28     ( -0.5  1.5 -0.5 ) //7
29     ( -0.5 -0.5  1.5 ) //8
30     (  10  -0.5  1.5 ) //9
31     (  11  -1.5  1.5 ) //10
32     (  11  -12   1.5 ) //11
33     (  13  -12   1.5 ) //12
34     (  13  -1.5  1.5 ) //13
35     (  10   1.5  1.5 ) //14
36     ( -0.5  1.5  1.5 ) //15
37     (  11  -52   1.5 ) //16
38     (  13  -52   1.5 ) //17
39     (  11  -52  -0.5 ) //18
40     (  13  -52  -0.5 ) //19
41 );
```

Zeile 17 bietet eine Konvertierungsoption (hier 1, da alle Punktkoordinaten in Meter angegeben werden). Beispielhaft wird Zeile 21 betrachtet: In diesem Fall sind die Koordinaten x=-0,5m,

y=-0,5m und z=-0,5m. Als Kommentar ist anschließend vermerkt, dass es sich um den Punkt 0 handelt (zählweise von 0 bis n). Bei der Eingabe der Punktekoordianten ist ein rechtshändiges Koordinatensystem zu verwenden [23]. Die erstellten Punkte zeigt Abbildung 4.10.

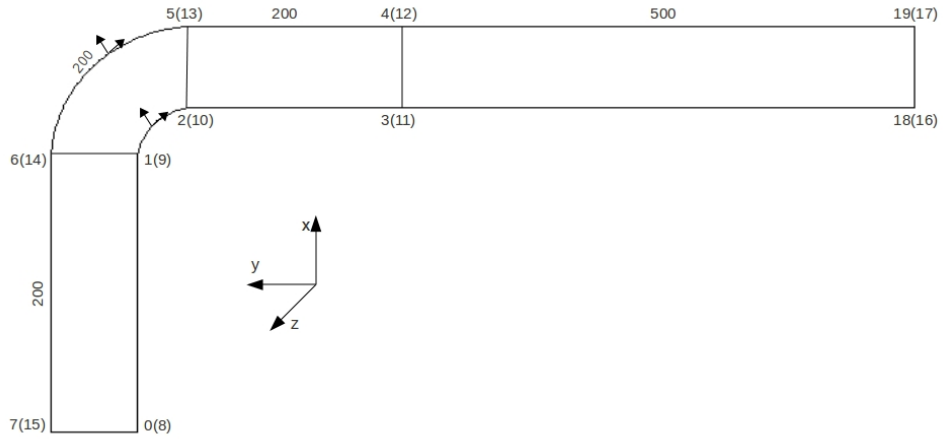


Abb. 4.10: Eckpunkte der Blöcke des Blocknetzes

Dargestellt sind die Koordinaten auf der x-y-Ebene mit einem z-Wert von -0,5m. In Klammern sind die Punkte auf der parallelen Ebene mit einem z-Wert von 1,5m angegeben. Weiterhin sind die vier geplanten Blöcke dargestellt. Diese werden im BlockMeshDict wie folgt definiert:

```
43 blocks
44 (
45     hex (0 1 6 7 8 9 14 15) (200 40 40)   simpleGrading (1 1 1)
46     hex (1 2 5 6 9 10 13 14) (60 40 40)   simpleGrading (1 1 1)
47     hex (2 3 4 5 10 11 12 13) (200 40 40)   simpleGrading (1 1 1)
48     hex (4 3 18 19 12 11 16 17) (40 500 40)   simpleGrading (1 1 1)
49 );
```

Beispielhaft wird Zeile 45 betrachtet. Zu Beginn wird die Form der Netzzellen festgelegt (hier „hex“ für Hexaeder). Anschließend erfolgt die Definition der Blöcke durch die Angabe der Block-Eckpunkte. In der zweiten Klammer erfolgt die Angabe der Zellen je Kante (hier von $0 \longrightarrow 1$ 200 Zellen, von $1 \longrightarrow 6$ 40 Zellen und in die verbleibende Richtung (hier z-Richtung) ebenfalls 40 Zellen, vgl. Abb. 4.10).

Abschließend wird die Kantenverdichtung der Zellen festgelegt. „simpleGrading“ gibt an, dass die Netzzellen nur in eine Richtung verdichtet werden sollen. Die Verdichtungsrichtung ist dabei ebenfalls durch die Definition des Blockes festgelegt (erster Wert gleich Verdichtung von Punkt $0 \longrightarrow 1$, zweiter Wert gleich Verdichtung von Punkt $1 \longrightarrow 6$ und dritter Wert gleich Verdichtung in positive z-Richtung). Die Werte (1 1 1) zeigen, dass keine Verdichtung der Netzzellen erfolgt.

Bisher sind die erstellten Blöcke rechteckig. Im Bereich des Bogens wären die Netzzellen daher nach der Vernetzung mit SnappyHexMesh nicht in Strömungsrichtung ausgerichtet. Aus diesem Grund wird der in Abbildung 4.10 gezeigte Bogen folgendermaßen erstellt:

```
51 edges
```

```
52 (
53     arc 2 1    ( 10.5 -0.633974596 -0.5 )
54     arc 5 6    ( 11    1.098076211 -0.5 )
55     arc 10 9   ( 10.5 -0.633974596  1.5 )
56     arc 13 14  ( 11    1.098076211  1.5 )
57 );
```

Beispielhaft wird Zeile 53 betrachtet. „arc 2 1“, definiert, dass zwischen den Punkten 2 und 1 ein Kreisbogen (engl. arc) erstellt wird. Zur Definition eines Kreisbogens ist ein weiterer Punkt auf dem Kreisbogen erforderlich. Dieser Punkt ist in Zeile 53 in x,y und z-Koordinaten in Klammern angegeben und in Abbildung 4.10 symbolisch eingezeichnet.
Abschließend können Flächen definiert und benannt werden:

```
59 boundary
60 (
61 );
62
63 // ************************************************************************* //
```

Im Falle des Kanalbogens kann auf eine Definition der äußeren Flächen des Block-Netzes verzichtet werden, da diese nicht mit Randbedingungen versehen und vielmehr durch SnappyHexMesh weggeschnitten werden.
Zur Erstellung des Block-Netzes wird anschließend das OpenFoam Terminal gestartet, der Projektordner aufgerufen und der Befehl **blockMesh** ausgeführt. Abbildung 4.11 zeigt das erstellte Block-Netz und die innen liegende - mit Salomé - erstellte Geometrie (rot).

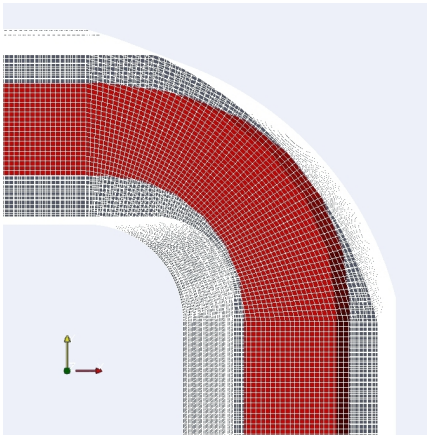


Abb. 4.11: Erstelltes Block-Netz mit innen liegender Kanalbogengeometrie

Die anschließende Vernetzung mit SnappyHexMesh erfolgt voll automatisiert anhand eines Ablaufskriptes. Das Script „SnappyHexMeshDict“ wird dafür zusätzlich in den Projektordner gelegt und editiert. Abbildung 4.12 zeigt die Ordnerstruktur nach der Ausführung von „BlockMesh“ und vor der Ausführung von „SnappyHexMesh“ im OpenFoam Terminal. Neu hinzugekommen ist weiterhin der Ordner „triSurface“, in dem die aus Salomé exportierte und editierte STL-Geometrie (vgl. Kapitel 4.3.1) abgelegt wird. Die Dateien „fvSchemes“ und „fvSolutions“ sind ebenfalls für die Ausführung von SnappyHexMesh erforderlich, haben aber keinen Einfluss auf die Vernetzung (eine genauere Betrachtung erfolgt daher wie bei der Datei „controlDict“ an

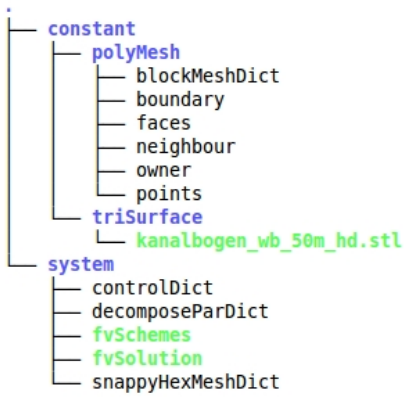


Abb. 4.12: Ordnerstruktur vor der Ausführung von SnappyHexMesh

späterer Stelle). Ebenfalls erforderlich ist die Datei „decomposeParDict“. Diese Datei steuert die Zerlegung des Netzes in Teilbereiche für eine parallele Berechnung auf mehreren Prozessoren. Da der Kanalbogen nur auf einem Prozessor berechnet wird, wird auf eine tiefer gehende Beschreibung der Datei an dieser Stelle verzichtet.
Die Datei „SnappyHexMeshDict“ beginnt ebenfalls mit einem Header. Nach dem Header erfolgt die Festlegung der durchzuführenden Schritte:

```
19 castellatedMesh  true;
20 snap             true;
21 addLayers        true;
```

Die Schritte werden in der aufgelisteten Reihenfolge durchlaufen, vorausgesetzt sie sind als wahr (engl. true) gekennzeichnet. Zu Beginn wird „castellatedMesh“ (zu Deutsch: Kronennetz) ausgeführt. Dabei werden die Zellen des Block-Netzes, die von dem Oberflächennetz der Geometrie geschnitten werden verfeinert, geteilt und der nicht benötigte Netzbereich entfernt (Im Falle des Kanalbogens das Netz außerhalb der Kanalgeometrie). Nach dieser Operation ist das Oberflächennetz zackig und es existieren Abstände zwischen den STL-Oberflächen und den Netzzellen, bzw. es ragen Netzzellen aus den STL-Oberflächen hinaus.
Daher wird anschließend „snap“ (zu Deutsch: das Schnappen) durchgeführt. Dabei werden die Zellen so angepasst, dass sie nach den Oberflächen der Geometrie „schnappen“, bzw. sich an sie anlegen. Als dritter Schritt wird eine Grenzschicht (engl. boundary layer) erstellt.
Im „SnappyHexMeshDict“ wird danach die Geometrie auf nachfolgend gezeigte Weise festgelegt:

```
30 geometry
31 {
32     kanalbogen_wb_50m.stl
33     {
34         type triSurfaceMesh;
35         name kanalbogen;
36         regions
37         {
38             walls
39             {
40             name wall;
```

```
41              }
42              inlet
43              {
44              name in;
45              }
46              outlet
47              {
48              name out;
49              }
50          }
51      }
52  };
```

Zeile 32 nennt die im Unterverzeichnis „TriSurface“ liegende STL-Geometriedatei (hier kanalbogen_wb_50.stl). Zeile 34 definiert die Art des Oberflächennetzes (hier Dreiecksfacetten). Anschließend wird in Zeile 35 der Name der Geometrie auf „kanalbogen“ festgelegt. Somit braucht bei späterer Adressierung nicht mehr der komplette Name ausgeschrieben werden. Weiterhin werden in Zeile 36-50 die im STL benannten Randflächen genannt (vgl. Kapitel 4.3.1) und ebenfalls mit Referenznamen versehen.

In den nächsten Zeilen erfolgen die Einstellungen für das CastellatedMesh, auf deren Darstellung an dieser Stelle verzichtet werden soll.

Anschließend wird in der SnappyHexMeshDict ein relativ neues Merkmal (engl. Feature) definiert. Erst seit der Version 2.0 ist das Merkmal „Kantenverfeinerung“ vorhanden [23]. Um dieses Merkmal nutzen zu können, müssen vorher im OpenFoam Terminal im Projekt Ordner folgende Befehle ausgeführt werden: „*.$WM_PROJECT_DIR/bin/tools/RunFunctions*“ und „*surfaceFeatureExtract -includedAngle 150*“.

Dadurch werden Dateien erzeugt, die die extrahierten Punkte, Kanten, Kantennormalen, usw. enthalten. Die nun im Unterordner vorhandenen Datei /constant/trisurface „kanalbogen_wb_50m.eMesh“ wird in der SnappyHexMeshDict verwendet:

```
94      // Explicit feature edge refinement
95      // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
96
97      // Specifies a level for any cell intersected by its edges.
98      // This is a featureEdgeMesh, read from constant/triSurface for now.
99      features
100     (
101         {
102             file "kanalbogen_wb_50m.eMesh"; //FeatureEdgeMesh
103             level 0; //level of refinement
104         }
105     );
```

Da später Randschichten vorgesehen werden sollen, wird auf eine Verfeinerung der Kanten verzichtet (Zeile 103). Trotzdem ist die Definition der .eMesh Datei an dieser Stelle erforderlich, damit sich das Netz beim „Snapping“ an die Kanten anpassen kann. Ohne diese Definition wird das Netz nur sehr unsauber an die Kanten angelegt.

Zeile 118 bis 142 der Datei BlockMeshDict legt die Oberflächenverfeinerung fest:

```
118     refinementSurfaces
```

```
119     {
120         kanalbogen
121         {
122             // Surface-wise min and max refinement level
123             level (0 0);
124         regions
125         {
126            walls
127            {
128            level (0 0);
129            }
130            inlet
131            {
132            level (0 0);
133            }
134            outlet
135            {
136            level (0 0);
137            }
138         }
139         }
140     }
141
142     resolveFeatureAngle 30;
```

Dabei wird in Klammern jeweils der minimale und maximale Verfeinerungslevel angegeben, zuerst für den gesamten Kanalbogen und anschließend für einzelne definierte Flächen. Das Verfeinerungslevel Null zeigt, dass keine Verfeinerung der Oberflächen vorgenommen wird. Als weitere Option steht die Verfeinerung einer Region innerhalb des Netzes zur Verfügung. Diese Option wird für den Kanalbogen nicht verwendet und daher hier nicht weiter erwähnt. Letzter Eintrag für die CastellatedMeshControls in Zeile 172 ist die Definition eines Punktes im Bereich des Netzes, der erhalten bleiben soll (Also ein Punkt innerhalb der Kanalgeometrie):

```
172     locationInMesh (1 0.5 0.5); // Inside point
```

Anschließend werden die Einstellungen für das „Snapping“ festgelegt:

```
182 // Settings for the snapping.
183 snapControls
184 {
185     //- Number of patch smoothing iterations before finding correspondence
186     //  to surface
187     nSmoothPatch 3;
188
189     //- Relative distance for points to be attracted by surface feature point
190     //  or edge. True distance is this factor times local
191     //  maximum edge length.
192     tolerance 1.0;
193
194     //- Number of mesh displacement relaxation iterations.
195     nSolveIter 300;
196
197     //- Maximum number of snapping relaxation iterations. Should stop
198     //  before upon reaching a correct mesh.
199     nRelaxIter 5;
200
201     //- Highly experimental and wip: number of feature edge snapping
```

```
202     //  iterations. Leave out altogether to disable.
203     nFeatureSnapIter 20;
204 }
```

Die gezeigten Einstellungen sind durch austesten und Vergleiche mit den Tutorials ermittelt worden. Als wichtigster Punkt für die Kantenauflösung hat sich nFeatureSnapIter (Zeile 203) herausgestellt. Hier wird das vorher erstellte .emesh verwendet.
Im Anschluss an diesen Schritt werden die Einstellungen für die Randnetzschicht festgelegt. Nach [8] ist für eine Simulation mit Wand-Funktionen ein Y-Plus Wert zwischen $n = 20$ und $n = 200$ anzustreben. Je genauer die Simulation sein soll, desto kleiner sollte der Y-Plus Wert gewählt werden, allerdings sollte der Y-Plus Wert nicht unter 20 liegen, da das Netz sonst in die viskose Unterschicht hinein reicht. Weiterhin sollte eine Randschicht bei der Nutzung von Wandfunktionen den wandnahen Bereich genau genug Auflösen und daher aus mindestens $n = 10$ Teilschichten im logarithmischen Bereich (vgl. Abb. 2.7) bestehen. Das Wachstumsverhältnis der Schichten normal zur Wand sollte höchstens $n_{ratio} = 1,25$ betragen. Es gilt die Zellhöhe der ersten Schicht abzuschätzen. Die Abschätzung wird wie in [9] beschrieben durchgeführt:

1. Berechnen der Reynolds-Zahl nach Gleichung 4.1 (Im Falle des Rohrbogens ist die Reynoldszahl auf 3×10^5 festgelegt).

2. Abschätzen der Wandreibung, z.B. durch die Wand-Reibungskorrelation von Schlichting [24]: $C_f = [2 \cdot log_{10}(Re_x) - 0,65]^{-2,3}$ für $Re_x < 10^9$. Für $Re_x = 3 \times 10^5$ ergibt $C_f = 4,678 \cdot 10^-3$.

3. Berechnen der Wandschubspannung durch $\tau_w = C_f \cdot \frac{1}{2} \cdot \rho \cdot U_x^2$ mit $U_x = 4,596 \frac{m}{s}$ (vgl. Kap. 4.2.1). Zusammen mit der Dichte von Luft bei 20 °C von $\rho = 1,1885 \frac{kg}{m^3}$ ergibt sich für $\tau_w = 5,872 \cdot 10^{-2} \frac{N}{m^2}$

4. Berechnen der Reibungsgeschwindigkeit: $u_* = \sqrt{\frac{\tau_w}{\rho}} = 0,222 \frac{m}{s}$

5. Berechnen des Wandabstandes für den angestrebten Y-Plus Wert: $y = \frac{y^+ \cdot \mu}{\rho \cdot u_*}$. Mit der dynamischen Viskosität für Luft bei 20°C von $\mu = 18,205 \cdot 10^{-6}\ Pa \cdot s$ und einem angestrebten Y-Plus Wert von 40 folgt für die Höhe der ersten Schicht y ein Wert von $y = 2,8 \cdot 10^{-3}\ m$

Die berechnete Grenzschicht wird folgendermaßen in der SnappyHexMeshDict eingestellt:

```
210 // Settings for the layer addition.
211 addLayersControls
212 {
213     // Are the thickness parameters below relative to the undistorted
214     // size of the refined cell outside layer (true) or absolute sizes (false).
215     relativeSizes false;
216
217     // Per final patch (so not geometry!) the layer information
218     layers
219     {
```

```
220         wall
221         {
222             nSurfaceLayers 16;
223         }
224     }
225
226     // Expansion factor for layer mesh
227     expansionRatio 1.05;
228
229
230     //- Wanted thickness of final added cell layer. If multiple layers
231     //  is the thickness of the layer furthest away from the wall.
232     //  See relativeSizes parameter.
233     finalLayerThickness 0.025;
234
235     //- Minimum thickness of cell layer. If for any reason layer
236     //  cannot be above minThickness do not add layer.
237     //  See relativeSizes parameter.
238     minThickness 0.0014;
```

Zeile 215 legt fest, dass alle Größenangaben zur Grenzschicht absolut erfolgen. Zeile 217 bis 227 legt fest, dass die Randschicht nur auf den Wänden, nicht aber am Einlass und am Auslass erstellt werden und das die Schicht aus 16 Teilschichten aufgebaut werden soll. Anschließend wird in Zeile 227 das Wachstumsverhältnis festgelegt, d.h., die Zellenhöhe nimmt zur Kanalmitte hin je Schicht um 1,05 zu. Zeile 233 legt die Höhe der am weitesten vom Rand entfernten Zelle fest. Die Höhe der letzten Schicht ergibt sich dabei aus der Höhe der ersten Zelle multipliziert mit dem Schichtwachstum hoch der Schichtanzahl: $y_l = y \cdot ratio^n$. Für die Höhe der letzten Zelle ergibt sich somit ein Wert von $y_l = 0,02448\ m$ (Im SnappyHexMeshDict aufgerundet auf 0,025 m).
Zeile 238 legt die minimale Schichthöhe fest. Diese wurde so festgelegt, dass der Y-Plus Wert nicht unter 20 fällt. Zur Abschätzung der Anzahl der Stützstellen in der wandnahen Schicht wird die Dicke der wandnahen Schicht nach [24] berechnet. Für die Grenzschichtdicke einer turbulenten Strömung folgt:

$$\delta = \frac{\nu}{u_m} 0,14 \frac{Re_x}{ln(Re_x)} 1,5. \tag{4.17}$$

Mit den Werten nach Kapitel 4.2.1, $Re_x = 3 \cdot 10^5$, $\nu = 1.532 \cdot 10^{-5}\ m^2/s$ und $u_m = 4,596$ m/s, beträgt die Schichtdicke demnach für den untersuchten Fall 16,65 mm. Mit dem eingestellten Zellwachstum von 1,05 existieren somit etwa 5 Stützstellen in der wandnahen Schicht und damit zu wenige im Vergleich zu der in [11] genannten Anzahl von mindestens 10 Stützstellen.
Ab Zeile 246 folgen erweiterte Einstellungen zur Netzqualität (Diese können im vollständigen SnappyHexMeshDict im Anhang A.2 betrachtet werden und entsprechen im wesentlichen den Werten aus den Tutorials). Besondere Beachtung gebührt allerdings Zeile 339-340:

```
339     //must be >0 for Fluent compatibility
340     minTriangleTwist 0.000001;
```

Da das Netz auch in Fluent verwendet werden soll, muss der Wert „MinTriangleTwist“ > 0 sein und wird daher vom Standardwert -1 auf den Wert $1 \cdot 10^{-6}$ verändert [23].

Um das Netz zu erstellen wird der Befehl:
snappyHexMesh > log.SnappyHexMesh&
im Projektordner im OpenFoam Terminal ausgeführt. Zur Vereinfachung der Netzerstellung ist es sinnvoll ein Ablaufskript zu schreiben. Beispiele dafür sind in den Tutorials in Form der „Allrun“-Skripte aufgeführt. Für den Kanalbogen befindet sich das Ablaufskript im Anhang A.3.
Das resultierende Netz zeigt Abbildung 4.13. Mit den gewählten Einstellungen hat das Netz

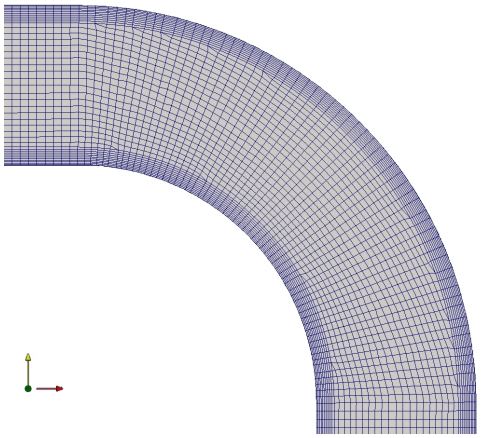


Abb. 4.13: Kanalbogen Berechnungsnetz, erstellt in SnappyHexMesh (mit Boundary Layern)

794328 Zellen. Die Netzqualität lässt sich mittels „checkMesh“ überprüfen. Im Anschluss können die qualitativ beanstandeten Zellen in ParaFOAM betrachtet und bewertet werden.

4.3.2 Freie Konvektion und Wärmeübergang an einer ebenen Wand

Im Gegensatz zum Kanalbogen wird das Berechnungsnetz für den vorliegenden Testfall, aufgrund der einfachen Geometrie (vgl. Kapitel 4.4.2 Abb. 4.19), ausschließlich in *Blockmesh* erstellt. Die Definitionen erfolgen dabei analog zu der Erstellung des Grundnetzes des Kanalbogens. Allerdings sind im Blockmesh die Randflächen für die Vergabe von Randbedingungen zu definieren:

```
40 boundary
41 (
42     floor
43     {
44         type wall;
45         faces
46         (
47             (1 5 4 0)
48         );
49     }
```

Dargestellt ist beispielhaft die Definition der Randfläche des Bodens (engl. floor). Zeile 42 benennt die Randfläche (hier floor). Dieser Name wird später bei der Zuweisung der Randbedingungen verwendet. Anschließend wird der Randflächentyp festgelegt (hier Wand, engl. wall). In Zeile 47 wird die Randfläche über die Eckpunkte definiert, wobei die Reihenfolge der Punkte

wichtig ist. Betrachtet man die Randfläche von außerhalb des Strömungsgebietes, so muss die Bennenung der Punkte in mathematisch positive Richtung gegen den Uhrzeigersinn erfolgen.
Um den Einfluss des Wandabstandes auf die Lösung zu untersuchen, werden verschieden feine Netze erstellt. Es wird ebenfalls das im ersten Testfall verwendete und verifizierte k-ω-SST-Turbulenzmodell mit Wandfunktionen genutzt. Bei der Wahl der ersten Zellhöhe an der Wand und damit der Definition des Y-Plus Wertes sind die Messdaten aus [3] nützlich (Abbildung 4.14).

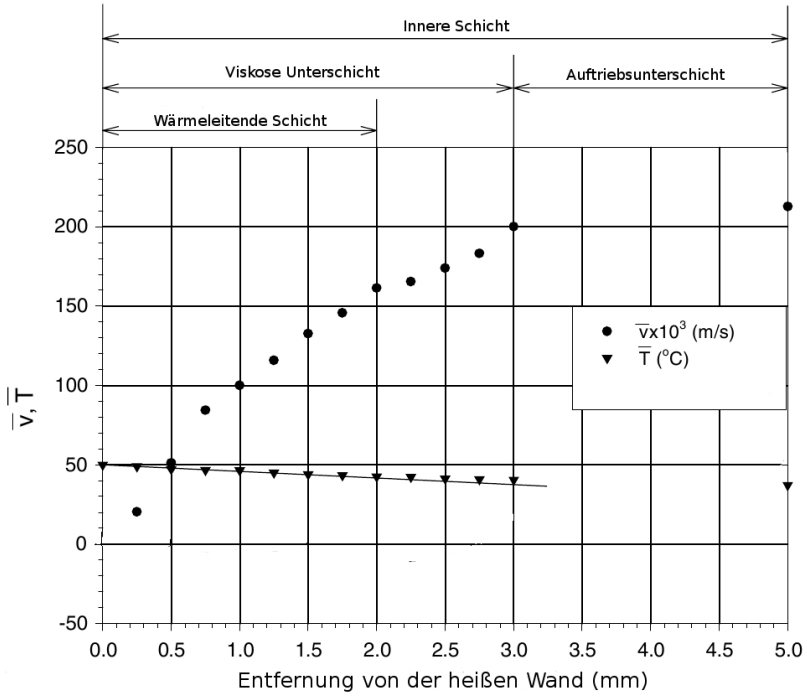


Abb. 4.14: Grenzschichtbereich bei Y=0,5 (modifiziert nach [3])

Dargestellt sind die gemittelten Werte der Strömungsgeschwindigkeit und der Temperatur im wandnahen Bereich. Es zeigt sich, dass die viskose Unterschicht bei etwa 3 mm endet. Damit die erste Zellschicht nicht in die viskose Unterschicht hineinreicht, sollte die erste Zellhöhe somit größer 3 mm sein.
Um den Einfluss des Wandabstandes auf die Lösung zu untersuchen, werden verschieden feine Netze erstellt (Siehe Tabelle 4.3).

Tab. 4.3: Berechnungsnetze für die Untersuchung der natürlichen Konvektion

Netz (x-,y-,z-Richtung)	Anzahl Netzzellen	min. Kantenlänge in m	max. Kantenlänge in m	Wandabstand in m
Netz 1 (25 x 25 x 50)	31250	0,03	0,03	0,03
Netz 2 (50 x 50 x 100)	250000	0,003	0,015	0,003
Netz 3 (75 x 75 x 75)	421875	0,01	0,02	0,01
Netz 4 (125 x 125 x 150)	2343750	0,006	0,01	0,006

Das Berechnungsnetz 1 ist mit einem Wandabstand von 30 mm grob aufgelöst. Der Y-Plus Wert kann ebenfalls der Studie [3] entnommen werden und beträgt in diesem Fall etwa 60. Netz 2 weist dabei neben der Verfeinerung eine Verdichtung der Zellen in Richtung Wand von 0,2 auf, wodurch die Zellenhöhe zum Rand hin abnimmt und eine größere Anzahl an Stützstellen im wandnahen Bereich vorhanden ist. Allerdings ist das innere Berechnungsgebiet nicht sehr viel feiner aufgelöst als beim ersten Berechnungsnetz. Dieses Vorgehen macht es erforderlich das Berechnungsgebiet in 4 Blöcke zu zerlegen (vgl. Anhang B.1). Der Wandabstand von Netz 2 liegt am Ende der viskosen Unterschicht bei 3 mm (Y-Plus etwa 6). Das dritte Netz liegt mit einem Wandabstand von 10 mm etwa bei einem Y-Plus-Werte von 25 und somit nahe dem empfohlenen dimensionslosen Wandabstand für das k-ω-SST-Turbulenzmodell von Y-Plus gleich 30. Das vierte Netz löst das Berechnungsgebiet nochmals feiner auf und erreicht ohne Abstufung der Netzzellen zur Wand hin einen Y-Plus-Wert von etwa 12 (6 mm Wandabstand).

4.3.3 Thermisch induzierter Auftriebsstrahl und Rauchschichtung

Für die Geometrieerstellung und Vernetzung wird wie beim Kanalbogen eine Kombination aus Salomé, BlockMesh und SnappyHexMesh verwendet. Das resultierende Netz zeigt Abb. 4.15.

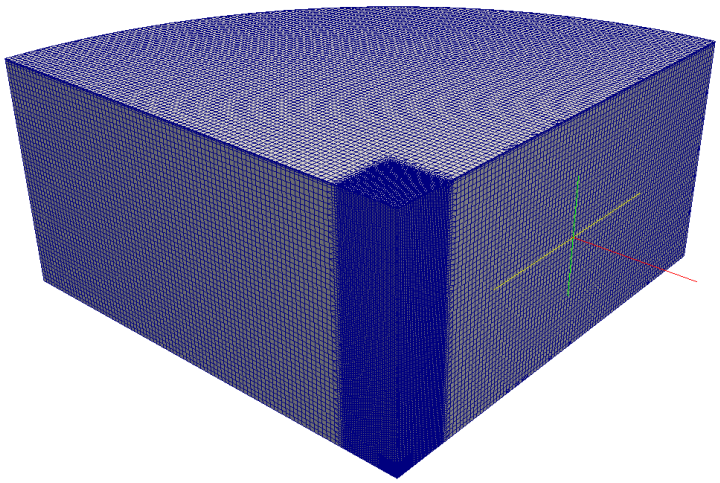


Abb. 4.15: Darstellung des Berechnungsnetzes für den dritten Testfall

Die maximale Zellweite von 144 mm orientiert sich an dem in [30] im Rahmen einer Netzunabhängigkeitsstudie ausgewählten Berechnungsnetz. Im Bereich des Strahlvolumens und der Wärmequelle, in denen die größten Strömungs-Gradienten erwartet werden, erfolgt eine Verfeinerung des Netzes. Die Verfeinerung wird im SnappyHexMeshDict definiert. Als Verfeinerungsgebiet können standardmäßig Blöcke, Zylinder und Kugeln vorgegeben werden. Weiterhin bietet SnappyHexMesh die Möglichkeit STL-Geometrien als Vernetzungsgebiet vorzugeben. Der Bereich im inneren der STL-Oberfläche wird dabei einem Zellen-Satz zugewiesen. Bei dieser Vorgehensweise wird das Netz an den STL-Oberflächen getrennt und es entstehen interne Oberflächen. Versuche mit dieser Vorgehensweise zeigten, dass die internen Oberflächen

die Berechnung beeinflussen können. So zeigten sich an den Übergängen Schwankungen der Geschwindigkeiten.
Die beste Zellqualität wird bei einer blockförmigen Verfeinerungen erwartet und daher auf diese Art durchgeführt (Vgl. Abb. 4.15). Die Verfeinerung ergibt als kleinste Zellweite im Bereich der Wärmequelle ein Wert von 0,0125 m, der nahe dem in [30] genannten Wert von 0,01 m liegt.
Eine weitere Schwierigkeit ergibt sich bei der Erstellung der Wandgrenzschicht an der Decke. Bei der Erstellung der Grenzschicht analog zum Kanalbogen in SnappyHexMesh zeigte sich, dass die Grenzschicht nicht bis in die Ecken durchgezogen wird. Die Netzqualität an der Absaugöffnung ist daher im Bereich der Decke nicht ausreichend hoch. Die Wandgrenzschicht wird deshalb nicht mit SnappyHexMesh, sondern mit dem Tool „refineWallLayer“ erstellt. Dabei werden die Netzzellen, die an die Decke angrenzen, normal zur Decke geteilt. Das Tool wird mehrfach angewendet, bis die gewünschte Feinheit erreicht ist. Mit diesem Tool kann allerdings nicht die Dicke der Grenzschicht beliebig variiert werden, da das Tool an die erste Zellhöhe des Ausgangsnetzes gebunden ist. Die Parameter des erstellten Berechnungsnetzes können Tabelle 4.4 entnommen werden.

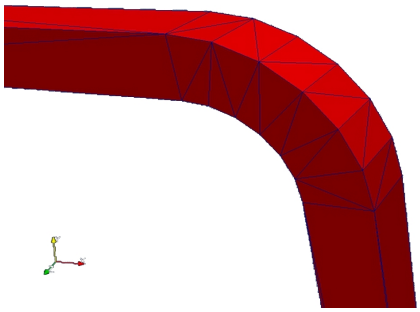
Tab. 4.4: Berechnungsnetz für die Untersuchung der Rauchausbreitung

Anzahl Netzzellen	min. Kantenlänge in m	max. Kantenlänge in m	Erste Zellhöhe Decke in m
1819772	0,0125	0,01	0,003

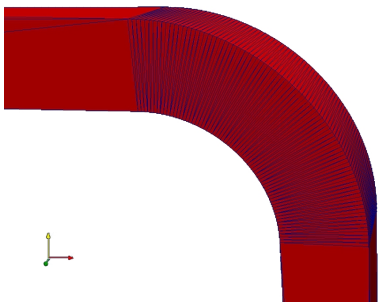
4.3.4 Bewertung der Geometrieerstellung und Vernetzung

Salomé

Analysiert man die in Salomé mit den Standardeinstellungen erstellte und als STL exportierte Geometrie des Kanalbogens in ParaView, so zeigt sich, dass die Auflösung der Oberflächen nicht hoch genug ist. Problematisch ist dieses im Bereich des Bogens (Abbildung 4.16a).



(a) STL-Auflösung mit Deflection coefficient 10^{-4}



(b) STL-Auflösung mit Deflection coefficient 10^{-6}

Abb. 4.16: STL-Dateien des Kanalbogens exportiert aus Salomé

Der Bogen wird lediglich durch 12 Dreiecke dargestellt. Dadurch ergeben sich Kanten in der Bogengeometrie, die sich auch im erstellen Berechnungsnetz zeigen und die Lösung beeinflussen können. Um die STL-Qualität zu verbessern ist es erforderlich, die Kanten höher aufzulösen. Dafür wird in Salomé der Beugungs-Koeffizient (engl.: Deflection coefficient) angepasst (Vom Standard Wert 10^{-4} auf den Wert 10^{-6}). Zu finden ist diese Einstellung in Salomé unter *File* $\rightarrow$ *Preferences*.
Werden die STL-Dateien mit dem kleineren Beugungs-Koeffizienten exportiert, so ist auch die Auflösung der STL Flächen deutlich feiner (siehe Abbildung 4.16b). Die Benutzerfreundlichkeit von Salomé ist weiterhin durch die folgenden Punkte eingeschränkt:

- Der Export der STL-Dateien, die Umbenennung von Hand und das anschließende Zusammenfügen sind mühsam und zeitintensiv.
- Skizzen können nach dem Beenden nicht editiert werden.
- Definition der Punkte einer Skizze durch Koordinaten ist zeitaufwendig und fehleranfällig. Deutlich schneller und intuitiver ist eine Feature-Basierte Modellierung, wie beispielsweise in Solid Works, Catia oder dem ANSYS Design Modeller.
- Die Dokumentation zu Salomé ist zu knapp.

BlockMesh und SnappyHexMesh

Die Vernetzung mittels BlockMesh und SnappyHexMesh funktioniert gut. Die Definition der Boundary Layer ist allerdings schwierig und es lässt sich schwer steuern, wie viele Zellschichten vorhanden sein sollen. Auch die Qualität der Layer könnte besser sein. Erst nach vielen Versuchen und verschiedensten Einstellungen konnte eine annehmbare Grenzschicht für den Kanalbogen erzeugt werden. Allerdings konnte auch in dieser Grenzschicht die geforderte Anzahl an Teilschichten in der logarithmischen Schicht nicht erreicht werden. Ähnlich verhielt es sich beim dritten Testfall. Eine Grenzschicht im Bereich der Decke konnte nicht mittels SnappyHexMesh, sondern erst im Anschluss an die Vernetzung mittels des Tools „refineWallLayer“ erzeugt werden.
Ein weiterer Schwachpunkt ist in der Zuweisung der Flächen für Randbedingungen in SnappyHexMesh zu sehen. Bei einer Betrachung der Geometrie des Modells für die Simulation des dritten Testfalls in ParaFOAM zeigt sich, dass die Netzzellen an den STL-Randflächen nicht getrennt werden, sondern nur die letzte Zelle an der STL-Oberfläche der Randfläche zugeordnet wird und die Vernetzung somit Patch-Unkonform erfolgt. Abbildung 4.17 zeigt dieses Phänomen für die Absaugöffnung.
Dadurch bedingt können die Randflächen zu klein ausfallen, so ist im dritten grundlegenden Testfall die Fläche der Absaugöffnung etwa 4 m^2 zu klein.

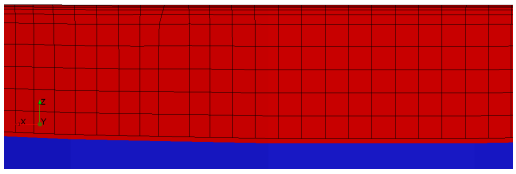


Abb. 4.17: Darstellung der fehlerhaft zugewiesenen Randfläche. (Rot=STL-Oberfläche der Absaugöffnung, blau=STL-Oberfläche der Wand, Netz= in OpenFOAM zugewiesese Absaugöffnung

4.4 Numerische Strömungsmodelle

Nach der Geometrieerstellung und Vernetzung werden an dieser Stelle die verwendeten Strömungsmodelle vorgestellt.

4.4.1 Turbulente Durchströmung eines Kanalbogens

Der Dichteeinfluss lässt sich anhand der Mach-Zahl Ma bestimmen. Sie stellt das Verhältnis der Strömungs- zur Schallgeschwindigkeit dar.

$$Ma = \frac{u_m}{c} \tag{4.18}$$

Für Strömungen mit Ma < 0,3 kann man das Gas als dichtebeständig ansehen [27]. Mit der nach Gleichung (4.16) bestimmten Geschwindigkeit $u_m = 4,596$ m/s und der Schallgeschwingikeit von Luft von 331 m/s ergibt sich eine Machzahl von 0,0139. Die Strömung kann daher als inkompressibel angenommen werden. Weiterhin ist die Temperatur der Strömung konstant (isotherm). Somit ist neben der Dichte auch die Viskosität der Strömung konstant [11].
Für die Randbedingungen reicht im Falle einer inkompressiblen isothermen Strömung die Vorgabe der Geschwindigkeit, des Druckes und der Viskosität aus. Wird weiterhin eine turbulente Strömung untersucht ($Re > Re_{krit}$, siehe Gl. (4.1)) und soll diese Strömung nicht durch eine direkte numerische Simulation (DNS) erfolgen, werden weitere Bedingungen für die Turbulenz benötigt. Im untersuchten Fall und in den darauf aufbauenden Simulationen wird das k-ω-SST-Modell aufgrund der in Kapitel 2.3.2 genannten Vorteile verwendet.
Am Einlass, 10 m vor dem Kanalbogen, wird ein Kolbenprofil konstanter Geschwindigkeit sowie k und ω entsprechend der Turbulenzintensität vor dem Kanalbogen vorgegeben. Nach [11] lässt sich k folgendermaßen abschätzen:

$$k = \frac{3}{2} \cdot {I_t}^2 \cdot {u_m}^2. \tag{4.19}$$

Dabei entspricht I_t der Turbulenzintensität in %. Mit der Vorgabe der Turbulenzintensität vor dem untersuchten Kanalbogen von 1% (vgl. [16]) und der Geschwindigkeit von 4,596 m/s ergibt sich für k ein Wert von $3,185 \cdot 10^{-3}$ $\frac{m^2}{s^2}$. Mit der Annahme, dass die turbulente kinematische Viskosität ν_t etwa dem 10 fachen Wert der kinematischen Viskosität entspricht (vgl. [11]) lässt

sich ω folgendermaßen berechnen:

$$\omega = \frac{k}{\nu_t}. \tag{4.20}$$

Mit der kinematischen Viskosität $\nu = 1,532 \cdot 10^{-}5\ \frac{m^2}{s}$ ergibt sich für $\omega = 20,793\ \frac{1}{s}$. Am Ausströmrand in einiger Entfernung nach dem Bogen wird ein Druckauslass durch die Vorgabe eines konstanten Druckes, entsprechend dem atmosphärischen Druck, über dem Ausströmrand definiert.
In CFD-Simulationen der Gebäudetechnik ist es sinnvoll transiente Strömungen zu betrachten. Daher wird die Strömung transient berechnet und im eingeschwungenen Zustand ausgewertet. Für die vorgestellten Modellannahmen eignet sich in OpenFoam der Solver *pisoFoam* mit Turbulenzmodell k-ω-SST. Die Ordnerstruktur dieses Solvers vor der Berechnung zeigt Abbildung 4.18.

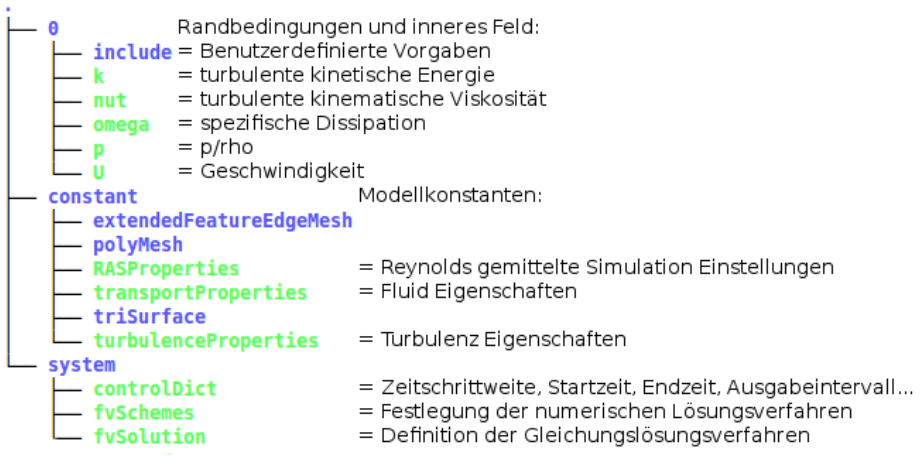


Abb. 4.18: Ordnerstruktur des Solvers PimpleFoam

Im Ordner „0“ werden die initialen Werte definiert. Die festgelegten Werte können Tabelle 4.5 entnommen werden.

Tab. 4.5: Start- und Randbedingungen für den ersten Testfall

	Einlass	Auslass	Wände
p in Pa	kein Gradient	Fester Wert: 0	kein Gradient
k in $\frac{m^2}{s^2}$	Fester Wert: 0,003185	kein Gradient	Wandfunkt., Start 0,003185
omega in $\frac{1}{s}$	Fester Wert: 20,793	kein Gradient	Wandfunkt., Start 20,793
nut in $\frac{kg}{ms}$	berechnet	berechnet	Wandfunkt., Start 0
U in $\frac{m}{s}$	fester Wert: U_x=4,596	kein Gradient	fester Wert: 0

Im Unterordner „include“ werden die vorgegebenen Werte der Geschwindigkeit und der Turbulenz am Einlass definiert. Dadurch können spätere Änderungen schneller eingepflegt werden.
Im Ordner „Constant“ befinden sich neben dem Berechnungsnetz Dateien zur Festlegung des

Turbulenzmodells und der Fluideigenschaften. Für die Fluideigenschaften reicht im vorliegenden Fall einer inkompressiblen Strömung die Definition eines Newtonschen-Fluid-Modells und die Angabe der kinematischen Viskosität aus. Da kein Referenzdruck angegeben ist, müssen die berechneten Drücke bei der Auswertung allerdings noch mit der Standarddichte multipliziert werden.
Im Ordner „system“ wird die Berechnung gesteuert, durch z.B. die Vorgabe der Zeitschritte sowie der Diskretisierungs- und Lösungsverfahren. Für die Diskretisierungsverfahren werden Verfahren zweiter Ordnung gewählt (z.B. Second Order Upwind), die eingestellten Lösungsverfahren orientieren sich an den Tutorials.
Die Zeitschrittweite orientiert sich an der Anforderung an die CFL-Zahl:

$$CFL = \frac{\Delta t \cdot u_m}{\Delta x} \leq 1. \tag{4.21}$$

Mit einer Zellweite in Hauptströmungsrichtung von Δt=0,05 m und der Geschwindigkeit u_m von $4,596\frac{m}{s}$ und einer angestrebten CFL Zahl von 1 ergibt sich für den Zeitschritt ein Δt von 0,01 s.
Für die Berechnung in Fluent wird das Netz mit dem OpenFOAM-Werkzeug „FOAMMeshToFluent“ in das Fluent-Format konvertiert. Die Einstellungen des Strömungsmodells in Fluent entsprechen den Angaben in OpenFOAM.

4.4.2 Freie Konvektion und Wärmeübergang an einer ebenen Wand

Das Berechnungsgebiet mit den Randbedingungen zeigt Abbildung 4.19.

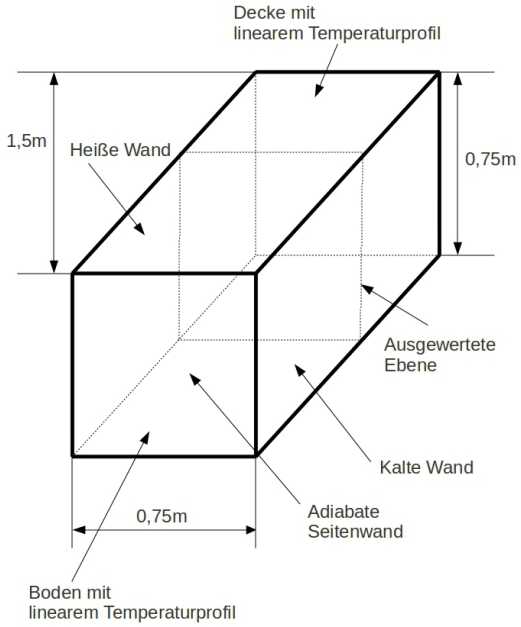


Abb. 4.19: Dreidimensionale schematische Darstellung des untersuchten luftgefüllten Raumes (modifiziert nach [3])

Die warme und die kalte Wand sind isotherm, die vordere und hintere Wand adiabat. Um den realen Randbedingungen der hoch wärmeleitenden Decke und Boden zu entsprechen sind diese, wie in [3] empfohlen, mit einem linearen Temperaturprofil versehen. Die im CFD-Modell konstanten Stoffwerte der Luft werden für die mittlere Temperatur $\frac{\vartheta_h+\vartheta_c}{2} = 30°C$ [17] entnommen (Siehe Tabelle 4.6).

Tab. 4.6: Stoffwerte der Luft beim Druck p=1 Bar und der mittleren Wärmeübertragungstemperatur $\vartheta_{mittel} = 30°C$ nach [17]

Mittlere Molmasse nMol in g/mol	Mittlere Wärmekapazität cp in J/(kg K)	dynamische Viskosität μ in Pa*s	Prandtl-Zahl Pr
28,9	1007	$1,8 * 10^{-5}$	0,71

Die mittels der Stoffwerte aus [17] und Gleichung (4.7) berechnete Rayleight-Zahl beträgt $1,46 \cdot 10^9$ und ist mit der im Experiment angegebenen Ra-Zahl von $1,58 \cdot 10^9$ vergleichbar. Für dieses Strömungsproblem eigenet sich nach [23] der Solver *BuoyantPimpleFoam*. Die Ordnerstruktur zeigt Abbildung 4.20.

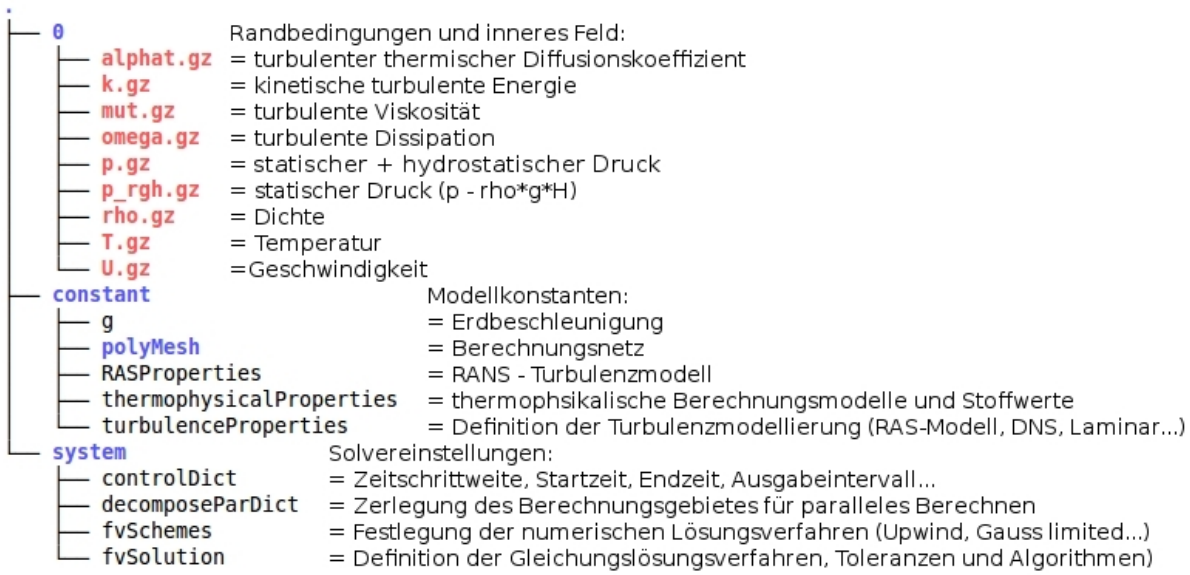


Abb. 4.20: Ordnerstruktur des Solvers BuoyantPimpleFoam

Im Vergleich zum bei der Simulation des Kanalbogens genutzten Solver *PisoFoam* werden im 0-Ordner mit der turbulenten thermischen Diffusion, der Dichte, der dynamischen Viskosität, der Temperatur und dem hydrostatischen Druck weitere Anfangswerte und Randbedingungen festgelegt.

Die Startbedingungen für die turbulenten Größen k und Omega können nach Gleichung (4.19) und (4.20) berechnet werden. Die Turbulenzintensität wird dabei mit 5 % angenommen und als Referenzgeschwindigkeit wird die Auftriebsgeschwindigkeit u_B eingesetzt, die nach [3] $u_B = 1\ m/s$ beträgt.

Für die Definition des linearen Temperaturprofils am Boden und der Decke wird das Tool *FunkySetFields* aus dem Toolset *Swak4Foam* (Swiss Army Knife for Foam, bzw. auf Deutsch: „Schweizertaschenmesser für Foam“) verwendet. Mit diesem Toolset ist es möglich ohne C++ Programmierung Randbedingungen, Felder, Quellen usw. zu setzen, die über die im Standard OpenFOAM-Paket enthaltenen Möglichkeiten hinausgehen. Der Download und Hinweise zur Kompilierung, sowie zur Anwendung des Toolsets können unter [14] gefunden werden. Die Gleichung für den linearen Temperaturverlauf wird folgendermaßen gesetzt (Konsolenbefehle):

```
1 funkySetFields -time 0 -field T -keepPatches -valuePatches "floor ceiling" -condition "pos().x
    >0 && pos().x<0.75 && pos().z>0 && pos().z<1500 && pos().y<0.75" -expression "-53.333*pos
    ().x+323.15"
2 funkySetFields -time 0 -field T -keepPatches -expression "303.15"
```

Der Befehl in der ersten Zeile setzt für das Temperaturfeld zum Zeitpunkt „0“ die Gleichung für das lineare Temperaturprofil ein. ”valuePatches“ legt fest, dass die Decke und der Boden mit dem linearen Temperaturprofil versehen werden. Weiterhin werden die Koordinatenbegrenzungen definiert. *FunkySetFields* legt das lineare Temperaturprofil neben der Decke und dem Boden für das gesamte innere Feld fest. Um den Startwert für das innere Feld wieder auf die mittlere Temperatur zu setzen ist der zweite Befehl erforderlich. Die festgelegten Werte können zusammenfassend Tabelle 4.7 entnommen werden.

Tab. 4.7: Start- und Randbedingungen für den zweiten Testfall

	Decke & Boden	Heiße Wand	kalte Wand	Seitenwände	inneres Feld
T in K	-53,333* pos().x +323,15	323,15	283,15	adiabat	303,15
p in Pa	berechnet, Start 101325	berechnet, Start 101325	berechnet, Start 101325	berechnet, Start 101325	101325
p_rgh in Pa	berechnet, Start 101325	berechnet, Start 101325	berechnet, Start 101325	berechnet, Start 101325	101325
alphat in $\frac{kg}{ms}$	Wandfunkt., Start 0	Wandfunkt., Start 0	Wandfunkt., Start 0	Wandfunkt., Start 0	0
k in $\frac{m^2}{s^2}$	Wandfunkt., Start 0,00375	Wandfunkt., Start 0,00375	Wandfunkt., Start 0,00375	Wandfunkt., Start 0,00375	0,00375
omega in $\frac{1}{s}$	Wandfunkt., Start 24,477	Wandfunkt., Start 24,477	Wandfunkt., Start 24,477	Wandfunkt., Start 24,477	24,477
mut in $\frac{kg}{ms}$	Wandfunkt., Start 0	Wandfunkt., Start 0	Wandfunkt., Start 0	Wandfunkt., Start 0	0
U in $\frac{m}{s}$	Wandfunk., Start 0	Wandfunkt., Start 0	Wandfunkt., Start 0	Wandfunkt., Start 0	0

Die thermophysikalischen Eigenschaften werden in der Datei ”thermophysicalProperties“ festgelegt:

```
18 thermoType           hRhoThermo<pureMixture<constTransport<specieThermo<hConstThermo<perfectGas
    >>>>>;
```

```
19
20 pRef              101325;
21
22 mixture
23 {
24     specie
25     {
26         nMoles          1;
27         molWeight       28.9;
28     }
29     thermodynamics
30     {
31         Cp              1007;
32         Hf              0;
33     }
34     transport
35     {
36         mu              1.8e-05;
37         Pr              0.7;
38     }
39 }
```

”ThermoType“ (Zeile 18) legt die thermophysikalischen Berechnungsmodelle fest. Eine Übersicht und Erklärungen zu den genutzten Modellen gibt Tabelle 4.8.

Tab. 4.8: Thermophysikalische Fluideigenschaften [23]

Modell	Erklärung
hRhoThermo	Allgemeines thermophysikalisches Berechnungsmodell basierend auf der Enthalpie h
pureMixture	Allgemeines thermophysikalisches Berechnungsmodell für passive Gasgemische
constTransport	Konstante Transporteigenschaften (Viskosität, Wärmekapazität, Wärmeleitfähigkeit, Wärme-Diffusion)
specieThermo	Thermophysikalische Eigenschaften der Spezies, abgeleitet von cp,h und/oder s.
hConstThermo	Konstantes cp Model mit der Entwicklung der Enthalpie h und Entropie s.
perfectGas	Verwendung der Zustandsgleichungen für ideale Gasgemische.

Weiterhin werden der Referenzdruck und die entsprechenden Stoffwerte nach Tabelle 4.6 gesetzt. Als Turbulenzmodell wird das im ersten Testfall validierte k-ω-SST-Turbulenzmodell mit Wandfunktionen verwendet.

4.4.3 Thermisch induzierter Auftriebsstrahl und Rauchschichtung

Das numerische Strömungsmodell orientiert sich an der Studie [30]. Die ausgewählte Modellvariante mit den Bezeichnungen der Begrenzungsflächen zeigt Abbildung 4.21. Der Absaug-

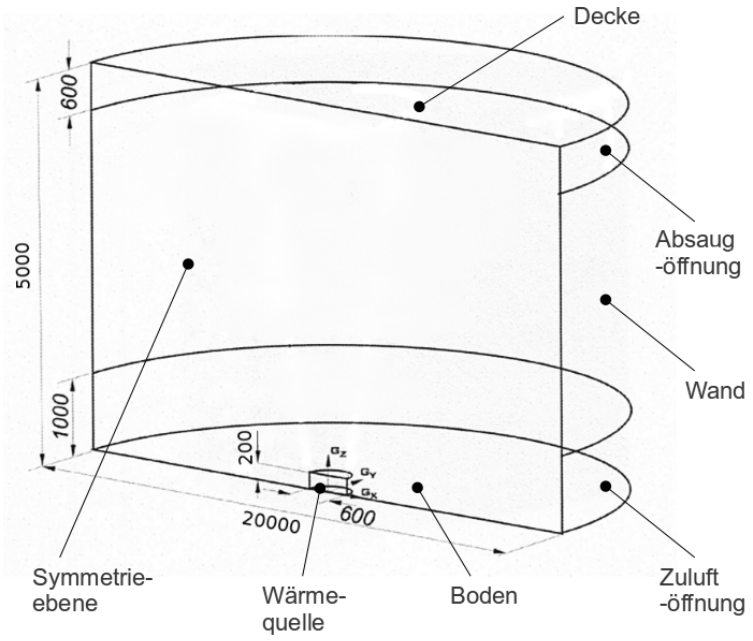


Abb. 4.21: Abmessungen des ausgewählten Berechnungstestfalls mit Randbedingungen (modifiziert nach [30])

volumenstrom beträgt 6500 m^3/h. Über $c = \dot{V}/A$ lässt sich die mittlere auf der Absaugöffnung festzulegende Geschwindigkeit berechnen, die normal zur Oberfläche der Absaugöffnung vorgegeben wird. Die weiteren Randbedingungen sind Tabelle 4.9 zu entnehmen.

Tab. 4.9: Start- und Randbedingungen für den dritten Testfall

	Boden, Decke und Wand	Zuluftöffnung	Absaugöffnung
T in K	kein Gradient	Einlass-Auslass-RB, fester Wert 293,15	Einlass-Auslass-RB, fester Wert 293,15
p in Pa	berechnet, Start 101325	berechnet, Start 101325	berechnet, Start 101325
p_rgh in Pa	Auftriebsdruck-RB, Start 101325	Totaldruck-RB, Start 101325	Auftriebsdruck-RB, Start 101325
alphat in $\frac{kg}{ms}$	Wandfunkt., Start 0	berechnet, Start 0	berechnet, Start 0
k in $\frac{m^2}{s^2}$	Wandfunkt., Start $3{,}44 \cdot 10^{-9}$	Fester Wert: $3{,}44 \cdot 10^{-9}$	kein Gradient
omega in $\frac{1}{s}$	Wandfunkt., Start 0,002246	Fester Wert: 0,002246	kein Gradient
mut in $\frac{kg}{ms}$	Wandfunkt., Start 0	berechnet, Start 0	berechnet, Start 0
U in $\frac{m}{s}$	Wandfunk., Start 0	berechnet, Start 0	Oberflächen-Normal-Fester-Wert-RB, 0,047893

Für die Definition der volumetrischen Wärmequelle wird der im zweiten Testfall genutzte Sol-

ver *buoyantPimpleFOAM* angepasst. Die Energiegleichung (in Enthalpieform; Enthalpie=h) wird dabei um einen Quellterm erweitert:

$$\frac{\partial}{\partial t}(\rho h) + \nabla \cdot Uh - \nabla \cdot \lambda \nabla h = 0 \longrightarrow \frac{\partial}{\partial t}(\rho h) + \nabla \cdot Uh - \nabla \cdot \lambda \nabla h = Q_h. \tag{4.22}$$

Die Solver-Dateien werden dafür in den User-Ordner kopiert, umbenannt, angepasst und neu kompiliert (vgl. Anhang C.3 - C.4). Der Solvercode wird in der Datei hEqn angepasst, die die allgemeine Transportgleichung für die Energie enthält (Anhang C.2). Weiterhin wird die Datei createFields.H um das skalare Feld Q_h erweitert (Anhang C.3). Dieses Feld definiert explizit die konstanten Werte des Quellterms Q_h für die Zellen (Einheit W/m^3). Die Definition der konstanten Werte wird über das Tool *SetFiels* mittels des *SetFieldsDictionary* im Systemordner durchgeführt (Anhang C.4). Dabei wird ein Zylinder entsprechend des Volumens der Wärmequelle definiert und den Zellen in diesem Volumen der konstante Wert der Wärmequelle vorgegeben. Dieser Wert ergibt sich aus der geplanten Wärmeabgabe der Quelle von $Q = 10\ kW$ und dem Volumen der Wärmequelle von $V = \frac{\pi}{4} d^2 h = 0{,}056549\ m^3$ zu $Q_h = \frac{Q}{V} = 176838\ \mathrm{W}/m^3$.
Die thermophysikalischen Eigenschaften entsprechen denen des zweiten Testfalls (vgl. Tabelle 4.8). Die Stoffwerte der Luft sind entsprechend [30] gewählt und Tabelle 4.10 zu entnehmen.

Tab. 4.10: Stoffwerte der Luft für den dritten Testfall nach [30]

Mittlere Molmasse nMol in g/mol	Mittlere Wärmekapazität cp in J/(kg K)	dynamische Viskosität μ in Pa*s	Prandl-Zahl Pr
28,97	1006	$1{,}82 * 10^{-5}$	0,7

Die Turbulenzmodellierung erfolgt wie bei den ersten Testfällen mit dem k-ω-SST-Turbulenzmodell. Bei den numerischen Untersuchungen erfolgt weiterhin keine Berücksichtigung des Wärmeaustausches durch Strahlung. Obwohl alle verwendeten Randbedingungen stationär sind wird abermals ein instationäres Verfahren angewendet. Dieses Vorgehen ist berechnungs- und zeitintensiver, verbessert aber die Konvergenz einer Berechnung, insbesondere bei auftriebsdominierenden Strömungen.
Dieser Testfall wird parallel auf vier Prozessoren berechnet. Dafür wird die „DecomposeParDict“ angepasst und ausgeführt. Das Berechnungsgebiet wird dabei in 4 Teile zerlegt, die jeweils von einem Prozessor berechnet werden. Gestartet wird die Berechnung mit dem Befehl „mpirun -np4 buoyantPimpleFoam -parallel > log.FOAM“. Nach der Berechnung müssen die Ergebnisse für die Betrachtung in ParaFoam wieder zusammengesetzt werden. Dieses erfolgt mit dem Befehl „reconstructPar“.

4.4.4 Bewertung der Strömungsmodellerstellung

Die Erstellung der Strömungsmodelle für die drei grundlegenden Testfälle konnte ohne größere Probleme durchgeführt werden. Sehr hilfreich waren dabei die mit OpenFOAM gelieferten Tutorials zu den Solvern und Tools. Für zu untersuchende Strömungsprobleme, die den Tutorials ähnlich sind, lassen sich somit recht schnell Strömungsmodelle aufsetzen. Schwieriger wird es bei relativ neuen Solvern und Randbedingungen, da zu diesen oftmals nur wenige Tutorials vorliegen. Für den im dritten Testfall verwendeten Solver *BuoyantPimpleFoam* beispielsweise gab es nur ein Tutorial für einen geschlossenen Raum ohne Ein- und Auslässe. In diesem Fall war es schwierig, die richtigen Bedingungen für den statischen Druck p_rgh und den hydrostatischen Druck p an der Absaugung und in der Domain zu ermitteln. Zwar ist der Code zu allen Tools, Randbedingungen, Solvern, usw. frei einsehbar, allerdings ist die Beschreibung oftmals nur sehr spärlich und der Programmablauf durch die objektbasierte Programmierung und den damit verbundenen Zugriff auf viele verschiedene Objekte an verschiedenen Orten relativ schwer nachzuvollziehen.
Mittels des Tools „Swak4Foam“, dass nicht Bestandteil des Programmpaketes von OpenFOAM ist, allerdings relativ einfach hinzu kompiliert werden konnte, ließ sich im zweiten Testfall relativ einfach die benutzerdefinierte Randbedingungen eines linearen Temperaturprofils setzen. Hilfreich ist, dass auch bei diesem Tool Tutorials mitgeliefert werden.
Weiterhin konnte im dritten Testfall der Solver erfolgreich um eine volumetrische Wärmequelle erweitert werden und auch die Definition der volumetrischen Wärmequelle über das Tool „SetFields“ war ohne Probleme möglich.
Ein Kritikpunkt neben der teilweise spärlichen Beschreibung ist allerdings, dass die Anzahl der Definitionsskripte mit steigender Komplexität des Solvers immer mehr zunehmen und es schwierig ist, den Überblick zu behalten. An dieser Stelle vermisst man relativ schnell ein gute grafische Benutzeroberfläche.

4.5 Validierung der Simulationen

Nachfolgend werden die Simulationsergebnisse aus OpenFoam für die drei Testfälle dargestellt und validiert. Die Validierungsmerkmale orientieren sich dabei an Kapitel 4.2.

4.5.1 Turbulente Durchströmung eines Kanalbogens

Ausgewertet wird ein Zeitschritt 20 s nach Beginn der Simulation. Zur Auswertung wird ParaFoam genutzt. Die Fluent-Daten werden daher als ParaFoam kompatibles Format (z.B. EnSight Case Gold) exportiert.

Druckverläufe

Zur Bestimmung der numerisch berechneten Druckverluste werden 20 Schnittebenen - 5 vor dem Bogen, 5 im Bogen und 10 nach dem Bogen - in den Strömungskanal gelegt (vgl. Anhang A.5). Auf diesen Ebenen wird das Flächen-Integral des Totaldruckes bestimmt und ausgewertet. Als qualitativer Vergleich dient das Diagramm nach [29] (vgl. Abb. 4.22).

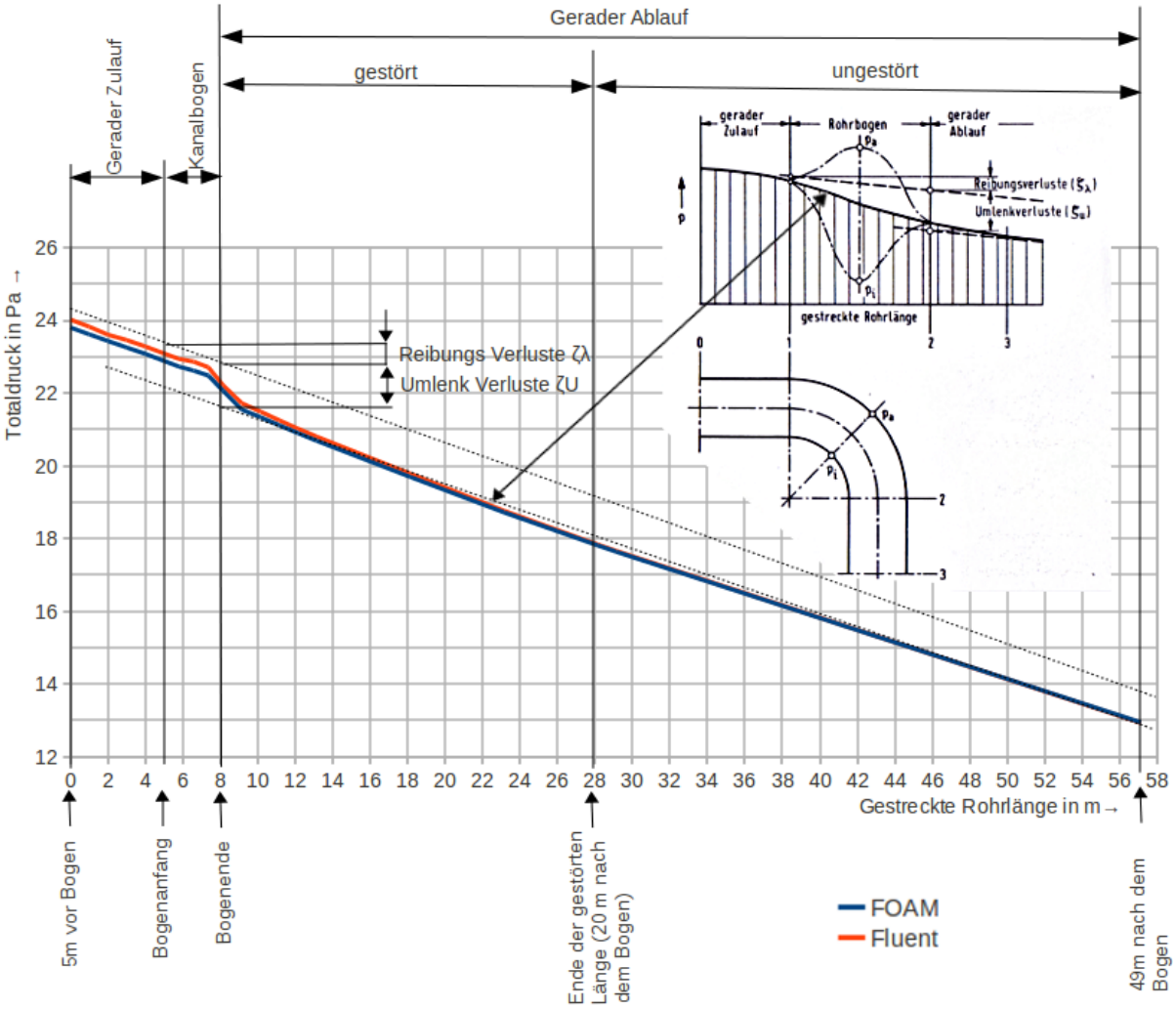


Abb. 4.22: Totaldruckverlust über die gestreckte Länge des untersuchten Kanalbogens. Abgebildeter Vergleich nach [29].

Demnach ist der Totaldruckverlust im geraden Zulauf des Kanalbogens anfangs linear (entsprechend dem Totaldruckverlust der Strömung durch ein gerades Rohr). Der erste Unterschied zeigt sich beim Eintritt in den Kanalbogen. Nach [29] und auch nach Abbildung 4.4 im Kapitel 4.1.1 beginnt der Druckabfall schon vor dem Bogen zu steigen und ist nicht mehr linear. Dieses Verhalten zeigt sich bei den numerisch berechneten Verläufen erst kurz vor dem Ende des Kanalbogens. Die gestörte Länge des geraden Ablaufs beträgt etwa 20 m. Auffällig ist weiterhin, dass die numerisch berechneten Werte aus Fluent und OpenFOAM vor dem Kanalbogen eine leichte Abweichung aufweisen, die etwa 20 m nach dem Kanalbogen nicht mehr vorhanden ist. Abbildung 4.23 zeigt die Verläufe der statischen Drücke an den Kanalbogenaußenflächen be-

rechnet in OpenFOAM und Fluent sowie einen Literaturvergleich nach [29].

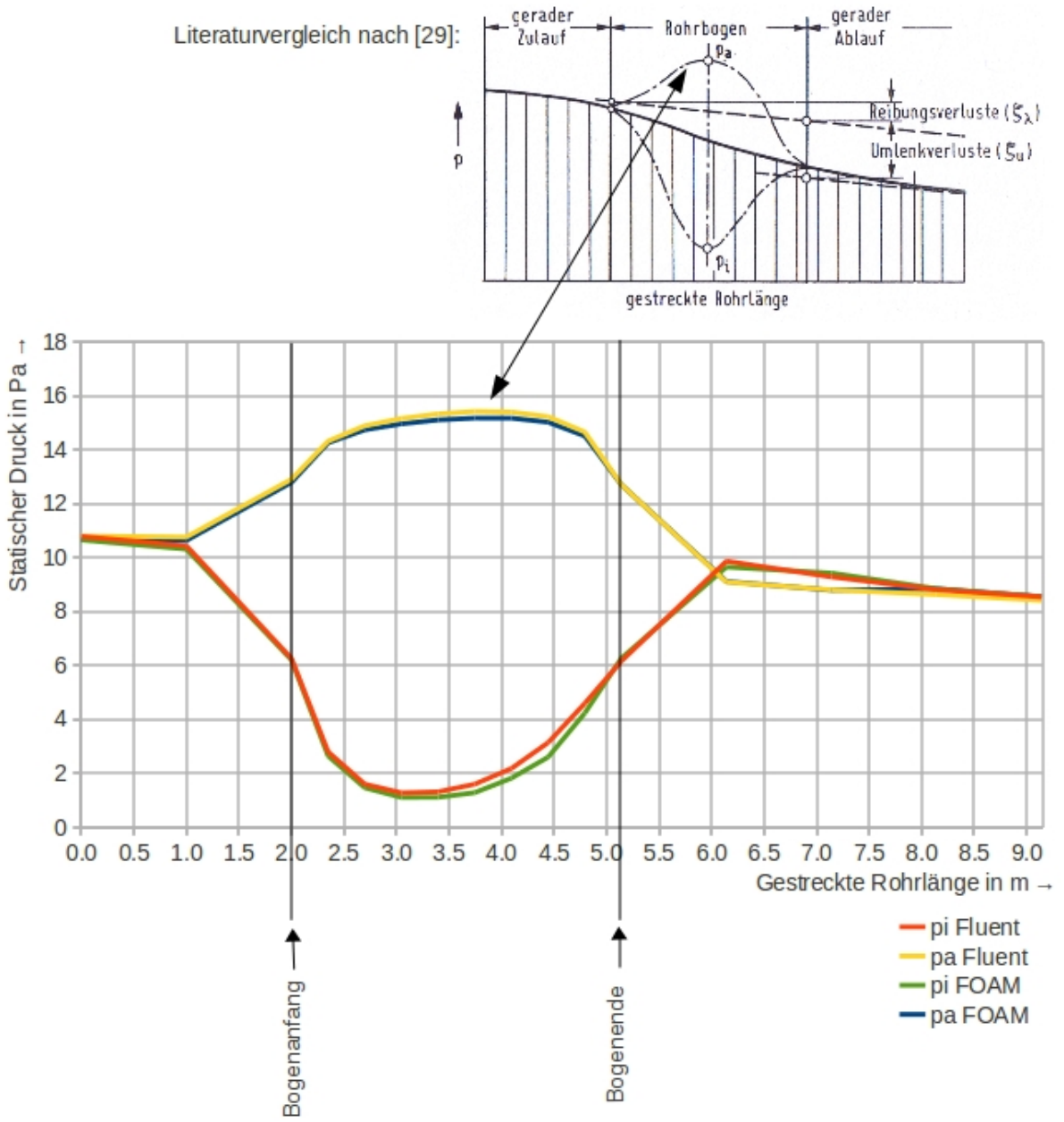


Abb. 4.23: Druckverteilung entlang der Kanalbogenaußenflächen berechnet in OpenFoam und Fluent. Literaturvergleich nach [29].

Die in Fluent und Foam berechneten Werte sind hier abermals sehr ähnlich. Abweichend zum in der Literatur gezeigten Verlauf nach [29] beginnt die Abweichung zwischen dem Druck auf der Innen- und Außenseite des Kanalbogens etwa einen Meter vor dem Bogen und endet etwa drei Meter nach dem Bogen. Ebenfalls differierend zu [29] fällt der Druck der Kanalbogenaußenseite in den Fluent und OpenFOAM Ergebnissen nach dem Kanalbogen kurzzeitig unter den Druck an der Kanalbogeninnenseite.

Druckverlustbeiwert

Als quantitativer Vergleich wird der Druckverlustbeiwert zwischen den Ebenen 2 m vor und 49 m nach dem Bogen bestimmt. Zwar ist im Diagramm 4.22 schon nach etwa 20 m kaum noch ein Einfluss des Bogens zu erkennen, bestimmt man aber die Druckverlustbeiwerte je 1 m-Kanalabschnitt nach Gleichung (4.11), liegen diese auch in 20 m Entfernung vom Kanalbogen noch etwas über dem Wert für ein gerades Kanalstück. Daher wird die maximal mögliche Länge von 51 m für die Bestimmung des Druckverlustbeiwertes gewählt.
Der theoretische Druckverlustbeiwert nach [16] setzt sich aus den Umlenkverlusten des Kanalbogens, den Reibungsverlusten des Kanalbogens und den Reibungsverlusten des geraden Zu- und Ablaufs zusammen. Der Druckverlustbeiwert des geraden Zu- und Ablaufs lässt durch zusammenführen der Gleichung (4.2) und Gleichung (4.11) durch

$$\zeta = \lambda \frac{l}{D_h} \tag{4.23}$$

berechnen. Mit der Länge $l = 51\ m$, $D_h = 1\ m$ und $\lambda = 0,015$ nach Kapitel 4.2.1 ergibt sich für die geraden Kanalstücke ein Druckverlustbeiwert von $\zeta = 0,766$. Mit dem Druckverlustbeiwert für den Kanalbogen von $\zeta = 0,196$ (vgl. Kapitel 4.2.1) ergibt sich somit ein gesamter theoretischer Druckverlustbeiwert zwischen den untersuchten Ebenen von 0,962. Die simulierten Druckverlustbeiwerte lassen sich nach [12] durch

$$\zeta_0 = \frac{\Delta p_{total}}{P_{dyn,0}} \tag{4.24}$$

bestimmen. $P_{dyn,0}$ ist der dynamische Druck der mittleren Strömungsgeschwindigkeit durch den Kanalbogen an der Stelle 0, in diesem Fall 2 m vor dem Kanalbogen. Die integrierten Drücke 2 m vor und 49 m nach dem Bogen sowie die aus den Simulationen berechneten Druckverlustbeiwerte zeigt Tabelle 4.11. Weiterhin sind die prozentualen Abweichungen der OpenFOAM und der Fluent Berechnungen aufgeführt.

Tab. 4.11: Totaldruckdifferenz (2 m vor, 49 m nach dem Bogen), dynamischer Druck 2 m vor dem Bogen, Druckverlustbeiwerte und Prozentuale Abweichung zum Theoriewert

	ΔP_{tot} in Pa	$P_{dyn,0}$ in Pa	ζ_{sim}	ζ_{the}	Abweichung in %
FOAM	10,3058	12,5524	0,821	0,962	14,65
Fluent	10,5265	12,5525	0,839	0,962	12,83

Die aus den Fluent und OpenFOAM Ergebnissen berechneten Druckverlustbeiwerte sind ähnlich, weisen allerdings eine nicht unerhebliche Abweichung zum theoretischen Druckverlustbeiwert auf. Eine Erklärung hierzu kann die nachfolgende Betrachtung der Geschwindigkeiten und der Ablösegebiete liefern.

Geschwindigkeiten und Ablösegebiete

Als weiteren qualitativen Vergleich werden verschiedene Geschwindigkeitsmerkmale ausgewertet. Es werden Geschwindigkeitsprofile, sowie Vektorplots auf verschiedenen Ebenen betrachtet.
Abbildung 4.24 zeigt den Geschwindigkeitsverlauf im Kanalbogen. Auf der Kanalbogeninnen-

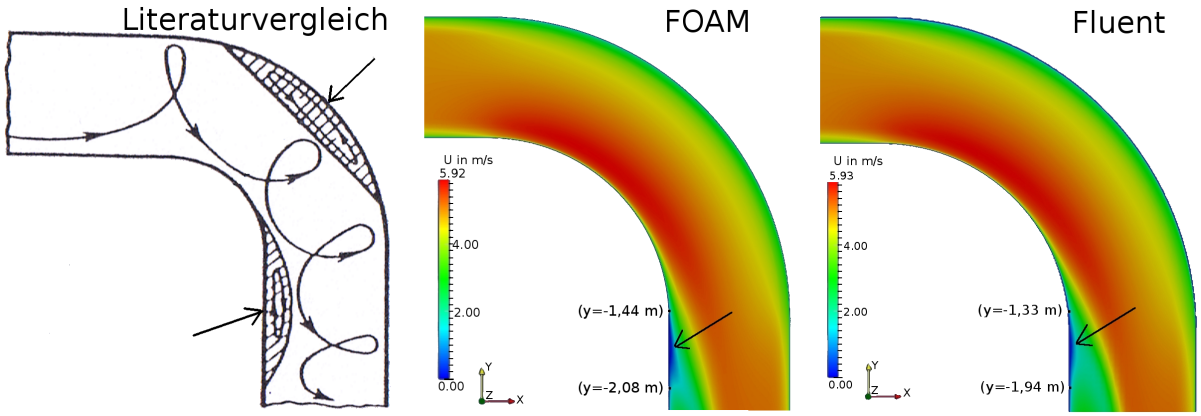


Abb. 4.24: Darstellung der Geschwindigkeitsverteilung im Bogen mit Angabe der y-Koordinate des Ablöse- und des Wiederanlegepunktes der Simulationen (Pfeile markieren die Ablösegebiete). Der Literaturvergleich nach [16] zeigt die Strömungsverhältnisse im Bogen mit der Darstellung der Ablösegebiete und einer Stromlinie.

seite ist in den Simulationen ein Gebiet einer sehr geringen Geschwindigkeit zu erkennen (Mittels Pfeil markiert). Dieses Gebiet beginnt in der Fluent-Berechnung etwa 0,11 m früher und endet ebenfalls früher. Entsprechend andersartig sind die Geschwindigkeitsprofile 0,5 m nach dem Austritt (Abb. 4.25), bei nahezu identischen Geschwindigkeitsprofilen zwei Meter vor und drei Meter nach dem Kanalbogen.

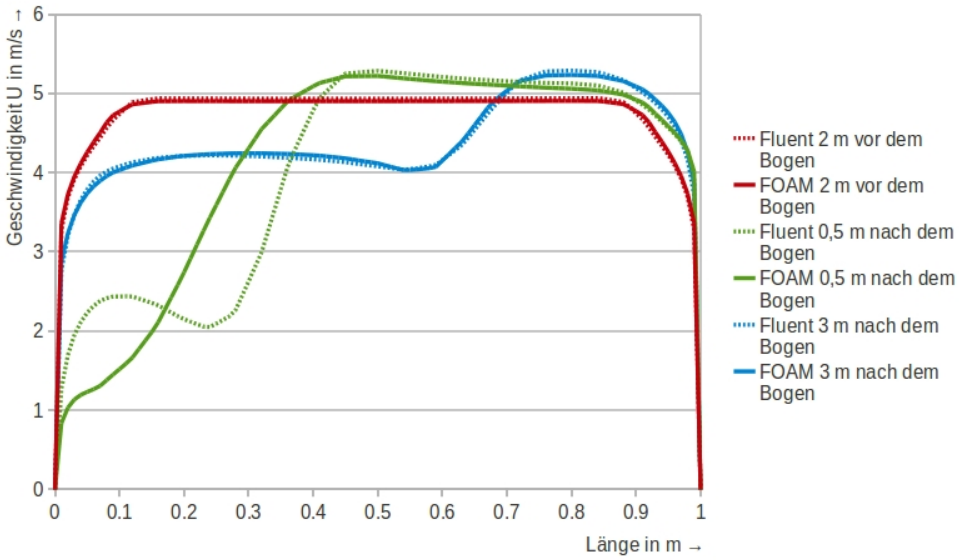


Abb. 4.25: Geschwindigkeitsprofile an verschiedenen Stellen im Kanalbogen (Berechnet von der Kanalinnen- zur Kanalaußenseite)

Betrachtet man die Strömungsvektoren in diesem Gebiet, so zeigt sich, dass die Strömung ablöst und im Bereich der Wand eine Rückströmung erfolgt (Zu erkennen an den gegen die Strömung nach oben zeigenden Vektorpfeilen nahe der Wand in Abbildung 4.26).

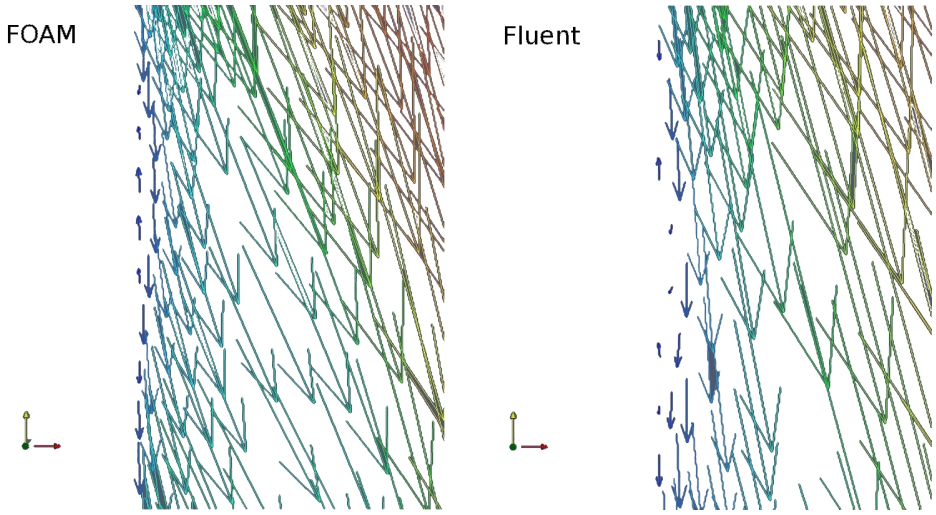


Abb. 4.26: Darstellung des Ablösegebietes als Vektorplot

Vergleicht man dieses Ablösegebiet mit der Darstellung der Strömung in Abbildung 4.2, so fällt auf, dass das Ablösegebiet auf der Kanalbogeninnenseite erst nach dem Kanalbogen beginnt und deutlich kleiner ausfällt.
Weiterhin ist zu erkennen, dass weder Fluent noch OpenFoam das nach Abbildung 4.2 und nach dem in Abbildung 4.24 gezeigten Vergleich nach [16] vorhandene Ablösegebiet an der Kanalbogenaußenseite berechnen. Hier zeigt sich die Grenze der auf dem Wirbelviskositätsansatz beruhenden Turbulenzmodelle.
Dazu nach [21]: *„Zweigleichungsmodelle sind auch dann nicht mehr geeignet, wenn die Krümmung der Stromlinien eine Rolle spielt. Die Krümmung kann die Turbulenz entweder verstärken oder abschwächen(...). Dies wird im Turbulenzmodell über den Produktionsterm berücksichtigt, der dann entweder positiv oder negativ sein muss. Im K-ε-Modell überwiegen meist die positiven Anteile, so dass der Produktionstherm selten negativ wird. Die vorhergesagte Wirbelviskosität ist daher im Falle stabilisierender Krümmung, z. B. (...) in einem Wirbel zu groß.“*
Turbulenzmodelle haben daher Probleme Ablösegebiete darzustellen. So lässt sich auch der abweichende Verlauf des Totaldruckverlustes über den Kanalbogen (vgl. Abb. 4.22) erklären. Durch das in der Simulation nicht vorhandene Ablösegebiet am Kanalbogenanfang verhält sich der Bogen an dieser Stelle weiterhin wie ein gerades Rohr und die Totaldruckverluste nehmen nicht zu (weiterhin konstanter Verlauf über die Länge). Zum Ende des Kanalbogens setzt dann eine Strömungsablösung ein und der Totaldruckverlust nimmt zu. Der Hauptanteil des Druckverlustes entsteht durch die sich ausbildende Sekundärströmung erster Art ab etwa der Kanalbogenmitte.
Ein Vergleich der berechneten Strömungsfelder 3 m nach dem Kanalbogenaustritt zeigt, dass Fluent und FOAM die Sekundärströmung erster Art nahezu identisch berechnen und diese auch

in etwa dem Strömungsfeld nach [28] entsprechen (Abbildung 4.27).

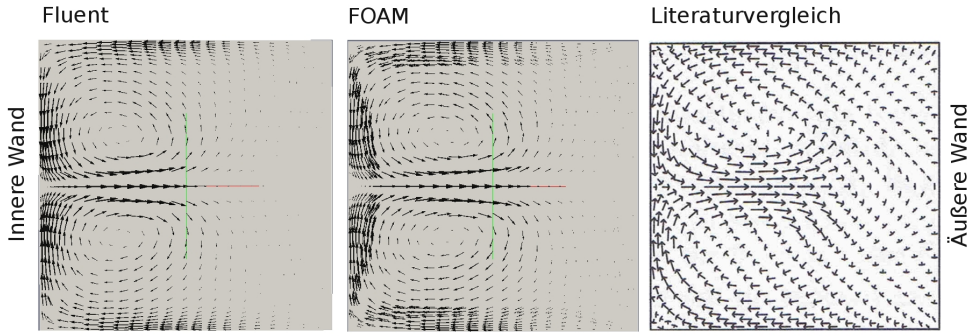


Abb. 4.27: Berechnetes Strömungsfeld (Sekundärströmungen) 3 m nach dem Bogenaustritt (Links: Fluent, Mitte: OpenFoam, Rechts: Abbildung nach [28])

Nach [25] bildet sich weiterhin eine Sekundärströmung zweiter Art (vgl. Abb. 4.2) in eckigen Kanälen aus. Erst bei einer 10-fachen Skalierung des Strömungsfeldes aus OpenFoam und Fluent werden Strömungen in der x,z-Ebene 4 m vor dem Kanalbogen sichtbar (Abb. 4.28). Auf

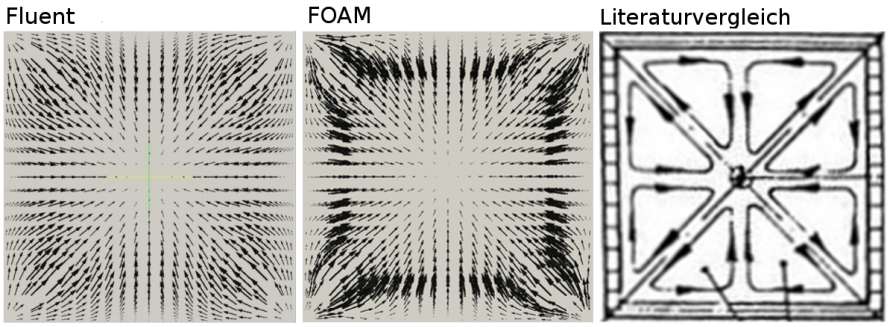


Abb. 4.28: Berechnetes Strömungsfeld (Sekundärströmungen) 4 m vor dem Bogeneintritt (Links: Fluent (10-fach skaliert), Mitte: OpenFoam (10-fach skaliert), Rechts: Abbildung nach [25])

den ersten Blick scheint die berechnete Strömung dabei ähnlich der in der Literatur gezeigten Sekundärströmung zweiter Art. Die Strömung vom Rand hin zur Mitte des Kanals entsteht allerdings aufgrund der Ausbildung des turbulenten Strömungsprofils nach der Vorgabe eines Kolbenprofils am Einströmrand. Die in der Literatur gezeigten Sekundärströmungen zweiter Art werden weder von Fluent noch von OpenFoam berechnet. Grund dafür ist auch hier das auf der Wirbelviskosität basierende Turbulenzmodell und die damit verbundene Annahme isotroper Turbulenz (die turbulente kinetische Energie berücksichtigt nicht die Richtungsabhängigkeit der Turbulenz). Nach [21] können Sekundärströmungen, in nicht kreisförmigen Rohren, eine direkte Folge der Anisotropie der Turbulenz sein. Die Entstehung kann demnach mit einer Vergrößerung der Zähigkeit, wie beim Wirbelviskositätsansatz, alleine nicht erklärt werden. Sie wird durch die richtungsabhängigen Reynolds-Spannungen hervorgerufen. Damit ist die Turbulenz anisotrop und der Effekt der Sekundärströmung zweiter Art lässt sich durch ein isotropes Turbulenzmodell nicht abbilden.

4.5.2 Freie Konvektion und Wärmeübergang an einer ebenen Wand

Ein Vergleich der Luftgeschwindigkeit parallel zur Wand (Abbildung 4.29) berechnet auf unterschiedlich feinen Netzen zeigt, dass OpenFOAM mit der richtig gewählten Netzfeinheit plausible, den Messungen in [3] (in den folgenden Vergleichen als HMT46 bezeichnet) entsprechende Ergebnisse liefert.

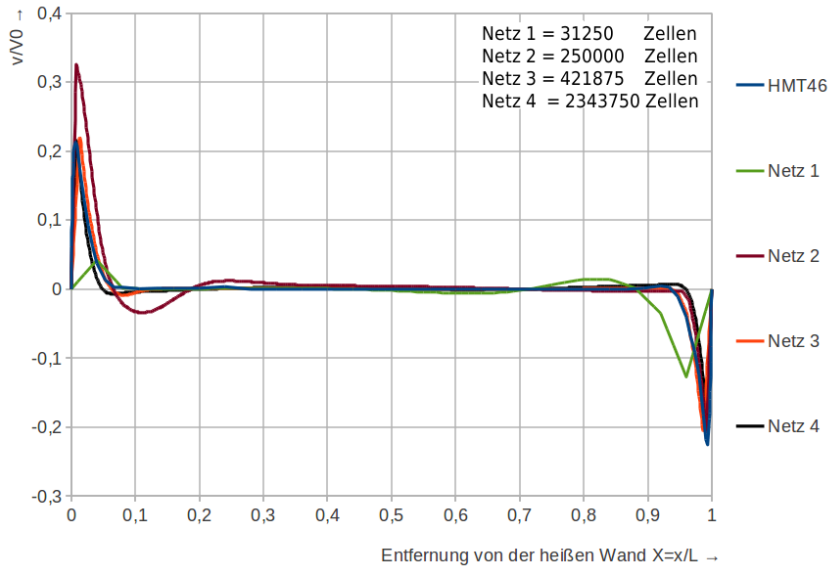


Abb. 4.29: Dimensionslose Geschwindigkeiten parallel zur heißen und kalten Wand in der Mitte des Raumes.

Für die Berechnung auf feineren Netzen können die Berechnungswerte der gröberen Netze als Startwert genutzt werden. Dafür kann das Tool „mapFields“ verwendet werden. Der Befehl „mapFields source -sourceTime latestTime -consistent > log.mapFields&“ interpoliert die Ergebnisse zwischen den Netzen. Im gezeigten Fall wird die Interpolation konsistent (engl: consistent) durchgeführt, d.h. das Berechnungsgebiet und die Randbedingungen der verschiedenen Berechnungsnetze stimmen überein. Für Fälle in denen die Konsistenz nicht gegeben ist wird auf [23] verwiesen.
Netz 1 ist offensichtlich zu grob aufgelöst. Die Zellen im Wandbereich reichen nicht aus, um das Geschwindigkeitsprofil genau genug aufzulösen. Dadurch bedingt sind die Stützstellen im Wandbereich durch Ecken im Profil zu erkennen, die Geschwindigkeit an der Wand steigt zu langsam an und die berechneten maximalen Geschwindigkeiten sind zu gering. Weiterhin sollte das Maximum der Geschwindigkeit parallel zur Wand an der warmen und kalten Wand in etwa den gleichen Wert ergeben, dieses ist beim Netz 1 nicht der Fall. Das zweite Netz 2 stellt den Anstieg der Geschwindigkeiten besser dar. Das Maximum der Geschwindigkeit schwankt allerdings um die experimentellen Werte und wie beim ersten Netz sind die Geschwindigkeiten an der warmen und kalten Wand abweichend. Im gezeigten Zeitschritt ist die geringe Rückströmung an der warmen Wand zu stark ausgeprägt und an der kalten Wand ist eine gute Übereinstimmung mit den experimentellen Daten gegeben. Netz 3 und 4 liefern gute Übereinstimmun-

gen mit den experimentellen Daten. Bei weiteren Untersuchungen zeigte sich, dass auf dem ersten und zweiten Netz die zweidimensionale Strömung in der Mittelebene nicht stabil ist und Strömungen in die Z-Richtung vorhanden sind (Abbildung 4.30). Erst beim Netz 4 sind keine

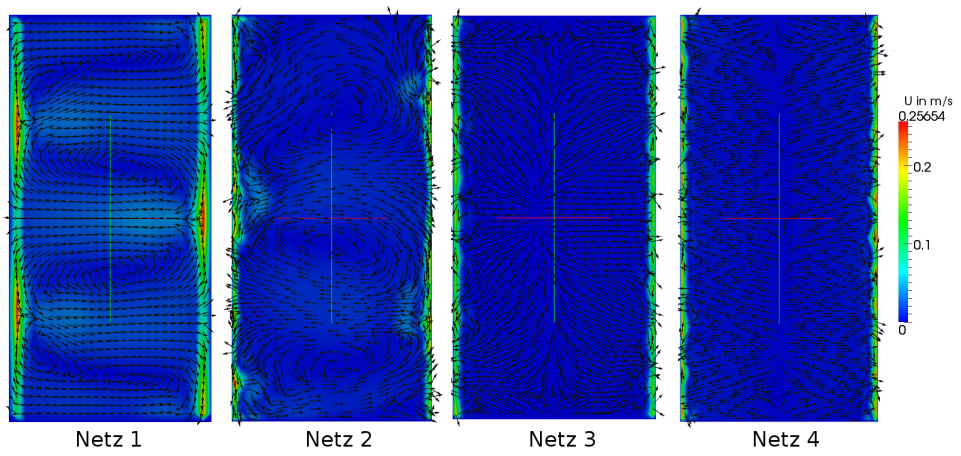


Abb. 4.30: Horizontale Schnittebene bei Y=0,5

Wirbel mit einer Ausdehnung in z-Richtung mehr zu erkennen und es kann von einer zweidimensionalen Strömung ausgegangen werden.
Für den Vergleich der Temperaturen wird die dimensionslose Temperatur

$$DT = \frac{\bar{T} - T_c}{T_h - T_c} \tag{4.25}$$

berechnet ($\bar{T}$ ist die mittlere Temperatur an der Stelle x, T_c die Temperatur der kalten Wand und T_h die Temperatur der warmen Wand) und auf einer Linie von der warmen zur kalten Wand in der Mitte des Raumes verglichen (Abbildung 4.31).

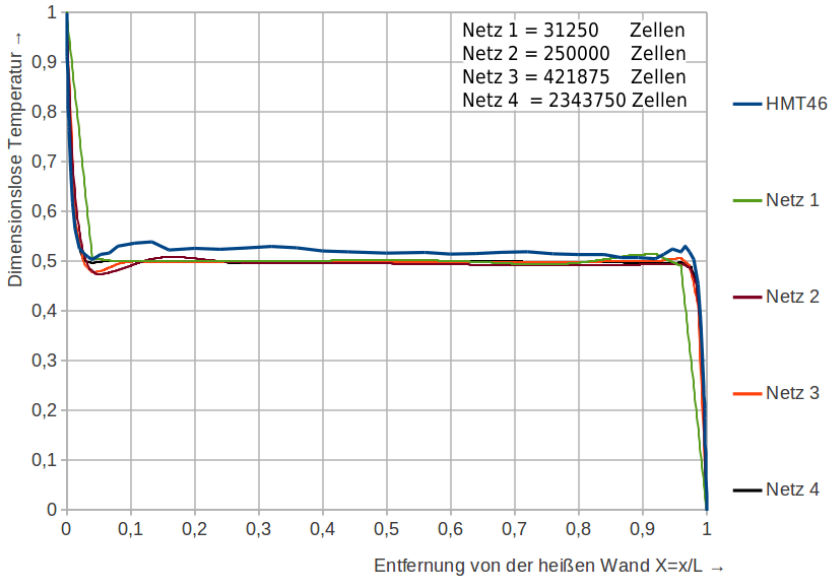


Abb. 4.31: Temperaturverteilung in der Mitte des Raumes bei Y=0,5

Im Vergleich zum Literaturwert fällt auf, dass die simulierte Temperatur auf allen Netzen in der Mitte des Raumes geringer ausfällt. Sie entspricht exakt der mittleren Temperatur von 30°C und ist auf die idealen Randbedingungen zurückzuführen. Der Anstieg der Temperatur von der Wand ins Fluid hinein fällt beim Netz 1 ähnlich wie bei der Geschwindigkeit zu gering aus. Netz 2-3 liefern vergleichbare Ergebnisse, der Unterschied zum ersten Netz ist allerdings gering. Bei der Temperaturverteilung über die Raumhöhe ist die Netzfeinheit entscheidender (Abbildung 4.32). Erst Netz 4 stellt die maximalen Temperaturen an der Decke und am Boden genau genug dar. Als weiterer Punkt wird die lokale Nusselt-Zahl sowie die mittlere Nusselt-Zahl entlang

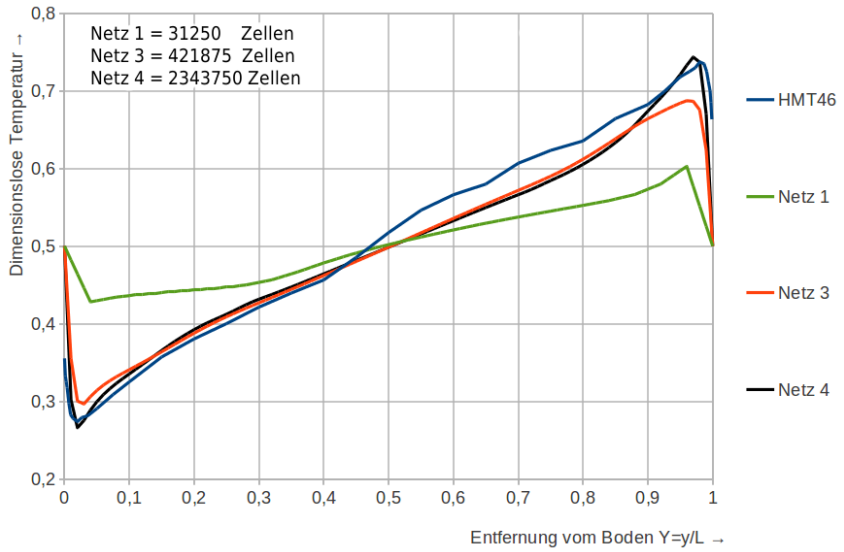


Abb. 4.32: Temperaturverteilung über die Höhe des Raumes bei X=0,5

der heißen Wand untersucht. Die Nusselt-Zahl kann durch

$$Nu = \frac{\alpha \cdot L}{\lambda} \tag{4.26}$$

bestimmt werden. L ist die Tiefe des Raumes. Der Wärmeübergangskoeffizient α lässt sich folgendermaßen bestimmen:

$$\alpha = \frac{\dot{Q}}{A \cdot (\vartheta_h - \vartheta_c)}. \tag{4.27}$$

Dabei ist A die Wärmeübertragungsfläche und $\dot{Q}$ der Wärmestrom. Setzt man Gleichung (4.27) in Gleichung (4.26) ein, ergibt sich:

$$Nu = \frac{\dot{Q}}{H \cdot (\vartheta_h - \vartheta_c) \cdot \lambda}. \tag{4.28}$$

Zur Bestimmung des Wärmestroms $\dot{Q}$ wird das OpenFoam-Werkzeug „WallHeatFlux“ in einer angepassten Form genutzt. Die Anpassung ist erforderlich, da das verwendete allgemeine physikalische Berechnungsmodell „hRhoThermo“ im Werkzeug nicht vorgesehen ist. Weiterhin ist eine Anpassung des Solvers *BuoyantPimpleFoam* nötig, da „WallHeatFlux“ die Dich-

te benötigt, diese aber nicht standardmäßig vom Solver ausgegeben wird. Für die Anpassung werden das Werkzeug und der Solver in das Verzeichnis „Benutzer/OpenFoam/Benutzer-2.0.1/applications/...“ kopiert, umbenannt, angepasst und neu kompiliert (Anhang B.2). Nach der Berechnung mit dem angepassten Solver kann das Werkzeug „WallHeatFlux“ auf alle Zeitschritte oder auf einen gewünschten Zeitschritt angewendet werden (OpenFoamTerminal: wallHeatFluxRho -compressible -time ...).
Anschließend können die lokalen Wärmeströme (Wärmestrom pro Wandzelle) in Paraview geladen werden. Mittels Gleichung (4.28) lässt sich die lokale Nusseltzahl direkt in ParaFOAM berechnen. Abbildung 4.33 zeigt die lokale Nusselt-Zahl in der Mitte des Raumes entlang der heißen Wand vom Boden bis zur Decke. In den experimentellen Ergebnissen nach [3] hat die

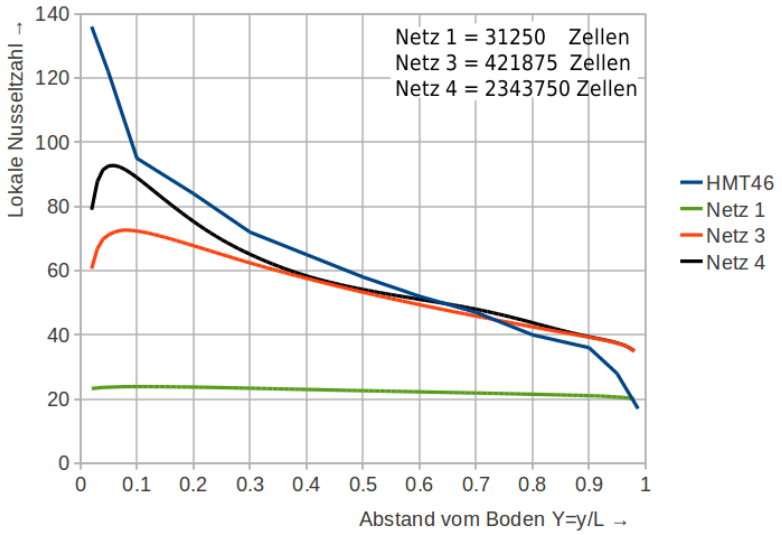


Abb. 4.33: Lokale Nusselt-Zahl entlang der heißen Wand

lokale Nusselt-Zahl ihr Maximum am Boden der heißen Wand - zurückzuführen auf die dünne thermische Grenzschicht - und nimmt dann mit der Lauflänge ab. Am oberen Ende der warmen Wand trifft die Fluidströmung auf die Decke und der Wärmeaustausch nimmt rapide ab. Bei dem gezeigten Verlauf existiert allerdings ein Unterschied zu der Veröffentlichung der identischen Studie in Heat and Mass Transfer 43 aus dem Jahr 2000 [2]. In dieser Veröffentlichung stellt der gezeigte Verlauf den adiabatischen und nicht den in der Studie untersuchten Fall mit perfekt wärmeleitender Decke und Boden dar. Dieser Verlauf entspricht nach [2] den in OpenFoam simulierten Verläufen. Es kann daher davon ausgegangen werden, dass bei der Neuveröffentlichung der Studie mit einer detallierten Angabe der Messwerte, die in dieser Arbeit genutzt werden, ein Fehler unterlaufen ist. Da sich die Verläufe in [2] und [3] aber nur im Bereich nahe der unteren Ecke unterscheiden, können die Daten dennoch für eine Validierung genutzt werden.
Beim Vergleich der simulierten Verläufe mit dem experimentellen Verlauf fällt auf, dass die simulierte lokale Nusselt-Zahl beim ersten Netz zu gering ausfällt. Dieses ist auf die nicht ge-

nügend aufgelöste thermische Grenzschicht zurückzuführen, wodurch nicht genug Wärme in den Raum transferiert werden kann. So lassen sich die zu geringen Temperaturen in Abbildung 4.31 und 4.32 erklären. Netz 3 und 4 liefern gute Ergebnisse im Bereich von 0,1 bis 1 Y, wobei Netz 4 den Verlauf etwas besser trifft. Am unteren Ende ist die lokale Nusselt-Zahl im Vergleich zu den experimentellen Ergebnissen zu niedrig. Zurückzuführen ist dieses auf eine zu geringe Netzauflösung im Bereich der Ecken des Raumes. Dadurch bedingt bildet sich am unteren Ende der Wand eine tote Ecke. Durch die hohe Temperatur des Fluids in diesem Bereich kann nur eine geringe Wärmemenge an das Fluid abgegeben werden. Da die Geschwindigkeiten in der unteren Ecke geringer sind, ist der Einfluss des groben Netzes an dieser Stelle größer. Der Unterschied zwischen Netz 3 und 4 zeigt, dass eine Netzverfeinerung insbesondere im Bereich der unteren Ecken bessere Ergebnisse liefert.
Eine Integration der lokalen Wärmeströme liefert den insgesamt durch die heiße Wand abgegebenen Wärmestrom, über den mittels Gleichung (4.28) die mittlere Nusselt-Zahl berechnet wird (Abbildung 4.34). Die Simulation auf Netz 4 ergibt trotz der immer noch zu groben Auflösung

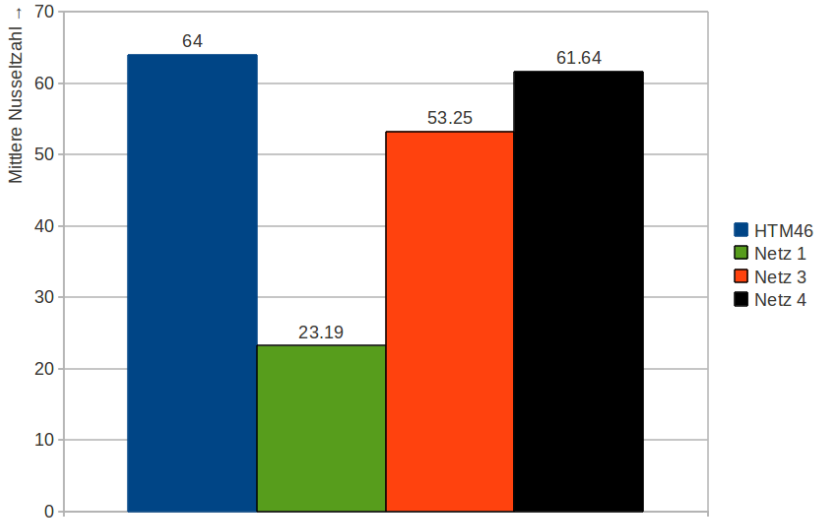


Abb. 4.34: Mittlere Nusselt-Zahl der heißen Wand

der Ecken eine Abweichung von 3,7 % von der in [3] angegebenen mittleren Nusselt-Zahl. Auf eine weitere Verfeinerung des Netzes wird daher an dieser Stelle verzichtet.

4.5.3 Thermisch induzierter Auftriebsstrahl und Rauchschichtung

Die Auswertung der in Kapitel 4.2 beschriebenen Untersuchungsmerkmale setzt eine ausgebildete Temperaturschichtung über die Raumhöhe voraus. Damit sich diese Schichtung ausbilden kann ist eine lange Berechnungszeit erforderlich. Zur Kontrolle der ausgebildeten Schicht werden die Änderungen im Strömungsgebiet zwischen zwei Zeitschritten betrachtet. Ändern sich diese kaum noch, kann von einer auskonvergierten Lösung ausgegangen werden. Für den betrachtete Raum wird dieser Zustand nach 11 simulierten Minuten erreicht. Die Berechnung auf

4 Prozessoren, bei der gegebenen Netzfeinheit (Vgl. Tabelle 4.4), Diskretisierungsverfahren zweiter Ordnung und einer Zeitschrittweite von 0,05 s, beträgt zwei Wochen. Die Zeitschrittweite ergibt sich aus der Forderung für die CFL-Zahl (vgl. Gl. (4.21)) ≤ 1. Nützlich ist dabei, dass OpenFOAM bei jedem Zeitschritt die CFL-Zahl ausgibt und die Zeitschrittweite entsprechend angepasst werden kann.
Die Ermittlung der Schichtdicke zeigt Abbildung 4.35. Die Temperaturprofile über die Raum-

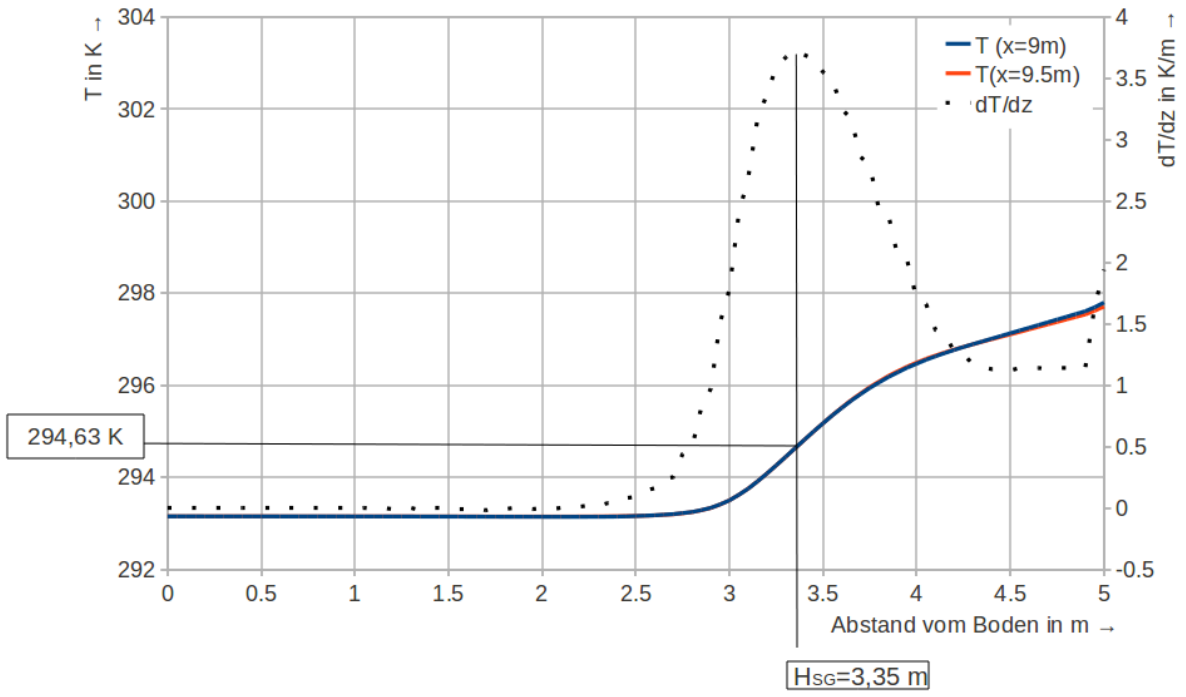


Abb. 4.35: Bestimmung der Schichtgrenze und der Grenztemperatur

höhe in einem Abstand von 1 und 0,5 m von der Wand werden gemittelt und die Änderung über die Raumhöhe bestimmt. An der Stelle der größten Temperaturänderung wird nach [30] die Schichtgrenze festgelegt. An dieser Stelle wird die Höhe der Schichtgrenze h_{SG} und die Schichtgrenztemperatur bestimmt. Diese sind im Diagramm mit eingezeichnet. Die Grenzschichthöhe der OpenFOAM-Berechnung beträgt $h_{SG} = 3,35\ m$. In [30] sind die Ergebnisse in der Form $h_{SG} = f(\dot{m}_{AB})$ dargestellt. Der Massenstrom der Absaugung $\dot{m}_{AB}$ kann mittles des OpenFOAM-Tools „patchAverage“ bestimmt werden. Mit diesem Tool wird die mittlere Geschwindigkeit der Absaugung $u = 0,0479$ m/s und die mittlere Dichte $\rho = 1,1902$ m/s bestimmt. Mit der Fläche der Absaugung $A = 35,81\ m^2$, die aus den in Kapitel 4.3.3 genannten Gründen kleiner ist als die in [30] angegebene Fläche, ergibt sich der Massenstrom zu:

$$\dot{m} = \rho UA = 2,04\ \frac{kg}{s}. \tag{4.29}$$

Für diesen Massenstrom kann [30] eine Schichtgrenzhöhe $h_{SG} = 3,4\ m$ entnommen werden, was einer Abweichung von 1,47 % zu den OpenFOAM-Ergebnissen (vgl. Abb. 4.35) entspricht.

Allerdings zeigt sich, dass sich entgegen der Annahmen in Kapitel 4.1.3 in der Rauchschicht keine isotherme Temperatur ausgebildet hat. Gegebenenfalls könnte eine längere Berechnungsdauer zu einer Annährung an die isotherme Temperatur beitragen. Aufgrund der zeitintensiven Berechnung und der geringen prozentualen Abweichung der Schichtgrenze wird an dieser Stelle auf eine weitere Berechnung verzichtet.
Die Geschwindigkeiten im Deckenstrahl zeigt Abbildung 4.36. VDI4 im Diagramm bezeichnet die Ergebnisse nach [30]. Es zeigt sich, dass die Verläufe in OpenFOAM qualitativ richtig berechnet werden. Allerdings sind die maximalen Geschwindigkeiten an den Stellen $r = 2\ m$ und $r = 4\ m$ zu gering und die Geschwindigkeiten fallen im weiteren Verlauf zu schnell ab. Eine

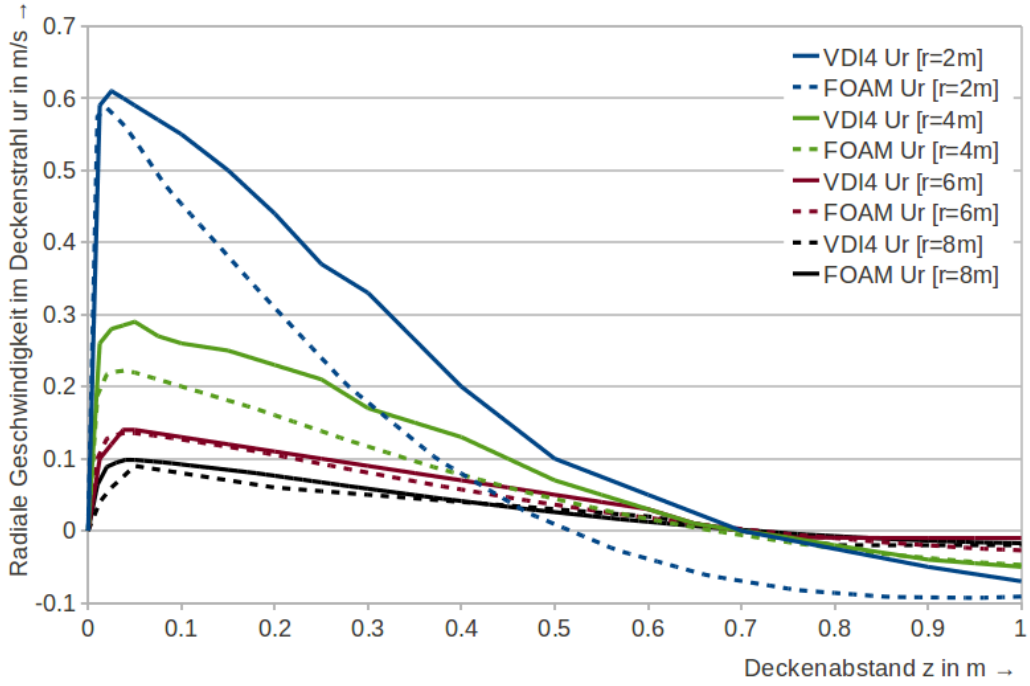


Abb. 4.36: Geschwindigkeitsentwicklung im Deckenstrahl. VDI4 stellen die Vergleichswerte nach [30] dar.

Analyse des in das System eingebrachten Wärmestroms $Q = \dot{m} \cdot cp \cdot \Delta T = \rho \cdot c \cdot A \cdot cp \cdot \Delta T$ über der Wärmequelle zeigt, dass dieser 500 Watt zu gering ausfällt. Hier kann die Abgrenzung der Wärmequelle (vgl. Kapitel 4.4.3) als Grund genannt werden. Ähnlich zu den Problemen der Abgrenzung der Randbedingungen in SnappyHexMesh (vgl. Abbildung 4.17) werden bei der Vorgabe über das Tool „SetFields“ nur ganze Zellen mit dem Wärmestrom belegt. Zellen, die aus dem definierten Zylinder der Wärmequelle hinausragen, erhalten damit keine Zuweisung. Dadurch bedingt ist das Volumen der Wärmequelle zu gering und somit ebenfalls der in das System eingebrachte Wärmestrom. Durch den zu geringen Wärmestrom sind die Geschwindigkeiten im Auftriebsstrahl ebenfalls geringer und auch nach der Umlenkung ist der Effekt zu beobachten. Dieses äußert sich in einer zu geringen Induzierung der Umgebungsluft und damit verbunden zu geringen Geschwindigkeiten im Verlauf des Deckenstrahls. Dieses gilt für die

Positionen $r = 2\ m$ und $r = 4\ m$. An den Positionen $r = 6\ m$ und $r = 8\ m$ beeinflusst die konstante Absaugung das Geschwindigkeitsprofil maßgeblich und die Geschwindigkeiten entsprechen den Werten aus [30]. Abbildung 4.37 zeigt abschließend die ermitteltet Grenzschichttemperatur als Iso-Fläche.

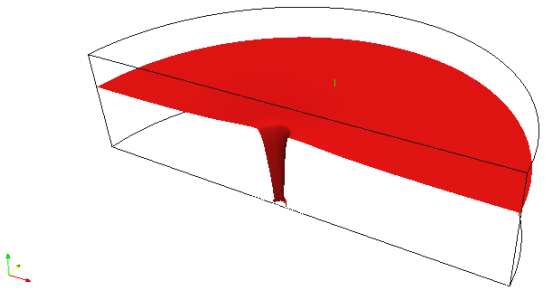


Abb. 4.37: Ermittelte Schichtgrenze. Darstellung als Isofläche anhand der Grenzschichttemperatur von 294,63 K.

4.6 Gesamtbewertung der grundlegenden Testfälle

Die Simulation der drei grundlegenden Testfälle zeigt, dass OpenFOAM für die Simulation grundlegender Strömungsformen aus dem Anwendungsgebiet der Gebäudetechnik grundsätzlich geeignet ist und mit Fluent vergleichbare Ergebnisse liefern kann. Allerdings ist der Aufwand der Modellierung, Vernetzung und das Aufsetzen des Strömungsmodells im Vergleich zu Fluent erheblich größer. Dieses beginnt mit der Erstellung der Geometrie in Slomé. Schon die Erstellung der einfachen Geometrien der Testfälle kostete, bedingt durch den Export einzelner STL-Flächen und die wenig intuitive Arbeitsweise, viel Zeit. Hier bieten andere Programme mit einer featurebasierten Vorgehensweise einen höheren Komfort.
Die automatische Vernetzung in SnappyHexMesh funktioniert gut. Lediglich die Erstellung von Boundary Layern bereitete Schwierigkeiten, allerdings gilt dieses auch für viele andere Vernetzungstools. Als weiterer Kritikpunkt kann die Erzeugung der Randflächen genannt werden. Hier zeigt sich, dass die Größe der Randflächen von dem vorher definierten Grundnetz abhängt (vgl. Abb. 4.17). Die Vernetzung ist nicht Patch-Konform, so können die Randflächen zu klein bzw. groß ausfallen.
Mittels des Tools „checkMesh“ lässt sich die Netzqualität unkompliziert überprüfen. Gibt das Tool keine Fehlermeldungen aus, kann davon ausgegangen werden, dass die Netzqualität für eine Berechnung ausreichend hoch ist. Allerdings weist das Tool nicht auf die Patch-Unkonforme Vernetzung hin.
Die Definition der Solvereinstellungen ist anhand der mitgelieferten Tutorials für einfache Testfälle schnell durchführbar (z.B. für den Kanalbogen). Bei komplexeren Simulationen fallen allerdings immer mehr einzelne Definitionsdateien an und es fällt schwer, den Überblick zu behalten. Auch sind die Tools, wie z.B. „WallHeatFluxRho“, teilweise nicht auf die mitgelieferten

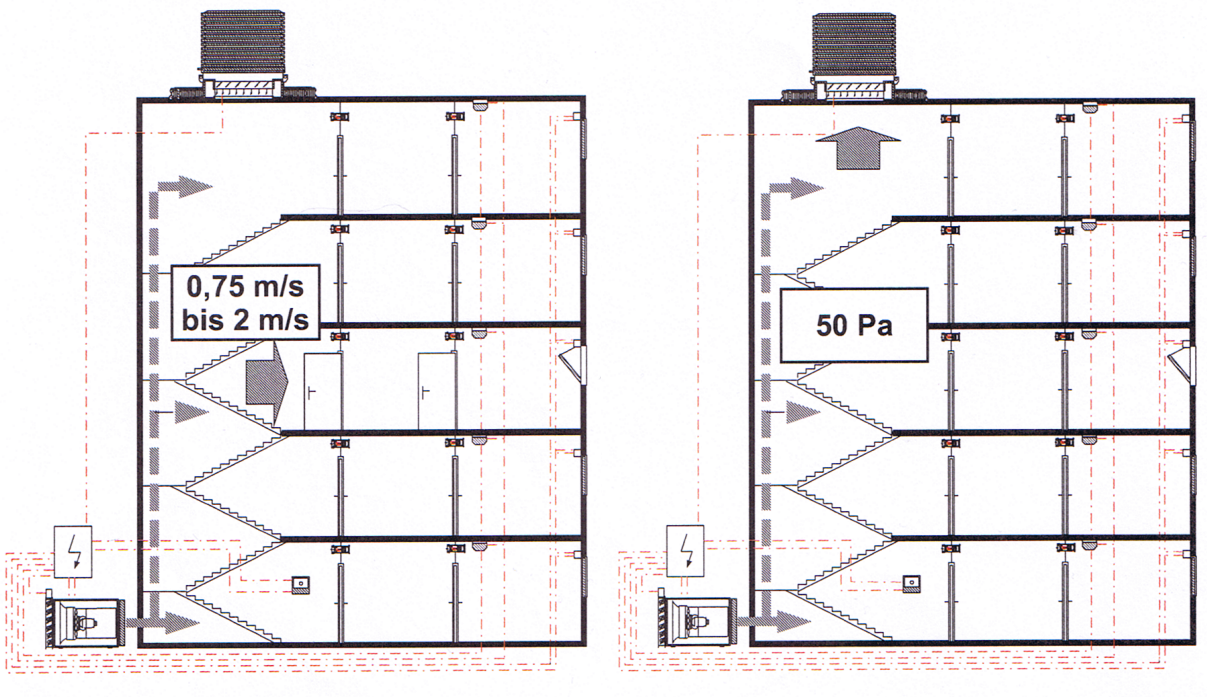
Solver anwendbar. Die Fehlermeldungen sind dabei teilweise schwer zu deuten, wodurch die Anpassung der Tools und Solver erschwert wird.
Im zweiten Testfall konnten mittels des Tools „mapFields“ erfolgreich Ergebnisse von groben Berechnungsnetzen auf feinere Berechnungsnetze interpoliert werden. Hierdurch ist die Möglichkeit gegeben Werte auf groben Netzen mit großen Zeitschritten zu berechnen und diese als Startwerte für eine Berechnung auf feineren Netzen zu nutzen, wodurch sich die Berechnungszeit erheblich verkürzen lässt.
Die Möglichkeiten der Anpassung der Tools und Solver an die eigenen Bedürfnisse haben der zweite und dritte Testfall gezeigt. So konnte für den zweiten Testfall das Tool „wallHeatFlux“ und der Solver *buoyantPimpleFoam* aufeinander abgestimmt werden. Im dritten Testfall wurde der Solver *BuoyantPimpleFoam* erfolgreich um die Berücksichtigung einer volumetrischen Wärmequelle erweitert, um somit vergleichende Rechnungen nach [30] durchführen zu können. Darüber hinausgehende Versuche, den Rauch durch eine „skalare Transportgröße“ oder eine „Spezies“ kenntlich zu machen waren, hingegen nicht erfolgreich. Ansätze hierzu liefern die Solver *scalarTransportFoam*, *reactingFoam* und das CFD-Online Forum. Erste Ansätze, die Darstellung der skalaren Transportgröße im Solver *scalarTransportFoam* in den Solver *buoyantPimpleFoam* zu integrieren, sind vielversprechend, liefern allerdings keine plausiblen Ergebnisse (Änderungsansätze für die Solver befinden sich auf der beiliegenden Daten-DVD). Dieses liegt daran, dass der Solver *scalarTransportFoam* für den stationären laminaren Fall geschrieben wurde. Eine Anpassung der Transportgleichung auf den instationären turbulenten Fall ist daher erforderlich, kann aber im Rahmen der vorliegenden Arbeit aus Ermangelung an Zeit nicht geleistet werden.
Die Residuen lassen sich mittels des Tools „foamLog“ auswerten. Dabei wird der Verlauf der Residuen aus der .log Datei des Solvers ausgewertet. Sollen die Residuen während der Berechnung ausgegeben werden, so muss eine entsprechende Skriptdatei geschrieben werden (vgl. Anhang A.4). Mittels des Open Source Tools Gnuplot lassen sich anschließend über den Befehl „gnuplot residuals -lam“ die Residuen ausgeben.
Hierbei zeigte sich allerdings als weiterer limitierender Faktor die Zeitschrittweite. Bei einer großen Zeitschrittweite divergierte die Lösung in allen Fällen. Die Einhaltung der CFL-Zahl nahe Eins ist daher zwingend erforderlich. Durch die dadurch bedingte teilweise sehr kleine Zeitschrittweite (0,001 s) und die große Anzahl an Zeitschritten steigt die Berechnungszeit an.
Als größter Kritikpunkt ist allerdings zu allen verwendeten Tools die knappe oder fehlende Dokumentation zu nennen, die eine Einarbeitung erschwert. Einschlägige CFD-Foren können bei der Arbeit mit OpenFOAM hilfreich sein, allerdings muss dabei beachtet werden, dass sich die Skript-Eingaben der Randbedingungen, Solvervorgaben usw. mit den verschiedenen OpenFOAM Versionen teilweise geändert haben und nicht auf neuere Versionen anwendbar sind.

5 Abschließender Testfall aus der aktuellen Forschung

Das vorherige Kapitel hat gezeigt, dass OpenFOAM für die Simulation grundlegender Strömungsformen aus dem Anwendungsgebiet der Gebäudetechnik grundsätzlich geeignet ist. In diesem Kapitel soll der Frage nachgegangen werden, ob OpenFOAM auch auf komplexere Fragestellungen anwendbar ist. Dafür soll die Treppenhausdruckbelüftungsanlagen eines Sicherheitstreppenhauses simuliert werden. In dieser Arbeit wird beispielhaft das Sicherheitstreppenhaus des Towers 185 (Gesamthöhe 192 m) betrachtet, das aktuell Thema in der Forschung bei Imtech ist. Untersucht werden der Druckverlust bei aufwärtsgerichteter Strömung und der Einfluss des thermischen Auftriebs. Zunächst werden die Grundlagen zusammengefasst.

5.1 Grundlagen

Im Baurecht gilt grundsätzlich das Prinzip, dass vertikal zwei Rettungswege zur Verfügung stehen müssen. Ist dieses, vor allem bei Hochhäusern, nicht möglich, muss der einzige vertikale Rettungsweg als Sicherheitstreppenhaus ausgeführt werden. Da Rauch den Treppenraum unpassierbar machen kann, muss das Eindringen von Rauch verhindert werden. Dafür wird mit Rauchschutzdruckanlagen im Treppenraum ein Überdruck zu den angrenzenden Geschossen aufgebaut. Abbildung 5.1 zeigt die zwei Kriterien, die ein Sicherheitstreppenhaus erfüllen muss. Sind im vermeintlichen Brandgeschoss beide Schleusentüren geöffnet (Abb. 5.1 a), so muss ei-



(a) (b)

Abb. 5.1: Anforderungen und Bemessungen eines Sicherheitstreppenhauses: a) Kriterium Geschwindigkeit b) Kriterium Druck [10]

ne Durchströmung der Türen mit einer Mindestgeschwindigkeit von 0,75 bis 2 m/s erfolgen. In dieser Phase schließt die Druckregelklappe im Kopf des Treppenraumes, damit der komplette Volumenstrom für die Durchströmung der offenen Tür bereitsteht. Im jeweiligen Brandgeschoss sind die Abströmklappen geöffnet, damit die Durchströmung sichergestellt werden kann.
Sind hingegen alle Türen im Treppenraum geschlossen (Abb. 5.1 b), so soll ein kontrollierter Überdruck von 50 Pa im Treppenraum aufgebaut werden. Die Türöffnungskraft am Türgriff darf dabei an keiner Stelle > 100 N betragen. In dieser Phase ist die Druckregelklappe im Kopf des Treppenraumes geöffnet und lässt die überschüssige Luftmenge entweichen. Dabei produziert sie gerade einen Druckverlust in der Größenordnung des geplanten Überdrucks.
Bei innenliegenden Treppenhäusern werden zusätzlich höhenabhängige physikalische Einflüsse zunehmend wichtiger. Hier sind Windlasten, der thermische Auftrieb und Treppenraumdruckverluste zu nennen.
Bei der Betrachtung der Thermik muss zwischen dem oben geschlossenen und offenen Treppenhaus unterschieden werden. Ist der Treppenraum verschlossen, hat die Luft im Treppenraum die Temperatur, die auch im übrigen Gebäude herrscht. Abbildung 5.2 zeigt qualitativ diesen Fall am Beispiel des verschlossenen Kamins im Winterfall. Durch die höhere Temperatur und

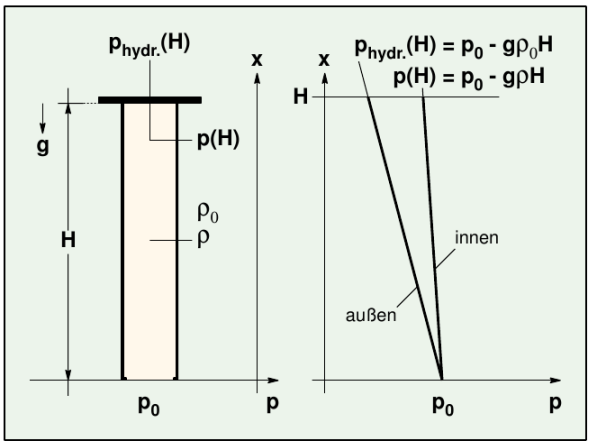


Abb. 5.2: Hydrostatische Druckverteilung inner - und außerhalb des versperrten Kamins [1].

der entsprechenden geringeren Dichte im Treppenhaus, sinkt der Druck über die Höhe im Treppenhaus weniger stark ab als in der Umgebung. Am Treppenhauskopf ($x = H$) herrscht damit eine Druckdifferenz von

$$\Delta p_{hyd}(H) = (\rho_{Aussen} - \rho_{Innen})gH. \tag{5.1}$$

Je nach Temperatur- und Höhenunterschied kann sich hieraus ein die geforderten 50 Pa übersteigender Druck an den Türen im geschlossenen Treppenhaus einstellen.
Wird das Treppenhaus geöffnet, wird die im Treppenraum eingeschlossene Luft beschleunigt und strömt nach oben aus dem Treppenhaus aus. Die sich bei der konvektiven Durchströmung des Treppenhauses einstellende Druckdifferenz zur Umgebung hängt dabei maßgeblich von den Widerständen im Treppenhaus sowie den Widerständen beim Eintritt in das Treppenhaus und

beim Austritt aus dem Treppenhaus ab. Analytische Berechnungen hierzu wurden in [1] am Beispiel eines Kamins durchgeführt (Abb. 5.3). Im reibungsfreien Zustand ist der Kamin voll

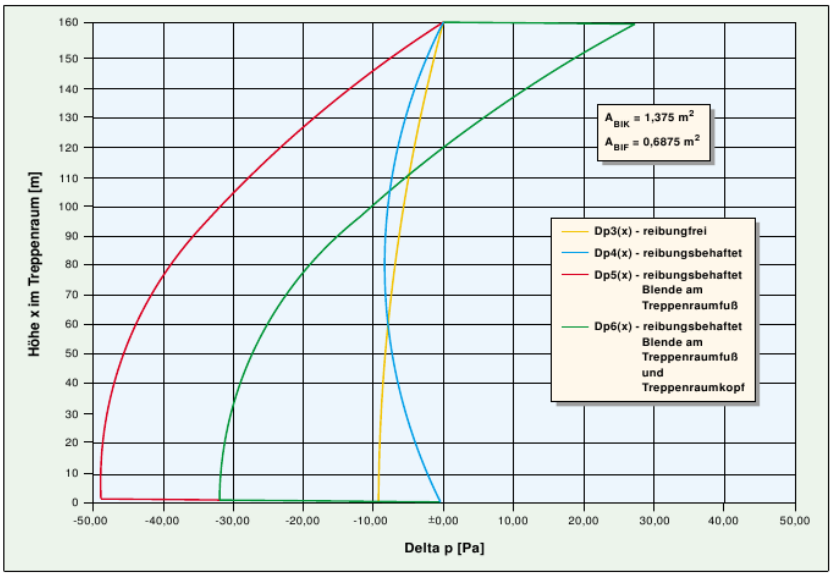


Abb. 5.3: Druckdifferenzen zwischen Treppenraum und Umgebung [1].

geöffnet und die Ein- und Ausströmöffnung gleich groß. Um den Einfluss der Einbauten und Treppenstufen abzubilden wird im zweiten Fall die Wandreibung im Kamin berücksichtigt. Die Fälle drei und vier berücksichtigen zusätzlich die Widerstände beim Ein- und Austritt in das Treppenhaus. A_{BIK} und A_{BIF} stellen die Öffnungen am Kopf und Fuß des Treppenhauses für die Berücksichtigung der Druckverluste am Ein- und Austritt dar.

Der Fall vier, der dem realen Treppenhaus am nächsten kommt, zeigt, dass durch den Auftrieb bis zur Höhe von 120 Metern im Treppenhaus ein Unterdruck herrscht. Hier würde Rauch aus einem Brandgeschoss in das Treppenhaus gesogen. Die Kenntnisse der Druckverluste und der Druckdifferenz zur Umgebung sind wichtig, um die Rauchschutzdruckanlage richtig auszulegen. Bei einem Betrieb der Rauchschutzdruckanlage stellen sich allerdings andere Druckverhältnisse als bei der freien konvektiven Durchströmung des Kamins ein. Abbildung 5.4 zeigt beispielhaft die Druckverhältnisse für den Winterfall. Die Effekte des Auftriebs und der Druckverlust im Treppenraum können genutzt werden, um einen definierten Überdruck aufzubauen. Am Treppenhauskopf wird ein kontrollierter Überdruck von 55 Pa aufgebaut. Im Winterfall ist der Auftriebsdruck am Treppenhauskopf deutlich größer. Für den 192 m hohen Tower 185 beispielsweise ergibt sich bei einer Außentemperatur von -10°C ($\rho_{Aussen} = 1,3245\ kg/m^3$) und einer konstanten Innenraumtemperatur von 20°C ($\rho_{Innen} = 1,1885\ kg/m^3$) mittels Gleichung (5.1) eine Druckdifferenz von 256,16 Pa. Bei dem Abbau des Druckes über die Druckregelklappe wird das Fluid im Treppenhaus beschleunigt und es stellt sich, wie dargestellt, bis zu einer gewissen Höhe ein Unterdruck im Treppenhaus ein. Über die Druckbelüftung sorgt man nun dafür, dass sich im Treppenhaus ein Volumenstrom und damit ein Druckverlust über die Höhe ergibt, der am Treppenhausfuß einen Druck entsprechend dem thermischen Auftrieb und dem

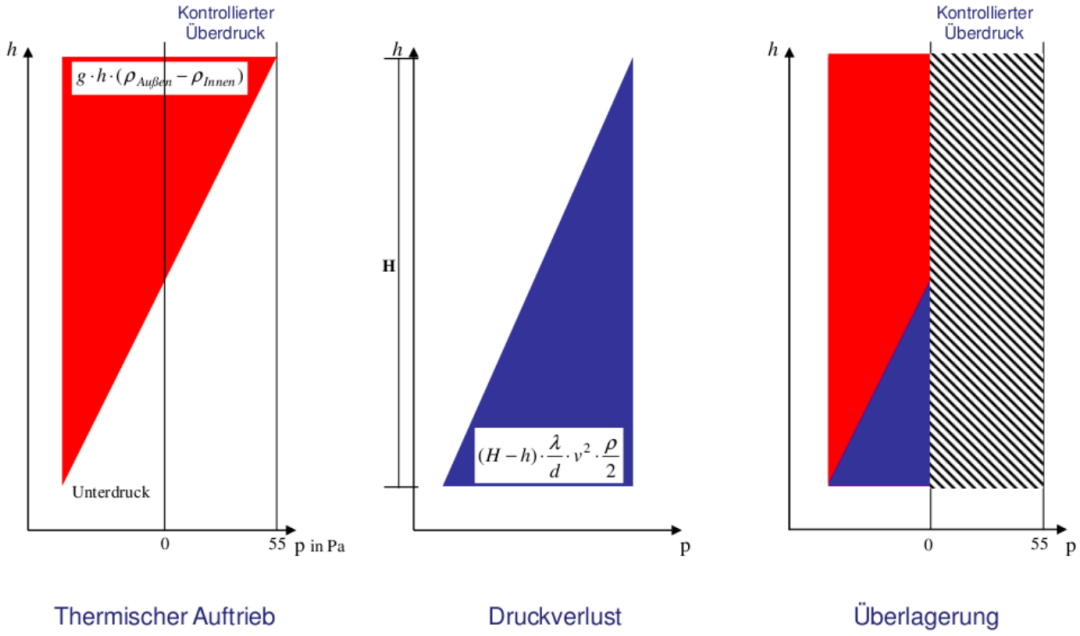


Abb. 5.4: Definierter Überdruck über das gesamte Treppenhaus im Winterfall [Imtech]

Unterdruck erzeugt. Wie in der Überlagerung dargestellt, stellt sich im Treppenhaus in diesem Fall ein über die Höhe konstanter Überdruck von 55 Pa ein.

5.2 Simulation

Für die Simulation des Sicherheitstreppenhauses wird der Fall vollständig geschlossener Türen im Treppenraum ausgewählt (vgl. Abb. 5.1 b). Weiterhin wird eine Simulation bei winterlichen Bedingungen angesetzt. Unter diesen Voraussetzungen gilt für die Randbedingungen:

- Konstanter eintretender Volumenstrom am Fuß des Treppenhauses mit Abströmung über die Druckregelklappe. Die Druckregelklappe erzeugt dabei einen konstanten Druckverlust der dem geplanten Überdruck von 50 Pa entspricht.
- Die Wandtemperaturen entsprechen der Raumtemperatur von 20°C.
- Die Temperatur der einströmenden Luft entspricht der gewählten Außentemperatur von -10°C.
- Der Auftrieb der Luftsäule soll berücksichtigt werden (Anfahrvorgang und stationärer Fall).

Bei Imtech wird diese Art der Simulation in Fluent durchgeführt. Der Druckverlust der Druckregelklappe kann dabei über eine „Porous Jump Boundary Condition“ realisiert werden. Mittels dieser Randbedingung kann eine dünne Membran mit einer vorgegebenen Druckverlust-Charakteristik gesetzt werden [13]. Der thermische Auftrieb im Treppenhaus wird in Fluent durch die Vorgabe einer konstanten Referenzdichte berücksichtigt. Die berechneten Modelle

sind voll kompressibel, d.h. die Boussinesq Approximation wird nicht verwendet. In diesem Fall erscheint die Referenzdichte im Body-Force Term in der Momentengleichung (vgl. Gl. (2.43)) in der Form $(\rho - \rho_0) \cdot g$ [13]. Im Fall des winterlichen Treppenhauses steht im Innenraum eine Luftsäule mit der Temperatur von 20°C. Um den Auftrieb der Luftsäule zu berücksichtigen, kann die Referenzdichte in Fluent auf die Dichte der Außentemperatur, beispielsweise -10°C gesetzt werden. In diesem Fall strebt die Luftsäule nach oben und es stellt sich der entsprechende Auftriebsdruck ein.
Für die Simulation in OpenFOAM soll der Solver *BuoyantPimpleFOAM* verwendet werden, der in den grundlegenden Testfällen der freien Konvektion und des thermischen Auftriebstrahls (vgl. Kapitel 4.5.2 und 4.5.3) erfolgreich validiert werden konnte. Die warmen Wände und die kalt einströmende Luft können abgebildet werden. Für den Druckverlust der Druckregelklappe konnte in OpenFOAM keine passende Randbedingung ermittelt werden. Versuche mit einer „Jump“ Boundary und mit einer „pressureFan“ Boundary führten nicht zum Erfolg. An dieser Stelle mag auch die fehlende Dokumentation dieser Randbedingungen der Grund sein.
Weiterhin ist es nicht gelungen, den thermischen Auftriebsdruck zu Beginn der Simulation darzustellen. Dieses liegt daran, dass für den Solver *BouyantPimpleFOAM* keine Referenzdichte angegeben werden kann, sondern die Referenzdichte im Solver durch die Mittelung über die gesamte Domain bestimmt wird und die winterliche Umgebung somit unberücksichtigt bleibt.
Die genannten Einschränkungen konnten nicht gelöst werden. Um dennoch die Möglichkeiten der in dieser Arbeit bisher untersuchten Solver darzustellen, wird die Simulation des Sicherheitstreppenhauses in zwei Teilsimulationen durchgeführt:

1. Druckverlustbestimmung bei isothermer Durchströmung, wie im Fall des Kanalbogens und Vergleich der Ergebnisse mit [5].

2. Simulation des thermischen Auftriebs über Druckrandbedingungen am Ein- und Austritt des Treppenhauses gemäß Abbildung 5.4 und Ermittlung des Ventilations-Volumenstroms, der sich im stationären Fall einstellt.

Nachfolgend wird das für beide Teilsimulationen verwendete Berechnungsnetz vorgestellt.

5.2.1 Vernetzung

Die Simulationen der grundlegenden Testfälle haben gezeigt, dass für die Simulationen und Auswertungen auf dem verwendeten Desktoprechner (4 Prozessoren mit jeweils 2,8 GHz, 12 GB Ram) ein Berechnungsnetz in der Größenordnung von maximal 2 Millionen Volumenzellen ideal ist. Berechnungsnetze mit mehr Volumenzellen benötigen eine deutlich höhere Berechnungszeit und auch Paraview stößt bei der gegebenen Rechnerkonfiguration an seine Grenzen. Für die CFD-Simulationen des Sicherheitstreppenhauses wird daher ein Ausschnitt des Treppenhauses ausgewählt, der mit dieser Volumenzellenanzahl darstellbar ist.
Die Geometrie des Treppenhauses lag als 3D-Modell vor. Nach [5] ist das Modell vereinfacht,

es sind keine Handläufe vorhanden und das Lichtauge ist geschlossen. Dieses Modell wird in Salomé eingelesen und die mit Randbedingungen zu belegenden Flächen (Einlass, Auslass, Wände), wie in Kapitel 4.3 beschrieben, einzeln exportiert und anschließend benannt wieder zusammengefasst. Auch das weitere Vorgehen orientiert sich an Kapitel 4.3. Das in Blockmesh erstellte Grundnetz weist eine Volumenzellengröße von 0,1 x 0,1 x 0,1 m auf. In SnappyHexMesh wird eine dreifache Verfeinerung der äußeren Begrenzungsflächen des Netzes eingestellt. Daraus resultiert eine kleinste Kantenlänge von 0,0125 m. Boundary-Layer werden nicht erstellt. Einen Ausschnitt des erstellte Netzes zeigt Abbildung 5.5. Das erstellte Netz besteht aus 1.513.979 Volumenzellen.

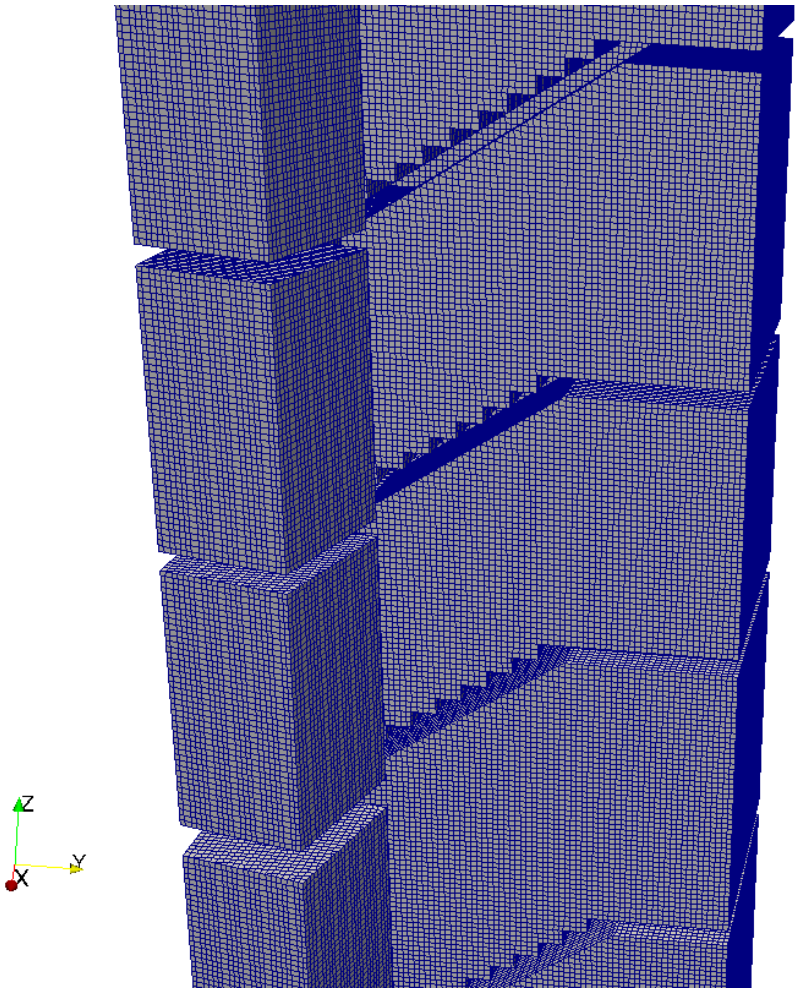


Abb. 5.5: Erstelltes Berechnungsnetz für das Sicherheitstreppenhaus

5.2.2 Druckverlustbestimmung

Die Bestimmung der Druckverluste entspricht im Wesentlichen der Simulation der Druckverluste im Kanalbogen (vgl. Kapitel 4.4.1). Am Einlass wird gemäß [5] ein konstanter Volumenstrom von 10,0864 m^3/s über eine Geschwindigkeitsrandbedingung vorgegeben. Die Fläche des Einlasses beträgt 11,08 m^2 und die Einströmgeschwindigkeit damit $c = \dot{V}/A = 0{,}91\ m/s$. Die Turbulenzintensität am Einlass wird mit 5 %, hervorgerufen durch die vorherigen Stockwerke, angenommen und k und ω mit Gleichung (4.19) und (4.20) berechnet. Alle weiteren Randbedingungen können Tabelle 5.1 entnommen werden.

Tab. 5.1: Start- und Randbedingungen für die isotherme Druckverlustbestimmung des Sicherheitstreppenhauses

	Einlass	Auslass	Wände
p in Pa	kein Gradient	Fester Wert: 0	kein Gradient
k in $\frac{m^2}{s^2}$	Fester Wert: 0,003185	kein Gradient	Wandfunkt., Start 0,003185
omega in $\frac{1}{s}$	Fester Wert: 202,676	kein Gradient	Wandfunkt., Start 202,676
nut in $\frac{kg}{ms}$	berechnet	berechnet	Wandfunkt., Start 0
U in $\frac{m}{s}$	fester Wert: U_z=0,91	kein Gradient	fester Wert: 0

Die Berechnung wird mit dem Solver „PisoFOAM“ durchgeführt, der im Testfall des Kanalbogens verifiziert werden konnte. Die Solvereinstellungen entsprechen dabei denen im Kapitel 4.4.1. Die Berechnung wird ebenfalls transient durchgeführt. Entsprechend [5] wird ein Zeitschritt 20 Sekunden nach dem Start der Simulation ausgewertet. Für die Bestimmung der Druckverluste je Geschoss werden Ebenen in das Treppenhausmodell gelegt, auf denen der Totaldruck bestimmt wird (Abb. 5.6). Wie dargestellt wird der Totaldruck jeweils zu Beginn des

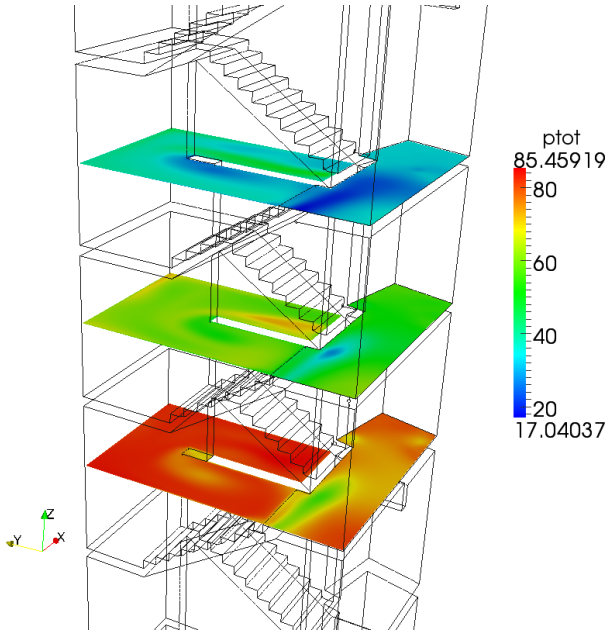


Abb. 5.6: Ebenen für die Totaldruckbestimmung im Sicherheitstreppenhaus (Totaldruck in Pascal)

Geschosses bestimmt. Abbildung 5.7 zeigt vergleichend den Verlauf des Totaldruckes der Geschosse 41-49 berechnet in OpenFOAM und Fluent. Geschoss 45 und 46 sind Sondergeschosse

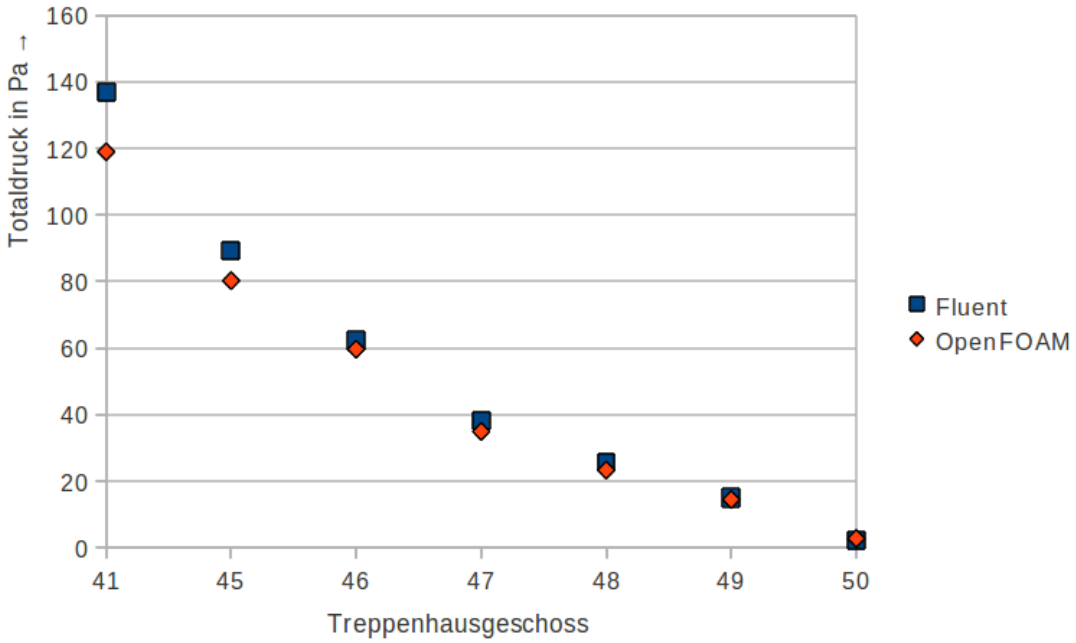


Abb. 5.7: Totaldruck beim Eintritt in die Geschossebene: Fluent Ergebnisse nach [5]

mit einer geringeren Geschosshöhe von drei Metern im Vergleich zu der normalen Geschosshöhe von 3,75 m. Der Drucksprung zum nächsten Geschoss und damit der Druckverlust sind höher. Die berechneten Werte sind bis auf den Einströmbereich mit der Fluent Berechnung aus [5] vergleichbar. Die resultierenden Druckverluste zeigt Abbildung 5.8.

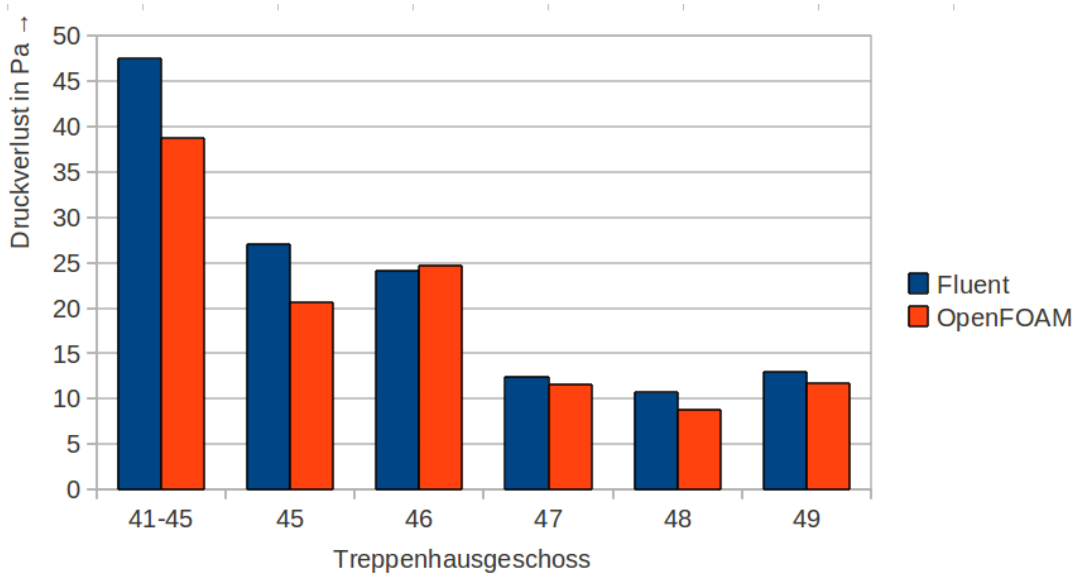


Abb. 5.8: Druckverlust über die Geschosshöhen: Fluent Ergebnisse nach [5]

Bei der OpenFOAM Berechnung ist bedingt durch den geringeren Totaldruck im Einströmbereich auch der Druckverlust zwischen den Geschossen 41-45 geringer. Zwischen den beiden

Sondergeschossen zeigen die OpenFOAM und die Fluent Berechnungen gegenläufige Ergebnisse und der Unterschied des Druckverlustes auf den Ebenen 45 und 46 ist bei der OpenFOAM Berechnung größer.
Als Vergleich soll weiterhin der spezifische Druckverlustbeiwert des gesamten Treppenhausausschitts dienen. Dieser kann nach Gleichung

$$\zeta = \frac{2\Delta p}{\Delta h \rho u^2} = \frac{2\Delta p A^2}{\Delta h \rho \dot{V}^2} \tag{5.2}$$

berechnet werden, wobei Δp den Druckverlust und Δh=35,8 m die Höhe des Treppenhausauschnitts darstellt und u eine für die Durchströmung des Treppenhauses charakteristische Strömungsgeschwindigkeit ist. Sie ergibt sich mit $u = \dot{V}/A$ aus dem auf eine charakteristische Fläche A bezogenen (Durchström-) Volumenstrom $\dot{V}$. Für die Auswertung wird als charakteristische Fläche die Grundfläche $A_G = 16,06\ m^2$ des Treppenhauses gewählt. Abbildung 5.9 zeigt den Vergleich der spezifischen Druckverlustbeiwerte für den Treppenhausausschnitt, berechnet in Fluent und OpenFOAM.

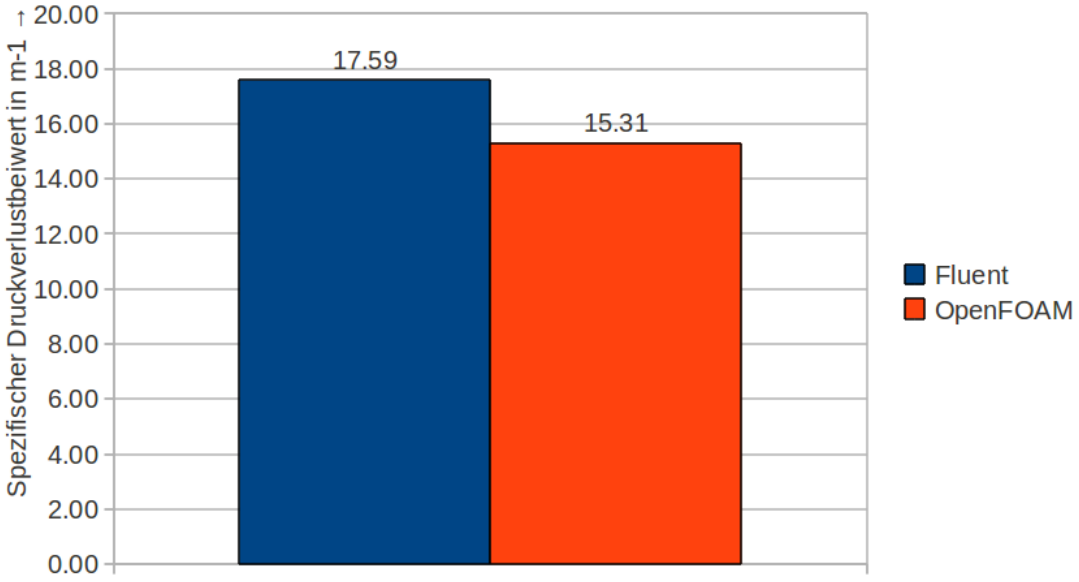


Abb. 5.9: Vergleich des spezifischen Druckverlustbeiwerts über den gesamten Treppenhausausschnitt: Fluent Ergebnis nach [5]

Die prozentuale Abweichung des spezifischen Druckverlustbeiwertes zwischen der Fluent- und OpenFOAM-Berechnung beträgt 12,96%. Das Berechnungsnetz nach [5] hat für den untersuchten Treppenhausauschnitt 6,2 Millionen Zellen, also 4,69 Millionen Zellen mehr als das in dieser Arbeit in OpenFOAM genutzte Berechnungsnetz. In [5] konnte gezeigt werden, dass eine netzunabhängige Lösung erst ab einer Volumenzellenanzahl von 11,0 Millionen Volumenzellen erreicht wird. Es ist daher davon auszugehen, dass die Netzfeinheit des OpenFOAM-Berechnungsnetzes nicht ausreichend hoch ist. Da aber gezeigt werden konnte, dass die in OpenFOAM berechneten Druckverluste über die Geschosshöhen tendenziell den Fluent Ergeb-

nissen entsprechen (vgl. Abb. 5.7) und ein größeres Berechnungsnetz aufgrund der limitierten Hardware nicht berechnet werden kann, wird an dieser Stelle auf eine weitere Verfeinerung des Berechnungsnetzes verzichtet.

5.2.3 Einfluss des thermischen Auftriebs

Der Einfluss des thermischen Auftriebs wird vereinfachend über Druckrandbedingungen abgebildet. Der in [5] untersuchte Ausschnitt des Treppenhauses ist 35,8 m hoch. Für eine Außentemperatur von -10°C ($\rho_{Aussen} = 1,3245\ kg/m^3$) und eine konstante Innenraumtemperatur von 20°C ($\rho_{Innen} = 1,1885\ kg/m^3$) ergibt sich nach Gleichung (5.1) eine Druckdifferenz von 47,8 Pa am Treppenhauskopf. Abbildung 5.10 zeigt die Zusammensetzung der Drücke für den untersuchten Treppenhausausschnitt. Da der Auftriebsdruck am Treppenhauskopf geringer ist

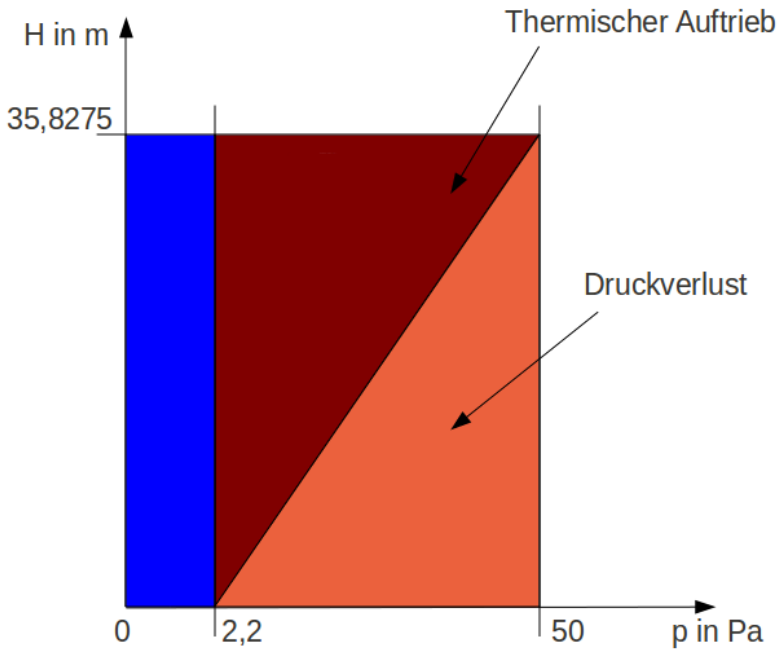


Abb. 5.10: Zusammensetzung der Drücke für den untersuchten Treppenhausausschnitt unter Berücksichtigung des thermischen Auftriebs

als der über die Druckregelklappe geregelte Überdruck von 50 Pa stellt sich abweichend zu Abbildung 5.4 ein positiver Druck am Treppenhausfuß von 2,2 Pa ein. Damit sich auch am Treppenhausfuß der geforderte Überdruck von 50 Pa einstellt, muss daher über den Zuluftvolumenstrom ein Druckverlust im Treppenhaus erzeugt werden, der eine Druckdifferenz über das Treppenhaus entsprechend der Auftriebsdruckdifferenz von 47,8 Pa erzeugt. Diese Annahme gilt allerdings nur für eine nahezu konstante Temperatur von 20°C am Treppenhauskopf. Im untersuchten winterlichen Fall strömt von unten -10°C kalte Luft in das Treppenhaus ein. Die hohe Speicherkapazität und die somit nahezu konstante Temperatur der Wände sorgt dafür, dass sich die kalte Luft bei der Durchströmung des Treppenhauses erwärmt. Bei einem Treppenhaus > 100 m ist daher davon auszugehen, dass die Temperatur am Austritt für die Betriebszeit

der Druckbelüftungsanlage im Brandfall etwa konstant bei 20°C liegt. Um diesen Einfluss am untersuchten Treppenhausabschnitt abzubilden wird die Wandtemperatur im Treppenhaus auf konstante 20°C gesetzt.
Die erforderliche Druckdifferenz kann für den Solver *buoyantPimpleFoam* über den statischen Druck ohne Berücksichtigung des hydrostatischen Anteils $p\_rgh = p + \rho gh$ vorgegeben werden. Am Eintritt in das Treppenhaus wird ein Druck entsprechend dem Auftriebsdruck von $p\_rgh = 47,8\ Pa$ (vorgegeben nach [4] über eine Totaldruck-Randbedingung) und beim Austritt aus dem Treppenhaus ein konstanter statischer Druck von $p\_rgh = 101325\ Pa$ vorgegeben. Der Druck p, der den hydrostatischen Anteil berücksichtigt, wird im Solver über $p = p\_rgh - \rho gh$ berechnet. Bei einer Höhe von 35,8 m ist es erforderlich das Druckfeld p für den Start der Berechnung zu definieren, da sich das Druckfeld sonst zu Beginn der Simulation erst einstellt und daraus instabile Strömungsverhältnisse mit Solverabbrüchen resultieren können. Das Druckfeld wird mittels des im grundlegenden Testfall „Freie Konvektion an einer Ebenen Wand“ (vgl. Kapitel 4.4.2) genutzten Tools „FunkySetFields“ initialisiert. Vorgegeben wird dabei die Gleichung $p(H) = p_0 - \rho gH$. Dabei entspricht p_0=101325 Pa, g beträgt 9,81 m^2/s und die Dichte wird entsprechend der Dichte von Luft bei 20°C $\rho_{(t=20°C)} = 1,1885\ kg/m^3$ angegeben. Die Turbulenzintensität beim Eintritt in das Treppenhaus wird als 5 % angenommen. Die gewählten Randbedingungen können zusammenfassend Tabelle 5.2 entnommen werden. Die Solver- und

Tab. 5.2: Start- und Randbedingungen für die isotherme Druckverlustbestimmung des Sicherheitstreppenhauses

	Einlass	Auslass	Wände
p in Pa	Druckhöhenprofil: $p(H) = p_0 - \rho gH$	berechnet: Start 0	berechnet: Start 0
p_rgh in Pa	Totaldruck: 101372,7996	Fester Wert: 101325	buoyantPressure: Start 0
k in $\frac{m^2}{s^2}$	Fester Wert: 0,003185	kein Gradient	Wandfunkt., Start 0,003185
omega in $\frac{1}{s}$	Fester Wert: 202,676	kein Gradient	Wandfunkt., Start 202,676
mut in $\frac{kg}{ms}$	berechnet	berechnet	Wandfunkt., Start 0
alphat in $\frac{kg}{ms}$	berechnet	berechnet	Wandfunkt., Start 0
U in $\frac{m}{s}$	pressureInletVelocity: Start 0	pressureInletOutletVelocity: Start 0	fester Wert: 0
T in K	fester Wert: 263,15	EinlassAuslass: 263,15	fester Wert: 293,15

Interpolationseinstellungen werden entsprechend Kapitel 4.4.3 gewählt. Nach 81 Sekunden treten im Treppenhaus kaum noch Änderungen in der Geschwindigkeit auf und es kann von einer nahezu stationären Strömung im Treppenhaus ausgegangen werden. Über das Tool „patchIntegrate“, welches auch im grundlegenden Testfall des thermisch induzierten Auftriebsstrahls genutzt wurde, lässt sich mittels des Befehls „patchIntegrate phi Einlass -latestTime“ der Massenstrom am Einlass des Treppenhauses bestimmen. Dabei wird das Feld des Massenstroms „phi“ über dem Einlass integriert. Es ergibt sich ein Massenstrom von 7,89 kg/s. Für den Ventilationsvolumenstrom ergibt sich über $\dot{V} = \frac{\dot{m}}{\rho_{(-10°C)}} = 5,96\ m^3/s$.

Abbildung 5.11 zeigt die Temperatur-, Geschwindigkeits- und Druckverläufe im Treppenhaus. Es zeigt sich, dass die Temperatur von 20°C im Treppenhauskopf nicht erreicht wird und die

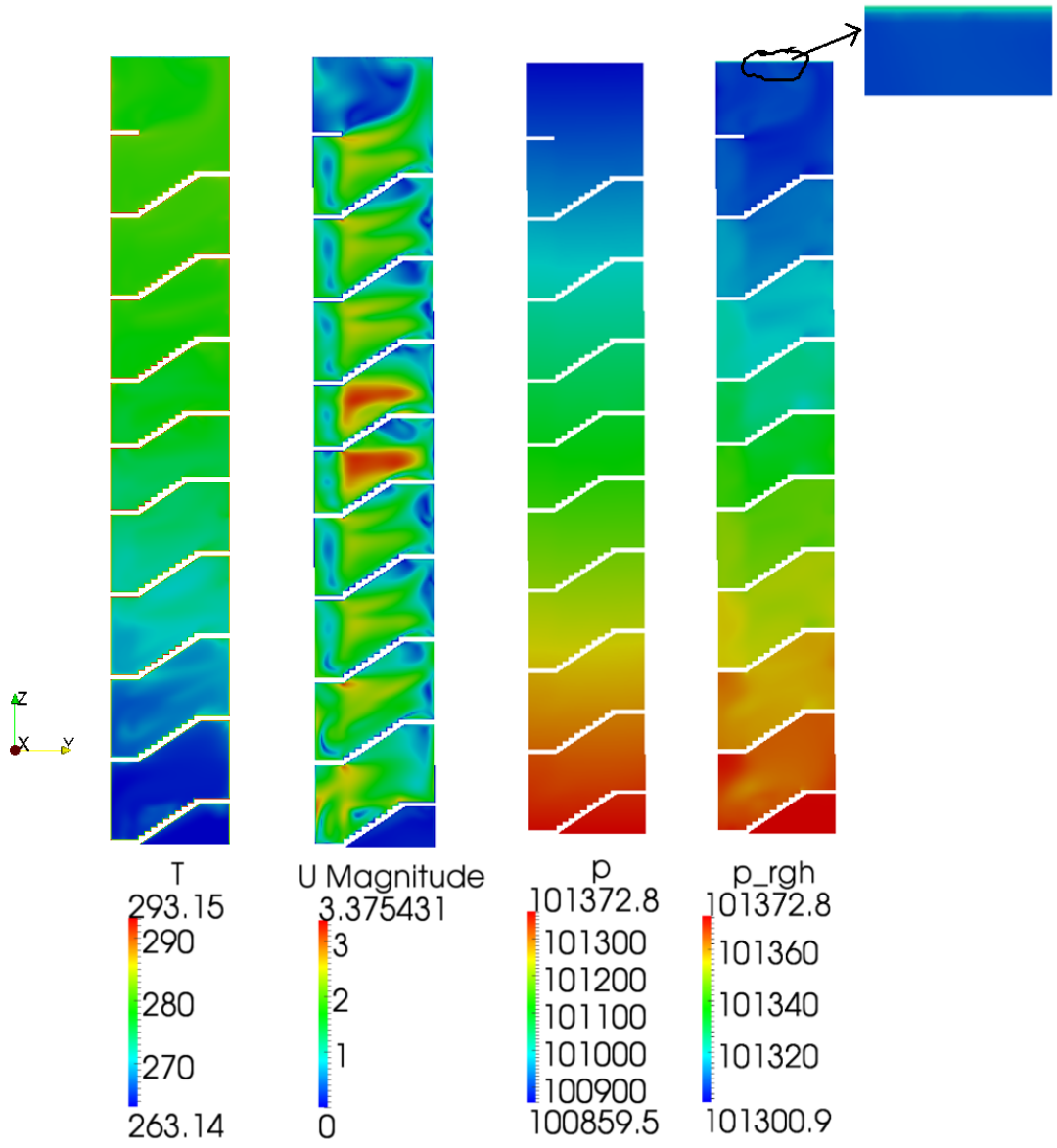


Abb. 5.11: Berechnete Strömungsbedingungen im Treppenhaus bei einer treibenden Druckdifferenz von 47,8 Pa: Temperatur in K, Geschwindigkeit in m/s, hydrostatischer Druck in Pa und statischer Druck in Pa

Annahme einer konstanten Temperatur somit nicht korrekt ist. Allerdings erwärmt sich die Luft bei der Durchströmung des Treppenhausauschnitts bis zum Treppenhauskopf um 20°C. Bei der Simulation eines gesamten Treppenhauses, wie beispielsweise dem Tower 185 mit 192 m Höhe, ist daher davon auszugehen, dass die Annahme einer konstanten Temperatur im Treppenhauskopf korrekt ist und die Berechnung auf die gezeigte Weise durchgeführt werden kann.
Die maximale Strömungsgeschwindigkeit wird in den Sondergeschossen erreicht. Bei der Betrachtung des statischen Druckes p_rgh fällt auf, dass der Druck vor dem Austritt aus dem Treppenhaus unter den am Auslass vorgegebenen Druck von 101325 Pa fällt und direkt am

Austritt sprunghaft auf den vorgegebenen Wert ansteigt. Welche Beeinflussung sich daraus für den bestimmten Volumenstrom im Treppenhaus ergibt lässt sich aufgrund der fehlenden Vergleichswerte nur schwer sagen.
Stellt man allerdings Gleichung (5.2) nach dem Druckverlust um und berechnet den resultierenden Druckverlust mit dem spezifischen Druckverlustbeiwert, ermittelt nach [5] für die Geschosse 41-50 $\zeta = 17.59\,m^{-1}$ und dem in OpenFOAM ermittelten Volumenstrom $\dot{V} = 5,96\,m^3/s$, so ergibt sich ein Druckverlust von 47,03 Pa. Mit einer Abweichung um 1,6% vom vorgegebenen Druckverlust von 47,8 Pa kann der ermittelte Volumenstrom als richtig angesehen werden.

5.3 Bewertung der Simulation des komplexen Testfalls

Die Vernetzung des Sicherheitstreppenhauses zeigt, dass die Vernetzung mit SnappyHexMesh auch für komplizierte Geometrien gut funktioniert. Allerdings wird die Geometrie für das Treppenhaus nicht in Salomé erzeugt, sondern eine vorhandene Geometrie in Salomé eingelesen und angepasst. Das Einlesen der Geometrie im STL-Format und auch die Anpassung der Geometrie ist dabei problemlos möglich. Relativ mühsam ist, wie in den grundlegenden Testfällen, die Vorgabe der Flächen für die Randbedingungen (vgl. 4.3). Die Vielzahl der vorhandenen Flächen im Fall des Treppenhauses erschwert die Zuweisung zusätzlich.
Das komplette Modell des Sicherheitstreppenhauses mit allen Randbedingungen ließ sich mit dem Solver *buoyantPimpleFoam* nicht abbilden, da es nicht gelungen ist den Druckverlust der Druckregelklappe mit den vorhandenen Randbedingungen darzustellen. Die aussichtsreichste Randbedingung ist die „porousBafflePressure“ Randbedingung, bei der sich allerdings nicht abschließend klären lassen konnte, ob diese Randbedingung mit dem Solver *buoyantPimpleFoam* kompatibel ist. Weiterhin könnte der Druckverlust durch die Abbildung einer porösen Zone im Berechnungsnetz abgebildet werden. Hierzu wäre z.B. der Solver *porousSimpleFoam* geeignet. Allerdings ist dieser Solver nur für eine stationäre inkompressible Strömung geeignet. Eine Implementierung der Gleichungen dieses Solvers in den Solver *buoyantPimpleFoam* kann hier die Lösung sein.
Weiterhin lässt sich der Einfluss des thermischen Auftriebs im Winterfall nicht abbilden. Die Referenzdichte kann nicht vorgegeben werden, sondern sie wird durch eine Mittlung über die gesamte Domain berechnet. In diesem Fall ist eine Anpassung des Solvers auf eine Referenzdichte erforderlich. Alternativ kann ein Bereich der Umgebung des Treppenhauses mit abgebildet werden. Bei einer Berücksichtigung der Umgebung des Treppenhauses wird allerdings ein sehr viel größeres Berechnungsnetz benötigt und die Zellenanzahl nimmt erheblich zu. Diese Möglichkeit kann daher nicht als optimale Lösung angesehen werden.
Die Darlegungen zeigen, dass eine weitreichende Anpassung des Solvers *buoyantPimpleFoam* für eine komplette Berechnung des Treppenhauses unter Berücksichtigung aller Randbedingungen erforderlich ist. Entsprechend der Anpassung des Solvers für die Darstellung der Rauchausbreitung im dritten grundlegenden Testfall kann die Anpassung im Rahmen dieser Arbeit nicht

geleistet werden und bleibt daher nachfolgenden Arbeiten vorbehalten.
Um die Möglichkeiten der untersuchten Solver darzustellen wurde die Simulation des Sicherheitstreppenhauses in zwei Teilsimulationen zerlegt. Die Simulation der Druckverluste im Treppenhaus bei einem festgelegten Volumenstrom und der Vergleich mit [5] zeigt, dass OpenFOAM mit dem Solver *pisoFOAM* im inkompressiblen isothermen Fall auch für komplexe Geometrien zu Fluent ähnliche Ergebnisse berechnet. Die nicht zu vernachlässigenden Abweichungen sind auf die limitierte Hardware und die damit verbundene limitierte Netzauflösung zurückzuführen. In der zweiten Teilsimulation, berechnet über Druckrandbedingungen mit dem Solver *buoyantPimpleFoam*, wurde gezeigt, welcher Volumenstrom erforderlich ist, um einen Druckverlust zu erzeugen, der dem thermischen Auftriebsdruck entspricht. Allerdings gilt dieses nur bei der idealen Annahme einer Überlagerung nach Abbildung 5.4 (linearer Verlauf des Druckverlustet, linearer Verlauf des Auftriebsdruckes, konstante Temperatur im Treppenhaus).
Abschließend lässt sich feststellen, dass sich mit den genannten Möglichkeiten von OpenFOAM die Funktion der Rauchschutzdruckanlage in der Simulation nicht überprüfen lässt. Hier ist auch die fehlende Darstellbarkeit der Rauchausbreitung (vgl. Kapitel 4.6) zu nennen, wodurch in einer Simulation nicht gezeigt werden kann, dass der Rauch zuverlässig zurückgehalten wird und nicht in das Treppenhaus eindringt.

6 Zusammenfassung, Fazit und Ausblick

Zusammenfassung

Ziel der vorliegenden Arbeit war es, die Möglichkeiten zum Einsatz von Open Source CFD Software für Strömungssimulationen im Bereich der Gebäudetechnik zu untersuchen. Zu diesem Zweck wurde zu Beginn der Arbeit eine Open Source CFD Werkzeugkette ausgewählt. Die Wahl fiel auf Salomé, BlockMesh, SnappyHexMesh, OpenFOAM und ParaFOAM. Anschließen wurden mit dieser Werkzeugkette drei grundlegende Strömungsformen aus dem Bereich der Gebäudetechnik untersucht. Die untersuchten Strömungsformen sind die turbulente Durchströmung eines Kanalbogens, der freie Wärmeübergang an einer ebenen Wand und die freie Turbulenz und Rauchschichtung am Beispiel des thermisch induzierten Auftriebsstrahls. Die Bedienbarkeit und Anwendbarkeit der untersuchten Programme wurde dabei für die Geometrieerstellung, Vernetzung und die Strömungsmodellerstellung jeweils vergleichend dargestellt. Zur Validierung wurden Fluent Ergebnisse und/oder experimentelle Ergebnisse genutzt. Abschließend wurde ein komplexes Modell eines Sicherheitstreppenhauses mit den Möglichkeiten der untersuchten Programme simuliert.

Bei der Simulation der drei grundlegenden Testfälle zeigte sich, dass OpenFOAM physikalisch richtige, mit Fluent vergleichbare Ergebnisse liefern kann und somit mit den untersuchten Programmen vollwertige CFD Simulationen durchgeführt werden können. Allerdings war der Aufwand, zu den Ergebnissen zu kommen, im Vergleich zu ANSYS Fluent erheblich größer. Dieses begann mit der Erstellung der Geometrie in Slomé, wobei schon die Erstellung der einfachen Geometrien der Testfälle, bedingt durch den Export einzelner STL-Flächen und die wenig intuitive Arbeitsweise, viel Zeit kostete. Nach einer gewissen Einarbeitungszeit lassen sich mit Salomé, BlockMesh und SnappyHexMesh allerdings gute Berechnungsnetze erstellen. Es zeigte sich aber, dass die Definition der Randflächen über STL-Oberflächen teilweise ungenau erfolgte und die Randflächen daher zu groß oder klein ausfielen.

Die Definition der Solvereinstellungen war anhand der mitgelieferten Tutorials für einfache Testfälle schnell durchführbar. Bei komplexeren Simulationen fielen allerdings immer mehr einzelne Definitionsdateien an und es war schwer, den Überblick zu behalten. Auch waren Tools, wie z.B. „WallHeatFlux“, teilweise nicht auf die mitgelieferten Solver anwendbar und mussten angepasst werden.

Einfache Anpassungen der Solver ließen sich relativ schnell durchführen. So konnte im dritten Testfall der Solver *buoyantPimpleFoam* um einen volumetrischen Wärmequellterm erweitert werden. Andererseits musste festgestellt werden, dass komplexere Anpassungen im Solver viel Zeit in Anspruch nehmen und schon das Verständnis der Lösungsschritte im Solver durch die objektbasierte Programmierung und die fehlende Dokumentation schwierig ist. So ist es im dritten Testfall nicht gelungen, die Rauchausbreitung mittels einer skalare Größe oder einer Spezies kenntlich zu machen.

In der abschließenden Simulation des Sicherheitstreppenhauses konnte ebenfalls kein Strö-

mungsmodell für das gesamte Treppenhaus erstellt werden. Der Hauptgrund hierfür waren fehlende oder unzureichend dokumentierte Randbedingungen und eine nicht festlegbare Referenzdichte. Eine weitreichende Anpassung des Solvers *buoyantPimpleFoam* für eine komplette Berechnung des Treppenhauses unter Berücksichtigung aller Randbedingungen wäre daher erforderlich. Diese Anpassung kann im Rahmen dieser Arbeit nicht geleistet werden und bleibt daher nachfolgenden Arbeiten vorbehalten.
Um die Möglichkeiten der in dieser Arbeit bisher untersuchten Solver darzustellen, wurde die Simulation des Sicherheitstreppenhauses in zwei Teilsimulationen zerlegt. Die Simulation der Druckverluste im Treppenhaus bei einem festgelegten Volumenstrom und der Vergleich mit [5] zeigte, dass OpenFOAM mit dem Solver *pisoFOAM* im incompressiblen isothermen Fall auch für komplexe Geometrien mit Fluent vergleichbare Ergebnisse berechnen kann. Die nicht zu vernachlässigenden Abweichungen sind auf die limitierte Hardware und die damit verbundene limitierte Netzauflösung zurückzuführen. In der zweiten Teilsimulation, berechnet über Druckrandbedingungen mit dem Solver *buoyantPimpleFoam*, hat sich weiterhin gezeigt, welcher Volumenstrom erforderlich ist, um einen Druckverlust zu erzeugen, der dem thermischen Auftriebsdruck entspricht. Allerdings gilt dieses nur bei idealen Annahmen und ersetzt keine komplette Simulation des Sicherheitstreppenhauses mit allen Randbedingungen.

Fazit und Ausblick

Die vorliegende Arbeit hat gezeigt, dass OpenFOAM in der Praxis der Gebäudesimulation durchaus einsetzbar ist. Allerdings sind die vorhandenen Solver nur für relativ einfache Strömungsmodelle ohne Anpassung einsetzbar. Für komplexere Strömungsformen (wie beispielsweise das Sicherheistreppenhaus) müssen Solver angepasst bzw. aus mehreren Solvern der gewünschte Solver erstellt werden. In diesem Fall sind sehr gute C++ Kenntnisse sowie ein gutes Verständnis für die strömungsmechanischen Gleichungen, Gleichungslöser und Diskretisierungsschemata der CFD erforderlich. Erschwerlich für das Verständnis der Arbeitsweise der Solver ist dabei die objektorientierte Programmierung, da dabei auf sehr viele Objekte an vielen Orten zurückgegriffen wird und die Lösungsschritte somit schwer nachzuvollziehen sind. Auch die im Vergleich zu Fluent teilweise nicht vorhandenen Randbedingungen und die unzureichende Dokumentation erschwert die Arbeit mit OpenFOAM. Der Aufbau komplexer Simulationsmodelle mit vielen Randbedingungen im Termingeschäft ist somit kaum vorstellbar.
Trotzdem sollte die Entwicklung von OpenFoam weiter verfolgt werden. Mit jeder neuen OpenFoam-Version kommen weitere Solver und Randbedingungen hinzu und OpenFOAM ist somit ohne Anpassungen für eine größere Anzahl an Strömungsproblemen einsetzbar. Weiterhin kann OpenFOAM für Strömungsprobleme genutzt werden, zu denen in Fluent/CFX kein entsprechendes Strömungsmodell umgesetzt werden kann. Hier ist der quelloffene, anpassbare Code klar von Vorteil.
Insofern steht zu hoffen, dass sich in Zukunft der Ansatz der quelloffenen Software weiter-

entwickelt und der Einsatz für immer mehr Strömungsformen ohne große Änderungen an den Solvern möglich ist.

Literaturverzeichnis

[1] ALBERS, K.-J. ; RAHN, B. : Strömungsverhältnisse in einem Sicherheistreppenraum - Einfluss der Thermik / Hochschule Esslingen. 2003. – Forschungsbericht

[2] AMPOFO, F. ; KARAYIANNIS, T. : Low turbulence natural convection in an air filled square cavity: Part I: the thermal and fluid flow fields. In: *International Journal of HEAT and MASS TRANSFER* 43 (2000), S. 849–866

[3] AMPOFO, F. ; KARAYIANNIS, T. : Experimental bechmark data for turbulent natural convection in an air filled square cavity. In: *International Journal of HEAT and MASS TRANSFER* 46 (2003), S. 3551–3572

[4] BAKKER, A. : *Lecture 6 -Boundary Conditions*. `http://www.bakker.org/dartmouth06/engs150/`, Abruf: 10.August 2012

[5] BEZHOLT, C. : *Numerische Bestimmung von Druckverlusten bei der Durchströmung von Treppenräumen unterschiedlicher Geometrien*, Georg-Simon-Ohm-Hochschule für angewandte Wissenschaften - Fachhochschule Nürnberg, Diplomarbeit, 2012

[6] BUNDESMINISTERIUM FÜR WIRTSCHAFT UND TECHNOLOGIE: *Open-Source-Software: Ein Leitfaden für kleine und mittlere Unternehmen*. `http://oss-broschuere.berlios.de/broschuere/broschuere-de.html#impressum`, Abruf: 29.Mai 2012

[7] BÖSWIRTH, L. : *Technische Strömungslehre: Lehr- und Übungsbuch*. Auflage 8. VIEWEG+TEUBNER, 2010. – ISBN 978–3–8348–0523–2

[8] CFD ONLINE: *Best practice guidelines for turbomachinery CFD*. `www.cfd-online.com/Wiki/Best_practice_guidelines_for_turbomachinery_CFD`, Abruf: 26.März 2012

[9] CFD ONLINE: *Y plus wall distance estimation*. `www.cfd-online.com/Wiki/Y_plus_wall_distance_estimation`, Abruf: 27.März 2012

[10] EICHELBERGER: *Rauchfreihaltung von Flucht- und Rettungswegen: Rauchschutz - Druckanlagen*. 2012

[11] FERZINGER, J. H. ; PERIC, M. : *Numerische Strömungsmechanik*. Auflage 3. Springer, 2008. – ISBN 978–3–540–67586–0

[12] FIEDLER, D.-I. E.: Richtig berechnet: Über die Druckverlustbestimmung von Hand und mittels Strömungssimulationen. In: *tab Das Fachmagazin der TGA-Branche: Fachbeiträge 669, Gebäudesimulation* (2009)

[13] FLUENT: *User's Guide*. Release 6.2. Lebanon, NH: Fluent Inc., 2005

[14] GSCHAIDER, B. : *Swak4Foam*. `http://openfoamwiki.net/index.php/Contrib/swak4Foam`, Abruf: 18.Mai 2012

[15] HERWIG, H. ; MOSCHALLSKI, A. : *Wärmeübertragung: Physikalische Grundlagen*. Auflage 2. Springer Verlag, 2009. – ISBN 978–3–8348–0755–7

[16] IDELCHIK, I. E.: *Handbook of Hydraulic Resistance*. Auflage 3. Jaico Publishing House, 2005. – ISBN 978–8179921180

[17] KABELAC, S. ; U.A.: *VDI-WÄRMEATLAS*. Auflage 10. Springer-Verlag Berlin Heidelberg New York, 2005. – ISBN 978–3–540–25504–8

[18] KING, A. : *OpenFOAM - Opensource and CFD*. `http://physics.ucsd.edu/students/courses/spring2009/physics142/Labs/Lab8/Tutorial1.pdf`, Abruf: 10.August 2012

[19] KÜMMEL, W. : *Technische Strömungsmechanik: Theorie und Praxis*. Auflage 3. TEUBNER, 2007. – ISBN 978–3–8351–0141–8

[20] LECHELER, S. : *Numerische Strömungsberechnung*. Auflage 1. Vieweg + Teubner, 2009. – ISBN 978–3–8348–0439–6

[21] OERTEL, H. ; LAURIN, E. : *Numerische Strömungsmechanik: Grundgleichungen - Lösungsmethoden - Softwarebeispiele*. Auflage 2. Vieweg Verlag, 2002. – ISBN 3–528–03936–1

[22] OPENCFD: *OpenFOAM The Open Source CFD Toolbox: Programmer's Guide*. Auflage 2.0.0. Boston, USA: OpenCFD Limited, 2011. – Programmierhandbuch

[23] OPENCFD: *OpenFOAM The Open Source CFD Toolbox: User Guide*. Auflage 2.0.0. Boston, USA: OpenCFD Limited, 2011. – Benutzerhandbuch

[24] SCHLICHTING, H. : *Boundary Layer Theory*. Auflage 7. Springer, 1997. – ISBN 0–07–055334–3

[25] SIEKMANN, H. E.: *Strömungslehre für den Maschinenbau*. Auflage 2. Springer, 2008. – ISBN 978–3–540–73989–0

[26] TIAN, T. : *OpenFOAM - CFD Toolkit C++*. `http://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_architektur/ibk/institute/news/o_IntegrPlanung03_2012/18_openfoam_cfdtoolkit_tian.pdf`, Abruf: 13. August 2012

[27] TRUCKENBROT, E. : *Fluidmechanik Band 1: Grundlagen und elementare Strömungsvorgänge dichtebeständiger Fluide*. Auflage 4. Springer-Verlag Berlin Heidelberg New York, 1996. – ISBN 973–3–540–79017–4

[28] TU, J. ; YEOH, G. H. ; LIU, C. : *Computational Fluid Dynamics: A Practical Approach*. Auflage 1. Elsevier, 2008. – ISBN 978–0–7506–8563–4

[29] WAGNER, W. : *Strömung und Druckverlust*. Auflage 3. Vogel Buchverlag, 1992. – ISBN 3–8023–1462–X

[30] WINKLER, T. : Beitrag zum Schichtungsverhalten von Brandrauch. In: *Fortschrittsberichte VDI: Reihe 4: Bauingenieurwesen* (2011)

Anhang

A Kanalbogen

A.1 BlockMeshDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
    ( -0.5 -0.5 -0.5 ) //0
    (  10  -0.5 -0.5 ) //1
    (  11  -1.5 -0.5 ) //2
    (  11  -12  -0.5 ) //3
    (  13  -12  -0.5 ) //4
    (  13  -1.5 -0.5 ) //5
    (  10   1.5 -0.5 ) //6
    ( -0.5  1.5 -0.5 ) //7
    ( -0.5 -0.5  1.5 ) //8
    (  10  -0.5  1.5 ) //9
    (  11  -1.5  1.5 ) //10
    (  11  -12   1.5 ) //11
    (  13  -12   1.5 ) //12
    (  13  -1.5  1.5 ) //13
    (  10   1.5  1.5 ) //14
    ( -0.5  1.5  1.5 ) //15
    (  11  -52   1.5 ) //16
    (  13  -52   1.5 ) //17
    (  11  -52  -0.5 ) //18
    (  13  -52  -0.5 ) //19
);

blocks
(
    hex (0 1 6  7  8  9  14 15) (200 40  40)   simpleGrading (1 1 1)
    hex (1 2 5  6  9  10 13 14) (60  40  40)   simpleGrading (1 1 1)
    hex (2 3 4  5  10 11 12 13) (200 40  40)   simpleGrading (1 1 1)
    hex (4 3 18 19 12 11 16 17) (40  500 40)   simpleGrading (1 1 1)
);

edges
(
    arc 2 1   ( 10.5 -0.633974596 -0.5 )
    arc 5 6   ( 11    1.098076211 -0.5 )
```

```
55      arc 10 9  ( 10.5 -0.633974596  1.5 )
56      arc 13 14 ( 11    1.098076211  1.5 )
57  );
58
59  boundary
60  (
61  );
62
63  // ************************************************************************* //
```

A.2 SnappyHexMeshDict

```
1   /*--------------------------------*- C++ -*----------------------------------*\
2   | =========                 |                                                 |
3   | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4   |  \\    /   O peration     | Version:  2.0.1                                 |
5   |   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
6   |    \\/     M anipulation  |                                                 |
7   \*---------------------------------------------------------------------------*/
8   FoamFile
9   {
10      version     2.0;
11      format      ascii;
12      class       dictionary;
13      object      autoHexMeshDict;
14  }
15
16  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18  // Which of the steps to run
19  castellatedMesh true;
20  snap            true;
21  addLayers       true;
22
23
24  // Geometry. Definition of all surfaces. All surfaces are of class
25  // searchableSurface.
26  // Surfaces are used
27  // - to specify refinement for any mesh cell intersecting it
28  // - to specify refinement for any mesh cell inside/outside/near
29  // - to 'snap' the mesh boundary to the surface
30  geometry
31  {
32      kanalbogen_wb_50m.stl
33      {
34          type triSurfaceMesh;
35          name kanalbogen;
36          regions
37          {
38              walls
39              {
40              name wall;
41              }
42              inlet
43              {
44              name in;
45              }
46              outlet
47              {
```

```
48              name out;
49              }
50          }
51      }
52  };
53
54
55
56  // Settings for the castellatedMesh generation.
57  castellatedMeshControls
58  {
59
60      // Refinement parameters
61      // ~~~~~~~~~~~~~~~~~~~~~~
62
63      // While refining maximum number of cells per processor. This is basically
64      // the number of cells that fit on a processor. If you choose this too small
65      // it will do just more refinement iterations to obtain a similar mesh.
66      maxLocalCells 1000000;
67
68      // Overall cell limit (approximately). Refinement will stop immediately
69      // upon reaching this number so a refinement level might not complete.
70      // Note that this is the number of cells before removing the part which
71      // is not 'visible' from the keepPoint. The final number of cells might
72      // actually be a lot less.
73      maxGlobalCells 10000000;
74
75      // The surface refinement loop might spend lots of iterations refining just a
76      // few cells. This setting will cause refinement to stop if <= minimumRefine
77      // are selected for refinement. Note: it will at least do one iteration
78      // (unless the number of cells to refine is 0)
79      minRefinementCells 0;
80
81      // Allow a certain level of imbalance during refining
82      // (since balancing is quite expensive)
83      // Expressed as fraction of perfect balance (= overall number of cells /
84      // nProcs). 0=balance always.
85      maxLoadUnbalance 0;
86
87      // Number of buffer layers between different levels. (Zellwachstum)
88      // 1 means normal 2:1 refinement restriction, larger means slower
89      // refinement.
90      nCellsBetweenLevels 1;
91
92
93
94      // Explicit feature edge refinement
95      // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
96
97      // Specifies a level for any cell intersected by its edges.
98      // This is a featureEdgeMesh, read from constant/triSurface for now.
99      features
100     (
101         {
102             file "kanalbogen_wb_50m.eMesh"; //FeatureEdgeMesh
103             level 0; //level of refinement
104         }
105     );
106
```

```
107
108
109     // Surface based refinement
110     // ~~~~~~~~~~~~~~~~~~~~~~~~
111
112     // Specifies two levels for every surface. The first is the minimum level,
113     // every cell intersecting a surface gets refined up to the minimum level.
114     // The second level is the maximum level. Cells that 'see' multiple
115     // intersections where the intersections make an
116     // angle > resolveFeatureAngle get refined up to the maximum level.
117
118     refinementSurfaces
119     {
120         kanalbogen
121         {
122             // Surface-wise min and max refinement level
123             level (0 0);
124         regions
125         {
126            walls
127            {
128            level (0 0);
129            }
130            inlet
131            {
132            level (0 0);
133            }
134            outlet
135            {
136            level (0 0);
137            }
138         }
139         }
140     }
141
142     resolveFeatureAngle 30;
143
144
145     // Region-wise refinement
146     // ~~~~~~~~~~~~~~~~~~~~~~
147
148     // Specifies refinement level for cells in relation to a surface. One of
149     // three modes
150     // - distance. 'levels' specifies per distance to the surface the
151     //   wanted refinement level. The distances need to be specified in
152     //   descending order.
153     // - inside. 'levels' is only one entry and only the level is used. All
154     //   cells inside the surface get refined up to the level. The surface
155     //   needs to be closed for this to be possible.
156     // - outside. Same but cells outside.
157
158     refinementRegions
159     {
160     }
161
162
163     // Mesh selection
164     // ~~~~~~~~~~~~~~
165
```

```
166     // After refinement patches get added for all refinementSurfaces and
167     // all cells intersecting the surfaces get put into these patches. The
168     // section reachable from the locationInMesh is kept.
169     // NOTE: This point should never be on a face, always inside a cell, even
170     // after refinement.
171     // This is an outside point locationInMesh (-0.033 -0.033 0.0033);
172     locationInMesh (1 0.5 0.5); // Inside point
173 
174     // Whether any faceZones (as specified in the refinementSurfaces)
175     // are only on the boundary of corresponding cellZones or also allow
176     // free-standing zone faces. Not used if there are no faceZones.
177     allowFreeStandingZoneFaces true;
178 }
179 
180 
181 
182 // Settings for the snapping.
183 snapControls
184 {
185     //- Number of patch smoothing iterations before finding correspondence
186     //  to surface
187     nSmoothPatch 3;
188 
189     //- Relative distance for points to be attracted by surface feature point
190     //  or edge. True distance is this factor times local
191     //  maximum edge length.
192     tolerance 1.0;
193 
194     //- Number of mesh displacement relaxation iterations.
195     nSolveIter 300;
196 
197     //- Maximum number of snapping relaxation iterations. Should stop
198     //  before upon reaching a correct mesh.
199     nRelaxIter 5;
200 
201     //- Highly experimental and wip: number of feature edge snapping
202     //  iterations. Leave out altogether to disable.
203     nFeatureSnapIter 20;
204 }
205 
206 
207 
208 
209 
210 // Settings for the layer addition.
211 addLayersControls
212 {
213     // Are the thickness parameters below relative to the undistorted
214     // size of the refined cell outside layer (true) or absolute sizes (false).
215     relativeSizes false;
216 
217     // Per final patch (so not geometry!) the layer information
218     layers
219     {
220         wall
221         {
222             nSurfaceLayers 16;
223         }
224     }
```

```
225
226     // Expansion factor for layer mesh
227     expansionRatio 1.05;
228
229
230     //- Wanted thickness of final added cell layer. If multiple layers
231     //  is the thickness of the layer furthest away from the wall.
232     //  See relativeSizes parameter.
233     finalLayerThickness 0.025;
234
235     //- Minimum thickness of cell layer. If for any reason layer
236     //  cannot be above minThickness do not add layer.
237     //  See relativeSizes parameter.
238     minThickness 0.0014;
239
240     //- If points get not extruded do nGrow layers of connected faces that are
241     //  also not grown. This helps convergence of the layer addition process
242     //  close to features.
243     nGrow 0;
244
245
246     // Advanced settings
247
248     //- When not to extrude surface. 0 is flat surface, 90 is when two faces
249     //  make straight angle.
250     featureAngle 30;
251
252     //- Maximum number of snapping relaxation iterations. Should stop
253     //  before upon reaching a correct mesh.
254     nRelaxIter 5;
255
256     // Number of smoothing iterations of surface normals
257     nSmoothSurfaceNormals 1;
258
259     // Number of smoothing iterations of interior mesh movement direction
260     nSmoothNormals 3;
261
262     // Smooth layer thickness over surface patches
263     nSmoothThickness 10;
264
265     // Stop layer growth on highly warped cells
266     maxFaceThicknessRatio 0.5;
267
268     // Reduce layer growth where ratio thickness to medial
269     // distance is large
270     maxThicknessToMedialRatio 0.3;
271
272     // Angle used to pick up medial axis points
273     minMedianAxisAngle 90;
274
275     // Create buffer region for new layer terminations
276     nBufferCellsNoExtrude 0;
277
278
279     // Overall max number of layer addition iterations. The mesher will exit
280     // if it reaches this number of iterations; possibly with an illegal
281     // mesh.
282     nLayerIter 50;
283
```

```
284     // Max number of iterations after which relaxed meshQuality controls
285     // get used. Up to nRelaxIter it uses the settings in meshQualityControls,
286     // after nRelaxIter it uses the values in meshQualityControls::relaxed.
287     nRelaxedIter 20;
288 }
289
290
291
292 // Generic mesh quality settings. At any undoable phase these determine
293 // where to undo.
294 meshQualityControls
295 {
296     //- Maximum non-orthogonality allowed. Set to 180 to disable.
297     maxNonOrtho 65;
298
299     //- Max skewness allowed. Set to <0 to disable.
300     maxBoundarySkewness 20;
301     maxInternalSkewness 4;
302
303     //- Max concaveness allowed. Is angle (in degrees) below which concavity
304     //  is allowed. 0 is straight face, <0 would be convex face.
305     //  Set to 180 to disable.
306     maxConcave 80;
307
308     //- Minimum pyramid volume. Is absolute volume of cell pyramid.
309     //  Set to a sensible fraction of the smallest cell volume expected.
310     //  Set to very negative number (e.g. -1E30) to disable.
311     minVol 1e-13;
312
313     //- Minimum quality of the tet formed by the face-centre
314     //  and variable base point minimum decomposition triangles and
315     //  the cell centre.  Set to very negative number (e.g. -1E30) to
316     //  disable.
317     //     <0 = inside out tet,
318     //      0 = flat tet
319     //      1 = regular tet
320     minTetQuality 1e-30;
321
322     //- Minimum face area. Set to <0 to disable.
323     minArea -1;
324
325     //- Minimum face twist. Set to <-1 to disable. dot product of face normal
326     //- and face centre triangles normal
327     minTwist 0.05;
328
329     //- minimum normalised cell determinant
330     //- 1 = hex, <= 0 = folded or flattened illegal cell
331     minDeterminant 0.001;
332
333     //- minFaceWeight (0 -> 0.5)
334     minFaceWeight 0.05;
335
336     //- minVolRatio (0 -> 1)
337     minVolRatio 0.01;
338
339     //must be >0 for Fluent compatibility
340     minTriangleTwist 0.000001;
341
342     //- if >0 : preserve single cells with all points on the surface if the
```

```
343     //  resulting volume after snapping (by approximation) is larger than
344     //  minVolCollapseRatio times old volume (i.e. not collapsed to flat cell).
345     //  If <0 : delete always.
346     //minVolCollapseRatio 0.5;
347
348
349     // Advanced
350
351     //- Number of error distribution iterations
352     nSmoothScale 4;
353     //- amount to scale back displacement at error points
354     errorReduction 0.75;
355
356
357
358     // Optional : some meshing phases allow usage of relaxed rules.
359     // See e.g. addLayersControls::nRelaxedIter.
360     relaxed
361     {
362         //- Maximum non-orthogonality allowed. Set to 180 to disable.
363         maxNonOrtho 75;
364     }
365 }
366
367
368 // Advanced
369
370 // Flags for optional output
371 // 0 : only write final meshes
372 // 1 : write intermediate meshes
373 // 2 : write volScalarField with cellLevel for postprocessing
374 // 4 : write current intersections as .obj files
375 debug 0;
376
377
378 // Merge tolerance. Is fraction of overall bounding box of initial mesh.
379 // Note: the write tolerance needs to be higher than this.
380 mergeTolerance 1E-6;
381
382
383 // ************************************************************************* //
```

A.3 Allrun Skript

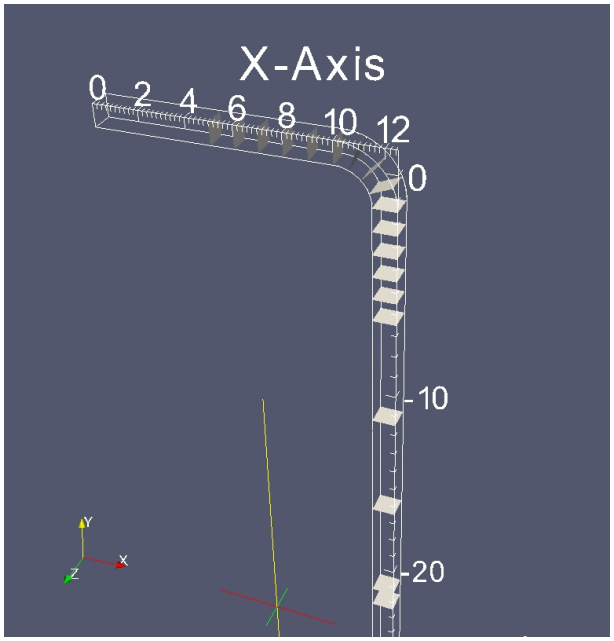
```
1 . $WM_PROJECT_DIR/bin/tools/RunFunctions
2 runApplication blockMesh
3 runApplication surfaceFeatureExtract -includedAngle 150 constant/triSurface/kanalbogen_wb_50m.
      stl kanalbogen_wb_50m
4 runApplication snappyHexMesh -overwrite
```

A.4 Skriptdatei Residuals

```
1 set logscale y
2 set title "Residuals"
3 set ylabel 'Residual'
4 set xlabel 'Iteration'
5 plot "< cat log.pisoFoam | grep 'Solving for Ux' | cut -d' ' -f9" title 'Ux' with lines,\
6 "< cat log.pisoFoam | grep 'Solving for Uy' | cut -d' ' -f9" title 'Uy' with lines,\
```

```
"< cat log.pisoFoam | grep 'Solving for Uz' | cut -d' ' -f9" title 'Uz' with lines,\
"< cat log.pisoFoam | grep 'Solving for omega' | cut -d' ' -f9" title 'omega' with lines,\
"< cat log.pisoFoam | grep 'Solving for k' | cut -d' ' -f9" title 'k' with lines,\
"< cat log.pisoFoam | grep 'Solving for p' | cut -d' ' -f9 | tr -d ','" title 'p' with lines
pause 1
reread
```

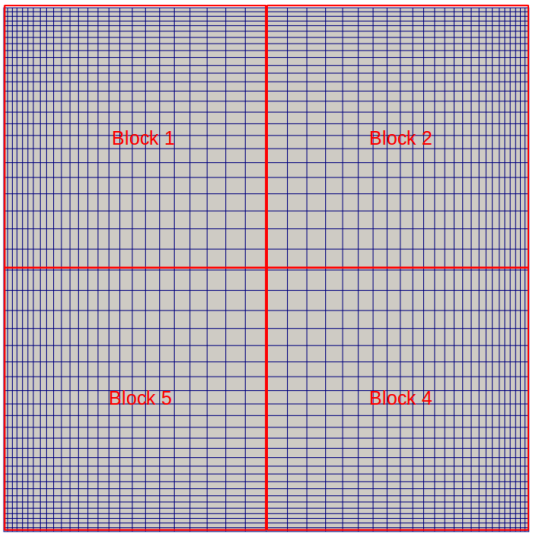
A.5 Ebenen für die Druckverlustbestimmung



Die Abbildung zeigt die 20 Schnittebenen - 5 vor dem Bogen, 5 im Bogen und 10 nach dem Bogen - in dem Strömungskanal, auf denen das Flächen-Integral des Totaldruckes bestimmt und ausgewertet wird.

B Freie Konvektion

B.1 Berechnungsnetz 2



Die Abbildung zeigt das zweite Berechnungsnetz. Das Berechnungsgebiet ist in 4 Blöcke zerlegen (rot markiert). Für die einzelnen Blöcke wurde eine Verdichtung der Zellen in Richtung Wand von 0,2 durchgeführt, wodurch die Zellenhöhe zum Rand hin abnimmt und eine größere Anzahl an Stützstellen im wandnahen Bereich vorhanden ist. Allerdings ist das innere Berechnungsgebiet, nicht sehr viel feiner aufgelöst als beim ersten Berechnungsnetz.

B.2 wallHeatFluxRho

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 1991-2009 OpenCFD Ltd.
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software; you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by the
    Free Software Foundation; either version 2 of the License, or (at your
    option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM; if not, write to the Free Software Foundation,
    Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
```

```
24
25 Application
26     wallHeatFluxRho
27
28 Description
29     Calculates and writes the heat flux for all patches as the boundary field
30     of a volScalarField and also prints the integrated flux for all wall
31     patches.
32
33 \*---------------------------------------------------------------------------*/
34
35 #include "fvCFD.H"
36 #include "basicRhoThermo.H"
37 #include "RASModel.H"
38 #include "wallFvPatch.H"
39
40 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
41
42 int main(int argc, char *argv[])
43 {
44     timeSelector::addOptions();
45     #include "setRootCase.H"
46     #include "createTime.H"
47     instantList timeDirs = timeSelector::select0(runTime, args);
48     #include "createMesh.H"
49
50     forAll(timeDirs, timeI)
51     {
52         runTime.setTime(timeDirs[timeI], timeI);
53         Info<< "Time = " << runTime.timeName() << endl;
54         mesh.readUpdate();
55
56         #include "createFields.H"
57
58         surfaceScalarField heatFlux =
59             fvc::interpolate(RASModel->alphaEff())*fvc::snGrad(h);
60
61         const surfaceScalarField::GeometricBoundaryField& patchHeatFlux =
62             heatFlux.boundaryField();
63
64         Info<< "\nWall heat fluxes [W]" << endl;
65         forAll(patchHeatFlux, patchi)
66         {
67             if (isA<wallFvPatch>(mesh.boundary()[patchi]))
68             {
69                 Info<< mesh.boundary()[patchi].name()
70                     << " "
71                     << sum
72                        (
73                            mesh.magSf().boundaryField()[patchi]
74                           *patchHeatFlux[patchi]
75                        )
76                     << endl;
77             }
78         }
79         Info<< endl;
80
81         volScalarField wallHeatFluxRho
82         (
```

```
83              IOobject
84              (
85                  "wallHeatFluxRho",
86                  runTime.timeName(),
87                  mesh
88              ),
89              mesh,
90              dimensionedScalar("wallHeatFluxRho", heatFlux.dimensions(), 0.0)
91          );
92
93          forAll(wallHeatFluxRho.boundaryField(), patchi)
94          {
95              wallHeatFluxRho.boundaryField()[patchi] = patchHeatFlux[patchi];
96          }
97
98          wallHeatFluxRho.write();
99      }
100
101     Info<< "End -> WallHeatFluxRho Mike" << endl;
102
103     return 0;
104 }
105
106 // ************************************************************************* //
```

B.3 createFields.H

```
1       Info<< "Reading thermophysical properties\n" << endl;
2
3       autoPtr<basicRhoThermo> pThermo
4       (
5           basicRhoThermo::New(mesh)
6       );
7       basicRhoThermo& thermo = pThermo();
8
9       volScalarField rho
10      (
11          IOobject
12          (
13              "rho",
14              runTime.timeName(),
15              mesh,
16              IOobject::READ_IF_PRESENT,
17              IOobject::AUTO_WRITE
18          ),
19          thermo.rho()
20      );
21
22      volScalarField& p = thermo.p();
23      volScalarField& h = thermo.h();
24      const volScalarField& psi = thermo.psi();
25
26
27      Info<< "Reading field U\n" << endl;
28      volVectorField U
29      (
30          IOobject
31          (
32              "U",
```

```
33              runTime.timeName(),
34              mesh,
35              IOobject::MUST_READ,
36              IOobject::AUTO_WRITE
37          ),
38          mesh
39      );
40
41      #include "compressibleCreatePhi.H"
42
43
44      Info<< "Creating turbulence model\n" << endl;
45      autoPtr<compressible::turbulenceModel> turbulence
46      (
47          compressible::turbulenceModel::New
48          (
49              rho,
50              U,
51              phi,
52              thermo
53          )
54      );
55
56      Info<< "Calculating field g.h\n" << endl;
57      volScalarField gh("gh", g & mesh.C());
58      surfaceScalarField ghf("ghf", g & mesh.Cf());
59
60      Info<< "Reading field p_rgh\n" << endl;
61      volScalarField p_rgh
62      (
63          IOobject
64          (
65              "p_rgh",
66              runTime.timeName(),
67              mesh,
68              IOobject::MUST_READ,
69              IOobject::AUTO_WRITE
70          ),
71          mesh
72      );
73
74      // Force p_rgh to be consistent with p
75      p_rgh = p - rho*gh;
76
77      Info<< "Creating field DpDt\n" << endl;
78      volScalarField DpDt
79      (
80          "DpDt",
81          fvc::DDt(surfaceScalarField("phiU", phi/fvc::interpolate(rho)), p)
82      );
```

C Rauchschichtung

C.1 buoyantPimpleFoamWithSource.C

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2004-2011 OpenCFD Ltd.
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Application
    buoyantPimpleFoamWithSource

Description
    Transient solver for buoyant, turbulent flow of compressible fluids for
    ventilation and heat-transfer.

    Turbulence is modelled using a run-time selectable compressible RAS or
    LES model.

\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "basicRhoThermo.H"
#include "turbulenceModel.H"
#include "fixedGradientFvPatchFields.H"
#include "pimpleControl.H"

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

int main(int argc, char *argv[])
{
    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H"
    #include "readGravitationalAcceleration.H"
    #include "createFields.H"
    #include "initContinuityErrs.H"
    #include "readTimeControls.H"
    #include "compressibleCourantNo.H"
    #include "setInitialDeltaT.H"
```

```
55
56      pimpleControl pimple(mesh);
57
58      // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
59
60      Info<< "\nStarting time loop\n" << endl;
61
62      while (runTime.run())
63      {
64          #include "readTimeControls.H"
65          #include "compressibleCourantNo.H"
66          #include "setDeltaT.H"
67
68          runTime++;
69
70          Info<< "Time = " << runTime.timeName() << nl << endl;
71
72          #include "rhoEqn.H"
73
74          // --- Pressure-velocity PIMPLE corrector loop
75          for (pimple.start(); pimple.loop(); pimple++)
76          {
77              if (pimple.nOuterCorr() != 1)
78              {
79                  p_rgh.storePrevIter();
80              }
81
82              #include "UEqn.H"
83              #include "hEqn.H"
84
85              // --- PISO loop
86              for (int corr=0; corr<pimple.nCorr(); corr++)
87              {
88                  #include "pEqn.H"
89              }
90
91              if (pimple.turbCorr())
92              {
93                  turbulence->correct();
94              }
95          }
96
97          rho = thermo.rho();
98
99          runTime.write();
100
101         Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
102             << "  ClockTime = " << runTime.elapsedClockTime() << " s"
103             << nl << endl;
104     }
105
106     Info<< "End\n" << endl;
107
108     return 0;
109 }
110
111
112 // ************************************************************************* //
```

C.2 hEqn.H

```
{
    fvScalarMatrix hEqn
    (
        fvm::ddt(rho, h)
      + fvm::div(phi, h)
      - fvm::laplacian(turbulence->alphaEff(), h)
     ==
        DpDt
      + q
    );

    hEqn.relax();
    hEqn.solve();

    thermo.correct();
}
```

C.3 createFields.H

```
    Info<< "Reading thermophysical properties\n" << endl;

    autoPtr<basicRhoThermo> pThermo
    (
        basicRhoThermo::New(mesh)
    );
    basicRhoThermo& thermo = pThermo();

    volScalarField rho
    (
        IOobject
        (
            "rho",
            runTime.timeName(),
            mesh,
            IOobject::READ_IF_PRESENT,
            IOobject::AUTO_WRITE
        ),
        thermo.rho()
    );

    volScalarField& p = thermo.p();
    volScalarField& h = thermo.h();
    const volScalarField& psi = thermo.psi();


    Info<< "Reading field U\n" << endl;
    volVectorField U
    (
        IOobject
        (
            "U",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
```

```
39      );
40
41      #include "compressibleCreatePhi.H"
42
43
44      Info<< "Creating turbulence model\n" << endl;
45      autoPtr<compressible::turbulenceModel> turbulence
46      (
47          compressible::turbulenceModel::New
48          (
49              rho,
50              U,
51              phi,
52              thermo
53          )
54      );
55
56      Info<< "Calculating field g.h\n" << endl;
57      volScalarField gh("gh", g & mesh.C());
58      surfaceScalarField ghf("ghf", g & mesh.Cf());
59
60      Info<< "Reading field p_rgh\n" << endl;
61      volScalarField p_rgh
62      (
63          IOobject
64          (
65              "p_rgh",
66              runTime.timeName(),
67              mesh,
68              IOobject::MUST_READ,
69              IOobject::AUTO_WRITE
70          ),
71          mesh
72      );
73
74      // Force p_rgh to be consistent with p
75      p_rgh = p - rho*gh;
76
77      Info<< "Creating field DpDt\n" << endl;
78      volScalarField DpDt
79      (
80          "DpDt",
81          fvc::DDt(surfaceScalarField("phiU", phi/fvc::interpolate(rho)), p)
82      );
83
84      volScalarField q
85      (
86          IOobject
87          (
88              "q",
89              runTime.timeName(),
90              mesh,
91              IOobject::MUST_READ,
92              IOobject::AUTO_WRITE
93          ),
94          mesh
95      );
```

C.4 setFieldsDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue q 0
);

regions
(
    cylinderToCell
    {
    p1 (0 0 0);
    p2 (0 0 0.2);
    radius 0.3;
        fieldValues
        (
            volScalarFieldValue q 176838
        );
    }
);

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

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences
Department Maschinenbau und Produktion

Formblatt **Erklärung zur selbstständigen Bearbeitung einer Masterthesis**

Zur Erläuterung des Zweckes dieses Blattes:

§ 16 Abs. 5 der APSOTIBM lautet:

„Zusammen mit der Thesis ist eine schriftliche Erklärung abzugeben aus der hervorgeht, dass die Arbeit bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit (§18 Absatz 1) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Dieses Blatt mit der folgenden Erklärung ist nach Fertigstellung der Arbeit durch jede/n Kandidat/en/in auszufüllen und jeweils mit Originalunterschrift als letztes Blatt des als Prüfungsexemplar der Masterthesis gekennzeichneten Exemplars einzubinden.

Eine unrichtig abgegebene Erklärung kann - auch nachträglich - zur Ungültigkeit des Masterabschlusses führen.

Erklärung

Hiermit versichere ich,

Name: Dahncke

Vorname: Mike

dass ich die vorliegende Masterthesis – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema

Untersuchung der Möglichkeiten zum Einsatz von Open Source CFD Software für Strömungssimulationen im Bereich der Gebäudetechnik

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der Masterthesis ist erfolgt durch

Hamburg		
Ort	Datum	Unterschrift im Original