

FancyFOAM
**Der Versuch einer umfassenden
Dokumentation von OPENFOAM**

Aljoscha N. Sander

27. Januar 2016

Inhaltsverzeichnis

1	Linux	1
1.1	Was, Warum und Wer ist Linux	1
1.2	Die wichtigsten Konsolenbefehle	2
2	Strömungsmechanik	3
3	Turbulenz	4
3.1	RANS	4
3.2	LES/DES	4
4	OpenFOAM	5
4.1	Installation	5
4.2	Struktur	5
4.2.1	constant	5
4.2.2	system	7
4.2.3	0	8
5	Gittergenerierung	9
5.1	blockMesh	9
5.2	snappyHexMesh	11
5.3	proprietäre Gittergeneratoren	16
6	Diskretisierungen und Gleichungslöser	18
6.1	Diskretisierungsverfahren: fvSchemes	18
6.1.1	Zeitliche Diskretisierung: ddtSchemes	18
6.1.2	Konvektiver Term: divSchemes	18
6.1.3	Gradienten: gradSchemes	18
6.1.4	Diffusiver Term: laplacianSchemes	19
6.1.5	Interpolation: interpolationSchemes	19
6.2	Gleichungslöser: fvSolution	19
6.2.1	Symmetrische Matrizen	19
7	Initial und Randbedingungen	20
8	Pre-Processing	22

9 Stationäre Rechnungen	23
9.1 SIMPLE-Algorithmus: SIMPLEFOAM	23
10 Instationäre Rechnungen	24
10.1 PIMPLE-Algorithmus: PIMPLEFOAM	24
10.2 PISO-Algorithmus: PISOFOAM	24
10.3 ICO-Algorithmus: ICOFOAM	24
11 Bewegte Gitter	25
11.1 MRF	26
11.2 AMI	27
11.3 RBF	28
11.4 dynamicTopoFvMesh	29
11.4.1 Mesquite	29
11.5 sixDoFRigidBodyMotionSolver	31
12 Mehrphasenrechnungen	32
13 Wärme	33
14 Post-Processing	34
15 Paraview	35
16 Nachschlagewerke und Literatur	36
Anhang	I
A constant	II
B system	IV
C 0	IX
D Formeln zu RANS-Randbedingungen	XI
E Checkliste zu <i>snappyHexMesh</i>	XIII
F Checkliste zu <i>pimpleFoam</i>	XIV

Abbildungsverzeichnis

4.1 Struktur einer OpenFOAM Simulation	6
5.1 blockMesh Gitter	9
5.2 Geometriedefinition snappyHexMesh	11
5.3 Zellverfeinerung snappyHexMesh	14
5.4 Ungeglättetes, verfeinertes Gitter in snappyHexMesh	14
5.5 geglättetes Gitter mit snappyHexMesh	15
5.6 Prismenschichten snappyHexMesh	17

Tabellenverzeichnis

1.1	Konsolenbefehle	2
4.1	Terme und ihre Bezeichnungen in OpenFOAM	8
7.1	Terme und ihre Bezeichnungen in OpenFOAM	20
11.1	Dynamische Gittertypen	26
11.2	Qualitätsmerkmale für <i>Mesquite</i>	29
11.4	Optimierungsalgorithmen	30
11.5	Constraints	31
11.6	Restraints	31
11.3	Optimierungsalgorithmen	31
D.1	Feste Variablen	XI
D.2	Empirische Formeln zur Abschätzung der turbulenten Viskosität $\tilde{\nu}$	XII
D.3	Empirische Formeln zur Abschätzung der turbulenten kinetischen Energie k	XII
D.4	Empirische Formeln zur Abschätzung der Dissipationsrate ϵ	XII
D.5	Empirische Formeln zur Abschätzung der spezifischen Dissipationsrate ω	XII

Quelltextverzeichnis

5.1	Geometriedefinition innerhalb von <i>snappyHexMeshDict</i>	12
5.2	<i>castellatedMeshControls</i> innerhalb von <i>snappyHexMeshDict</i>	13
5.3	<i>snapControls</i> innerhalb von <i>snappyHexMeshDict</i>	15
5.4	<i>addLayerControls</i> innerhalb von <i>snappyHexMesh</i>	15
A.1	Datei <i>constant/polyMesh/blockMesh</i>	II
B.1	Datei <i>system/fvOptions</i>	IV
B.2	Datei <i>system/fvSchemes</i>	V
B.3	Datei <i>system/fvSolution</i>	VI
B.4	Datei <i>system/controlDict</i>	VIII
C.1	Datei <i>0/U</i>	IX
C.2	Datei <i>0/p</i>	X

Abkuerzungen

IMO	International Maritime Organization
MDOF	Multiple degree-of-freedom
SDOF	Single degree-of-freedom
SWATH	Small Waterplane Area Twin Hull

Symbolverzeichnis

Symbol	Einheit	Beschreibung
L	m	Laenge
A	mm ²	Flaeche

Dokumentation der wichtigsten Eigenschaften von OpenFOAM.

1 Linux

OpenFOAM ist grundsätzlich auf verschiedenen Betriebssystemen lauffähig. Innerhalb dieses Dokuments wird jedoch davon ausgegangen, dass OpenFOAM unter einem modernen UNIX Betriebssystem (streng genommen sollte dadurch MAC OS X davon ausgenommen werden, da MAC grundlegende POSIX-Richtlinien links liegen lässt und (mal wieder) den armen Nutzern vorgaukelt es würde alles viel besser machen. Was in diesem Fall nicht stimmt, da es durch die POSIX-Violation erheblichen Aufwand erfordert OpenFOAM unter MAC zum laufen zu bekommen. Sollte der Leser aber unter MAC OS X arbeiten, gehe ich davon aus, dass er entweder eine Virtuelle Maschine mit einem anständigen Betriebssystem zur Verfügung hat oder so gut ist, dass er OpenFOAM gepatcht und von Hand kompiliert hat. Sollte letzteres der Fall sein, darf der Leser getrost das folgende Kapitel überspringen) wie bspw. Linux läuft.

1.1 Was, Warum und Wer ist Linux

Linux ist streng genommen zunächst mal eine Software, welche in der Lage ist mit Hardware zu kommunizieren und Kommunikationskanäle zwischen den verschiedenen Hardwarekomponenten eines Computers zur Verfügung zu stellt. Linux stellt somit den elementaren Teil eines Betriebssystems dar (für Leser die nicht wissen, was was ein Betriebssystem ist; Wikipedia ist treuer als so manches Haustier und geduldiger als so manche Ehefrau... ;-)). In Kombination mit viel anderer Software (welche gemeinhin als Distribution bezeichnet wird) stellt es ein freies ¹ Betriebssystem zur Verfügung. Populäre Distributionen sind u.A. Ubuntu, Debian, OpenSUSE, Fedora, Arch Linux, Cent OS, uvm. Diesen Distributionen ist gemein, dass Sie alle im Kern auf Linux basieren und dem POSIX-Standard genügen.

Die vorhandenen Linuxdistributionen bietet den unschlagbaren Vorteil, dass es von vielen Nerds, verteilt über den ganzen Globus ² verteilt entwickelt wird. Wobei die verschiedenen Distributionen verschiedene Anwender im Fokus haben. Allen gemein ist es jedoch, dass es äußerst einfach ist neue Software auf Linux zum laufen zu bringen. Was wiederum der freien Zugänglichkeit der Software geschuldet ist. Anyways, es gibt viele, viele weitere Gründe warum Linux wunderbar und fabelhaft ist, sicherlich ebenbürtig viele Gründe die gegen das

¹frei nicht immer im Sinne von kostenlos. Frei in dem Sinn, dass der Quellcode frei Verfügbar ist und jeder diesen Quellcode nutzen darf (durchaus auch zur kommerziellen Nutzung).

²Fun Fact: und darüber hinaus; die Laptops der ISS-Besatzung laufen auf Debian; die Rechner auf Arktisforschungsstationen und Tiefseerobotern ebenfalls. Linux IST überall.

Betriebssystem sprechen ³.

Der Grund warum dieses Dokument mit einem Kapitel über Linux beginnt ist einfach: OpenFOAM ist genau genommen kein einzelnes Programm, sondern eine gigantische Bibliothek (auch als Framework bezeichnet) an Software, deren Gemeinsamkeit darin besteht, dass sie alle über eine Konsole bedient werden. Linux stellt eine POSIX-konforme Konsolenumgebung zur Verfügung, welche, unter den richtigen Umständen und mit dem richtigen Wissen genutzt, äußerst effizientes Arbeiten erlaubt. Dazu folgen im nächsten Kapitel die wichtigsten Befehle.

1.2 Die wichtigsten Konsolenbefehle

Befehl	Aktion
man BEFEHL	Der wichtigste Befehl überhaupt; öffnet ein Handbuch mit Erklärungen und Beispielen zum Befehl BEFEHL
cd PFAD	change directory. Wechselt in das angegebene Verzeichnis
cp [-rv] QUELLE ZIEL	kopiert QUELLE nach ZIEL. vorsicht, überschreibt auch Dateien. Das kopieren von Verzeichnissen ist nur mit der Option -r möglich
rm [-rfv] DATEI VERZEICHNIS1 ...	löscht die/das DATEI1/VERZEICHNIS1 ^{4 5}
cat DATEI	gibt den gesamten Inhalt von DATEI auf der Konsole aus.
less DATEI	öffnet die Datei DATEI und ermöglicht ein skrollen, springen und durchsuchen der Datei mittels Tastenkombinationen.
tail [-fn ZAHL] DATEI	gibt den letzten Teil einer Datei aus oder gibt kontinuierlich aus was in die Datei DATEI geschrieben wird.
BEFEHL1 BEFEHL2	das Sonderzeichen (auch als pipe bezeichnet) verknüpft zwei Befehl miteinander, so dass der output des ersten Befehls als input für den zweiten Befehl genutzt werden kann.
grep [-Rinl] SCHLAGWORT VERZEICHNIS/ DATEI	durchsucht die Datei DATEI oder alle Dateien im Verzeichnis VERZEICHNIS nach SCHLAGWORT.
find VERZEICHNIS -iname NAME -type TYP -exec BEFEHL	Durchsucht VERZEICHNIS nach Dateien des Typs TYP mit dem namen NAME und führt (wenn so spezifiziert) den befehl BEFEHL aus.

Tabelle 1.1: Konsolenbefehle

³u.A.: <http://ubuntuforums.org/showthread.php?t=1852199> , <http://linuxhaters.blogspot.de/>

2 Strömungsmechanik

- Impulsgleichung
- Energiegleichung
- Masseerhaltung
- Impulsgleichung für inkompressible, newtonsche Fluide (N-S Gleichung)
- Impulsgleichung für kompressible, newtonsche Fluide

Wir gehen zunächst von einem inkompressiblen, newtonschen Medium für das die Kontinuumshypothese gilt aus. Dadurch wird die Impulsgleichung zur Navier-Stokes-Gleichung (Gleichung 2.1):

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{\frac{p}{\rho} + gk}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j^2} \quad (2.1)$$

Durch die Inkompressibilität vereinfacht sich die Kontinuumsgleichung zu Gleichung 2.2:

$$\frac{\partial u_i}{\partial x_i} = 0 \quad (2.2)$$

Der erste Term von Gleichung 2.1, $\frac{\partial u_i}{\partial t}$ wird als instationärer Term bezeichnet, er repräsentiert den Einfluss der Änderung des Impulses über die Zeit. Der zweite Term $u_j \frac{\partial u_i}{\partial x_j}$, auch als konvektiver Term bezeichnet, beschreibt die Impulstransport aufgrund von Geschwindigkeitsänderungen. Dieser Term ist nichtlinear. Der dritte Term $-\frac{\frac{p}{\rho} + gk}{\partial x_i}$ ist der Druckgradient. In OpenFOAM wird der Druckgradient mit weiteren äußeren Kräften, wie z.B. der Gravitation (bei mehrphasigen Systemen) oder elektromagnetischen Kräften zusammengefasst. Der letzte Term $\nu \frac{\partial^2 u_i}{\partial x_j^2}$, auch als diffusiver Term bezeichnet.

3 Turbulenz

3.1 RANS

3.2 LES/DES

4 OpenFOAM

OpenFOAM ist ein in C++ geschriebenes Framework zum lösen von partiellen Differentialgleichungen. Hauptanwendungsbereich ist die numerische Strömungssimulation, aber auch Finanzmathematik und mechanische Analysen sind möglich.

4.1 Installation

Grundsätzlich stehen zwei verschiedene Installationswege zur Verfügung: Das Übersetzen des Quellcodes in Maschinencode (als Kompilation bezeichnet) oder die Installation mit Hilfe eines Paketmanagers.

installationsa

4.2 Struktur

Zur Simulation einer Problemstellung wird zunächst ein eigenes Verzeichnis mit beliebigem Namen benötigt. Die Struktur innerhalb dieses ist jedoch fest vorgegeben. Abbildung 4.1 zeigt die Struktur innerhalb des Simulationsordners.

4.2.1 constant

Der *constant* Ordner beinhaltet im Subordner *polyMesh* das von numerische Gitter im OpenFOAM Format. Handelt es sich bei dem Gitter um ein mit *blockMesh* generiertes blockstrukturiertes Gitter, so liegt dort auch die Datei *blockMeshDict*, Siehe Listing A.1, innerhalb derer die Geometrie und die Randbedingungen definiert werden. Des Weiteren beinhaltet *constant* unter Umständen das Unterverzeichnis *triSurface*, innerhalb dessen Oberflächengitter zur Generierung komplexer Gitter mit den Gittergeneratoren *snappyHexMesh*, *foamyQuadMesh* und *foamyHexMesh* abgelegt sind (Siehe Kapitel Gittergenerierung).

Zwangsweise vorhanden sein muss die Datei *transportProperties*. Diese Datei legt die kinematische Viskosität sowie die Art von Fluid (Newtonsche, Bingham'sche, etc.) fest.

In den Dateien *RASProperties*, *LESProperties* und *turbulenceProperties* werden etwaige Turbulenzmodelle definiert.

Eine gesonderte Rolle spielte die Datei *dynamicMeshDict*; Soll innerhalb der Simulation mit bewegten Gittern gearbeitet werden, so wird das Verfahren, sowie die Bewegung hier definiert.

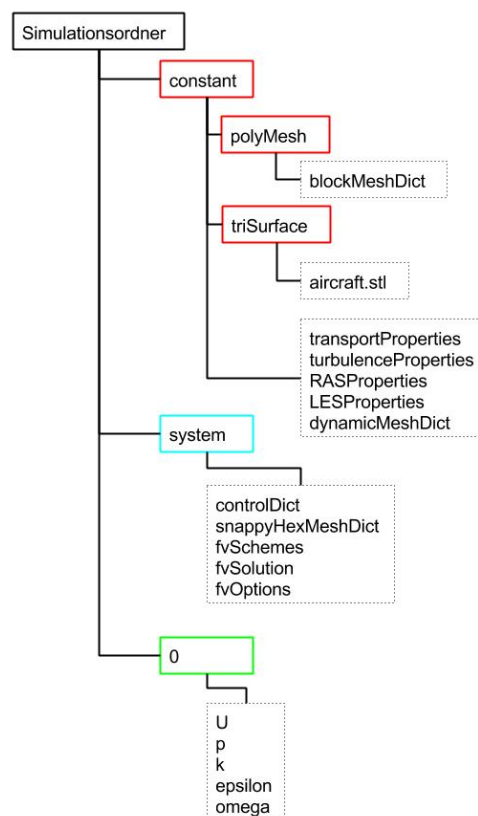


Abbildung 4.1: Die Ordnerstruktur innerhalb einer OpenFOAM Simulation. Der system ordner beinhaltet die Einstellungen der Lösungsprogramme, der 0 Ordner beinhaltet die Initialbedingungen und der constant Ordner das Gitter und die Randbedingungen

4.2.2 system

Mit den Dateien innerhalb dieses Ordners wird die Simulation gesteuert.

controlDict Mit dieser Datei wird die Simulation gesteuert; nachfolgend sind die wichtigsten Parameter und ihre Optionen aufgeführt.

application : legt den zu verwendenden Gleichungslöser fest; nicht zwingend notwendig, nur in Kombination mit den von OpenFOAM bereitgestellten RunFunctions benötigt.

startFrom : legt fest ob die Simulation von einem festgelegten Zeitpunkt aus startet oder vom letzten verfügbaren Zeitschritt. Soll ein Zeitschritt festgelegt werden, muss als Wert 'startTime' eingetragen werden. Ansonsten 'latestTime'.

startTime : wird nur benötigt falls ein fester Startzeitpunkt festgelegt werden soll.

endTime : definiert das Ende der Simulation.

deltaT : bezeichnet die Zeitschrittweite und bei instationären Simulationen damit die Courantzahl. Bei stationären Zeitschritten ist die Zeitschrittweite überflüssig, auf 1 gesetzt zeigt sie somit die Anzahl an Iterationen an.

writeControl : steuert das Speichern. Soll der momentan gerechnete Zeitschritt noch beendet werden und danach die Simulation abgeschlossen, muss als Wert 'writeNow' eingetragen werden. Andere Optionen sind timeStep und XXX .

writeInterval : definiert die Speicherintervalle.

writeFormat : definiert ob beim Speichern die Daten in binärem Format oder in menschlich-lesbarem (ascii) gespeichert werden sollen. Binary reduziert den benötigten Speicherplatz auf etwa 20 % des menschlich-lesbaren.

runTimeModifiable : lässt zu ob Textdateien zur Kontrolle der Simulation geändert und damit die Simulation zur Laufzeit modifiziert werden darf.

functions : Soll zur Laufzeit bereits PostProcessing betrieben werden (bspw: grafische Schnitte, Strömungsbeiwerte, etc), müssen diese hier definiert werden

anderen
Parameter
raus-
suchen

snappyHexMeshDict Diese Datei wird als Steuerdatei für den Gittergenerator *snappyHexMesh* benötigt. Details dazu finden sich im Kapitel Gittergenerierung.

fvSchemes Diese Datei beinhaltet die zu verwendenden Diskretisierungs und Interpolationsverfahren. Grundsätzlich gibt es in OpenFOAM viele verschiedene Verfahren. Erforderlich für das Starten einer Simulation ist zunächst, dass alle in den vorhandenen Gleichungen auftretenden Terme durch die Verfahren abgedeckt sind. Tabelle 4.1 zeigt die Terme und ihre Bezeichnungen in OpenFOAM. Das Kapitel Diskretisierungen behandelt die vorhandenen Verfahren und ihre Effekte.

Term	Bezeichnung in OpenFOAM
Instationärer Term $\frac{\partial u_i}{\partial t}$	ddtSchemes
Konvektiver Term $u_j \frac{\partial u_i}{\partial x_j}$	divSchemes
(Druck)Gradient $-\frac{p}{\rho} + gk$	gradSchemes
Diffusiver Term $\nu \frac{\partial^2 u_i}{\partial x_j^2}$	laplacianSchemes
Interpolationsverfahren	interpolationSchemes

Tabelle 4.1: Terme und ihre Bezeichnungen in OpenFOAM

fvSolution Innerhalb dieser Datei werden die Algorithmen zur Gleichungslösung eingestellt. Des Weiteren werden Relaxationsfaktoren und Konvergenzkriterien hier definiert.

fvOptions Diese Datei beinhaltet die Beschreibung bestimmter Patches, beispielsweise zur Simulation von porösen Medien oder zur Simulation mit anderen Bezugssystemen.

4.2.3 0

Innerhalb des '0' Ordners werden die Initialbedingungen festgelegt; alle zur Simulation benötigten Größen erhalten ihre eigene Datei, innerhalb derer die Initialwerte festgelegt werden. Für Details: Kapitel Initial und Randbedingungen.

5 Gittergenerierung

5.1 blockMesh

blockMesh ist ein einfacher Gittergenerator für blockstrukturierte (Hexaeder) Gitter. *blockMesh* benötigt als input die datei *constant/polyMesh/blockMeshDict* in welcher die Geometrie und Gittergenerierung festgelegt werden. Abbildung 5.1 zeigt ein solches Gitter. Aufgerufen wird *blockMesh* normalerweise ohne Parameter im Wurzelordner der Simulation.

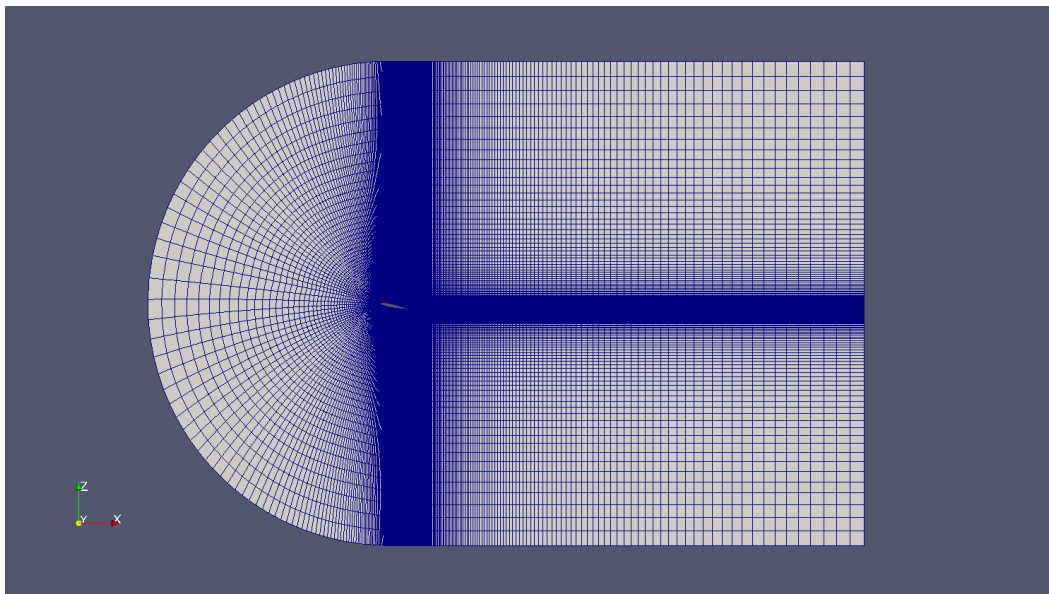


Abbildung 5.1: Ein mit *blockMesh* generiertes Gitter eines NACA 0012 Flügelprofils. Deutlich zu erkennen ist das Grading im Gitter.

Listing A.1 zeigt das zur Generierung des in Abbildung 5.1 genutzte *blockMeshDict*.

Grundsätzlich ist die Gittergenerierung relativ einfach, einige Punkte sind jedoch zu beachten:

- *scaleToMeters*: Der Parameter *scaleToMeters* (Zeile 18 in Listing A.1) *blockMeshDicts* ermöglicht es die zu erstellenden Gitter automatisch zu skalieren. Dieser Parameter kann schnell zu Fehlern in den Rechnungen führen, da die Größe des Gitters nicht sofort erkennbar ist und dadurch die zu simulierenden Größen, welche von der Gitterlänge (bspw. Geschwindigkeit, kinematische Viskosität) schnell um Größenordnungen abweichen können.

- *boundary*: Die Randbedingungen der Gitter werden bereits im *blockMeshDict* festgelegt (in Listing A.1 ab Zeile 116). Dabei ist wichtig, dass die richtige Option gewählt wird. Es stehen in OpenFOAM verschiedene Randbedingungen zur Verfügung (Siehe dazu Kapitel OpenFOAM). Ist noch nicht sicher, um welche Art Randbedingung es sich handelt, ist *type patch* die beste Wahl.
- *Uhrzeigersinn*: Sollte es beim Generieren des Gitters zu Schwierigkeiten kommen, so lohnt es sich immer die Reihenfolge der abgezählten Knoten (*vertices*) zu überprüfen. Sind diese nicht in der richtigen Reihenfolge kommt es zu Verdrehungen innerhalb des Gitters.

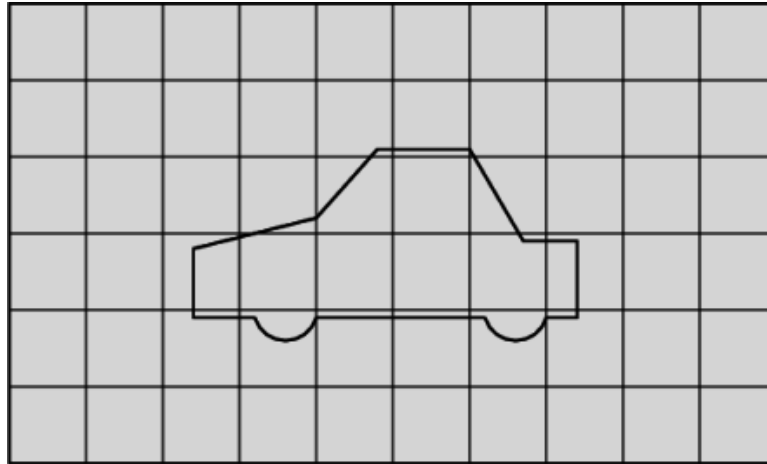


Abbildung 5.2: Geometrie innerhalb eines blockstrukturierten Gitters für die Gittergeneration mit *snappyHexMesh*. Das blockstrukturierte Gitter muss die Oberfläche überall dort umschließen, wo es zu einer Interaktion zwischen Fluid und Geometrie kommen soll.

5.2 *snappyHexMesh*

snappyHexMesh generiert aus einem blockstrukturierten Gitter (*blockMesh*) ein hybrides Gitter. Mit diesem Tool können bequem komplexe Geometrien in Gitter umgesetzt werden. Abbildung 5.2 zeigt den schematischen Aufbau. Dabei wird von der komplexen Geometrie ein Oberflächengitter (In Abbildung 5.2 als STL surface bezeichnet) benötigt, welches von dem blockstrukturierten Gitter umgeben ist. Das Tool unterstützt viele Oberflächengitter (u.a.: STLs). Als zusätzliche Fähigkeit können prismatische Oberflächenzelle generiert werden, welche (Druck)Gradienten besser abbilden können. Das Tool ist voll parallelisiert, diese Möglichkeit sollte auch genutzt werden, da der Speicherverbrauch hoch ist (pro 1Mio Zellen etwa 1 GB RAM)

Der Aufruf von *snappyHexMesh* erfolgt aus dem Wurzelordner der Simulation. Als nützliche Parameter seien hier die folgenden aufgezählt:

- -parallel soll *snappyHexMesh* parallel genutzt werden, so ist dieser Parameter unerlässlich.
- -overwrite *snappyHexMesh* schreibt alle drei Bearbeitungsschritte in eigene Zeitordner; ist dies nicht gewünscht (da meist unpraktikabel), hilft dieser Parameter. Das Gitter wird in den constant Ordner geschrieben.

snappyHexMesh wird durch eine Inputdatei (*system/snappyHexMeshDict*) gesteuert. Die Gittergenerierung mit *snappyHexMesh* lässt sich in drei Schritte unterteilen.

- Verfeinern (*castellatedMesh*)

- Glätten (*snap*)
- Oberflächenzellen hinzufügen (*addLayers*)

Das *snappyHexMeshDict* ist diesen Schritten entsprechend gegliedert. Zunächst wird die Geometrie festgelegt (Listing 5.1):

Listing 5.1: Geometriedefinition innerhalb von *snappyHexMeshDict*

```

1 geometry
2 {
3     oberflaeche.stl
4     {
5         type triSurfaceMesh;
6         name oberflaeche;
7     }
8
9     verfeinerung
10    {
11        type searchableBox;
12        min (-1.0 -0.7 0.0);
13        max ( 8.0  0.7 2.5);
14    }
15 };

```

Dabei sind in *snappyHexMesh* mehrere Typen von Geometrien erlaubt:

- *closedTriSurfaceMesh*: benötigt als Input eine geschlossene Oberflächengeometrie
- *distributedTriSurfaceMesh*: _____
- *searchableBox*: generiert eine Box über die längste Diagonale fest. Inputparameter sind $\min(x\ y\ z)$ und $\max(x\ y\ z)$.
- *searchableCylinder*: generiert einen Zylinder. Inputparameter sind $\min(x\ y\ z)$, $\max(x\ y\ z)$ und *radius* *r*.
- *searchableDisk* _____
- *searchablePlane* _____
- *searchablePlate* _____
- *searchableSphere*: generiert eine Kugel. Inputparameter sind *radius* *r* und *centre* (*x y z*)
- *searchableSurfaceCollection* _____
- *searchableSurfaceWithGaps* _____
- *triSurfaceMesh*: benötigt als Input eine Oberflächengeometrie.

raus finden was das ist

testen

testen

testen

dafuq?

dafuq?

Danach folgen die Einstellung für das Verfeinern der Zellen, dem ersten Schritt innerhalb von *snappyHexMesh* (*castellatedMeshControls*)

Listing 5.2: *castellatedMeshControls* innerhalb von *snappyHexMeshDict*

```

1  castellatedMeshControls
2  {
3      // maximal zulässige Anzahl von Zellen pro Prozessor
4      // ACHTUNG: bricht Verfeinerung ab sobald Anzahl erreicht ist
5      maxLocalCells 100000;
6
7      // maximal zulässige Anzahl von Zellen über alle Prozessoren summiert
8      // ACHTUNG: bricht Verfeinerung ab sobald Anzahl erreicht ist
9      maxGlobalCells 2000000;
10
11     // sollten weniger als diese Anzahl an Zellen verfeinert werden, wird abgebrochen
12     minRefinementCells 10;
13
14     // maximal zulässiges Ungleichgewicht an Zellanzahlen zwischen Prozessoren. Wird dieses Niveau überschritten, werden die Zellen neu
15     // verteilt
16     maxLoadUnbalance 0.10;
17
18     // Anzahl an Zellschichten zwischen den Verfeinerungen
19     nCellsBetweenLevels 3;
20
21     // wird das zusätzliche Tool surfaceFeatureExtract genutzt wird die davon generierte Geometrie hier eingebunden und das
22     // Verfeinerungslevel festgelegt
23     features
24     (
25         {
26             file "motorBike.eMesh";
27             level 6;
28         }
29     );
30
31     // die Oberflächen zu denen hin verfeinert werden soll, werden hier festgelegt. Dabei kann es sich sowohl um die vorher eingeleiten
32     // Oberflächengeometrien aus Dateien als auch um intern festgelegte Geometrien, bsp searchableSphere handeln.
33     refinementSurfaces
34     {
35         oberflaeche
36         {
37             level (min max);
38         }
39     }
40
41     // Dieser Winkel zwischen zwei angrenzenden Zellen entscheidet darüber, ab wann das nächste Verfeinerungslevel an der Oberfläche
42     // eingesetzt werden soll. Start bei min.
43     resolveFeatureAngle 45;
44
45     // Soll ein ganzer Zellbereich innerhalb eines bestimmten Raumes auf ein bestimmtes Maß verfeinert werden, wird dies hier festgelegt
46     refinementRegions
47     {
48         refinementBox
49         {
50             mode inside;
51             levels ((1E15 4));
52         }
53     }
54
55     // diese Koordinate entscheidet darüber welcher Teil des Gitters behalten wird; der innerhalb der Geometrie oder der ausserhalb.
56     locationInMesh (3.0001 3.0001 0.43);
57
58     \todo{was ist das?}
59     allowFreeStandingZoneFaces true;
60 }

```

Nach dem dieser Schritt abgeschlossen ist, ist das Gitter sowohl innerhalb als auch ausserhalb der Geometrie zur Geometrie hin verfeinert (siehe Abbildung 5.3).

Mit Hilfe des Parameters `locationInMesh (x y z)` wird nun entschieden welcher Teil des Gitter erhalten wird und welcher Teil entfernt wird. Dies gilt ausdrücklich nur wenn es sich bei der Geometrie um eine Wand handelt. Es ist grundsätzlich auch möglich die Verfeinerungen in beide Richtungen beizubehalten.

Nach dem Ausschneiden sieht das Gitter wie folgt aus (Abbildung 5.4):

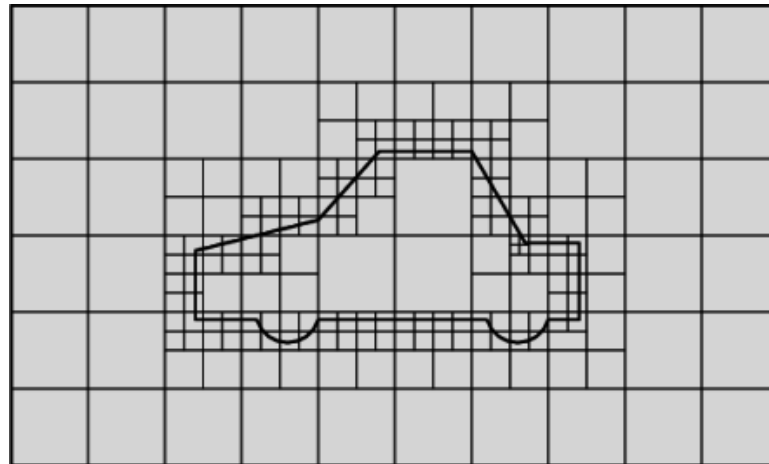


Abbildung 5.3: Die Zellverfeinerungen in der Umgebung der Oberflächegeometrie.

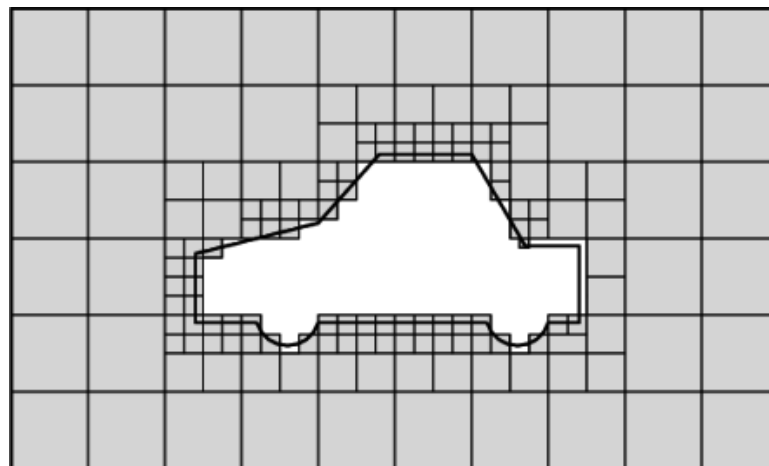


Abbildung 5.4: Das ungeglättete, verfeinerte Gitter in snappyHexMesh.

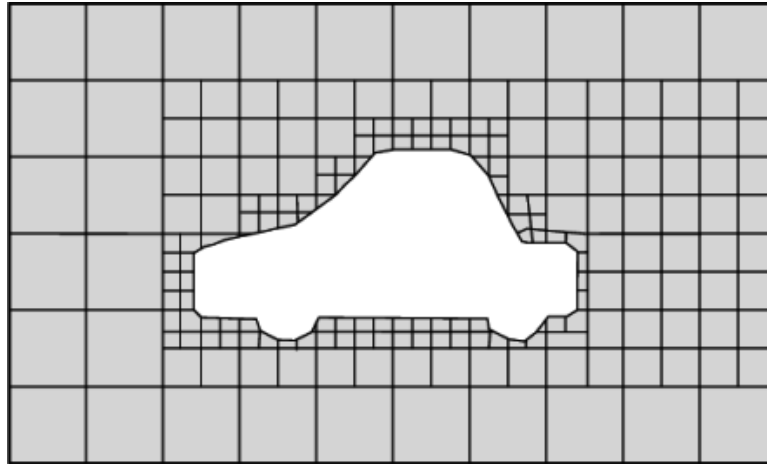


Abbildung 5.5: Das an der Oberfläche geglättete Gitter.

Auf die Einstellungen für die Verfeinerungen folgen die Einstellungen für das Glätten der Oberfläche, der zweite Schritt innerhalb von *snappyHexMesh* (*snapControls*, Listing 5.3)

Listing 5.3: *snapControls* innerhalb von *snappyHexMeshDict*

```

1 snapControls
2 {
3     // Dieser Parameter legt fest wie viele Iterationen (Wiederholungen) snappyHexMesh darauf verwenden darf die Oberfläche zu glätten.
4     nSmoothPatch 3;
5
6     // Dieser Faktor legt die Auflösungstoleranz fest; Die tatsächliche Toleranz entspricht der lokalen Kantenlänge multipliziert mit
7     // diesem Faktor.
8     tolerance 2.0;
9
10    // Lösungsiterationen des Gleichungslösers, welcher die Gitterpunkte verschiebt
11    nSolveIter 30;
12
13    // Relaxationsiterationen für den Gleichungslöser
14    nRelaxIter 5;
15
16    // Die folgenden Parameter gelten nur für den Fall das mit surfaceFeatureExtract gearbeitet wurde
17    nFeatureSnapIter 10;
18
19    implicitFeatureSnap false;
20    explicitFeatureSnap true;
21
22    multiRegionFeatureSnap false;
23 }
```

Nach erfolgreicher Beendigung dieses Schrittes hat das Gitter folgende Form (Abbildung 5.5):

Im letzten Schritt werden die Prismenschichten an den festgelegten Oberflächen generiert. Hierzu werden die folgenden Einstellungen benötigt (Listing 5.4):

Listing 5.4: *addLayerControls* innerhalb von *snappyHexMesh*

```

1 addLayersControls
2 {
3     // Hier wird angegeben, ob es sich um zum Grundgitter relative Größenangaben (true) oder um absolute Größenangaben (false) handelt.
4     relativeSizes true;
5
6     layers
```



```
7 {
8     geometriename //nicht name der datei, sondern der name des patches
9     {
10         nSurfaceLayers 3;
11     }
12 }
13
14 // relativer Wachstumsfaktor
15 expansionRatio 1.0;
16
17 // relative Dicke der letzten prismatischen Zellschicht zur unveränderten Grundgitterzelle
18 finalLayerThickness 0.3;
19
20 // relative Mindestdicke
21 minThickness 0.1;
22
23 // sollten bestimmte Bereich nicht mit Zellen bedeckt werden können, kann die Anzahl der nGrow Iterationen helfen Bedeckung zu
    erreichen.
24 nGrow 0;
25
26 // Ab welchem Winkel zwischen zwei Zellen sollen keine Zellen mehr generiert werden
27 featureAngle 60;
28
29 // finger weg \todo{dafuq tis shit?}
30 slipFeatureAngle 30;
31
32 // relaxationsiterationen
33 nRelaxIter 3;
34
35 // iterartionen zum Glätten der Flächennormalen
36 nSmoothSurfaceNormals 1;
37
38 // Glättungsiterationen des internen Gitters
39 nSmoothNormals 3;
40
41 // iterationen zum glätten der Zellschichtdicken
42 nSmoothThickness 10;
43
44 // Ab wann das Glätten der Dicke unterbrochen werden soll; entspricht dem Verhältnis von Zelldicke zu Fläche
45 maxFaceThicknessRatio 0.5;
46
47 // keine ahnung
48 maxThicknessToMedialRatio 0.3;
49
50 // ebenfalls keine ahnung
51 minMedianAxisAngle 90;
52
53 // bis hier her hat eh niemand gelesen
54 nBufferCellsNoExtrude 0;
55
56 // Maximale Iteration von Iterationen
57 nLayerIter 50;
58 }
```

5.3 proprietäre Gittergeneratoren

Es existieren eine Reihe freiere und proprietärer Gittergeneratoren, für welche OpenFOAM Konvertierungstools zur Verfügung stellt.

liste:

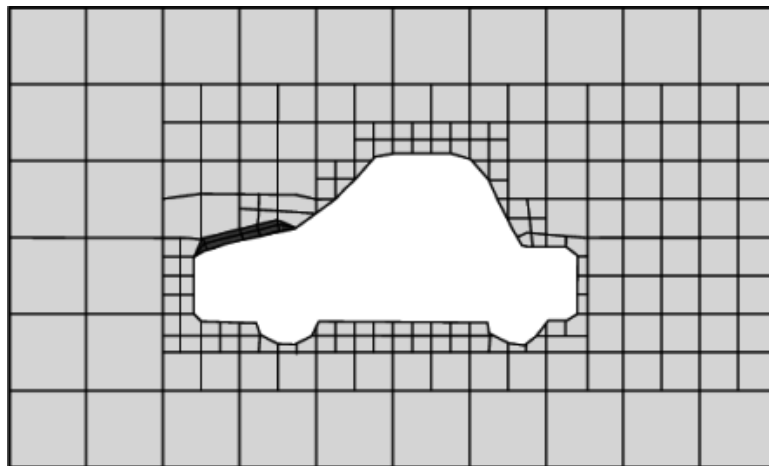


Abbildung 5.6: Die Abbildung zeigt das Hinzufügen von prismatischen Zellschichten an der Oberfläche der Geometrie.

6 Diskretisierungen und Gleichungslöser

6.1 Diskretisierungsverfahren: **fvSchemes**

6.1.1 Zeitliche Diskretisierung: **ddtSchemes**

OPENFOAM stellt folgende zeitliche Diskretisierungsverfahren zur Verfügung:

- CoEuler
- CrankNicolson
- Euler
- SLTS
- backward
- bounded
- localEuler
- steadyState

6.1.2 Konvektiver Term: **divSchemes**

CoBlended Gamma GammaV LUST MUSCL MUSCLV Minmod MinmodV OSPRE OSPREV Phi QUICK QUICKV SFCD SFCDV SuperBee SuperBeeV UMIST UMISTV biLinearFit blended clippedLinear cubic cubicUpwindFit downwind filteredLinear filteredLinear2 filteredLinear2V filteredLinear3 filteredLinear3V fixedBlended limitWith limitedCubic limitedCubicV limitedLinear limitedLinearV limiterBlended linear linearFit linearPureUpwindFit linearUpwind linearUpwindV localBlended localMax localMin midPoint outletStabilised pointLinear quadraticFit quadraticLinearFit quadraticLinearUpwindFit quadraticUpwindFit reverseLinear skewCorrected upwind vanAlbada vanAlbadaV vanLeer vanLeerV weighted

6.1.3 Gradienten: **gradSchemes**

CoBlended Gamma Gamma01 LUST MUSCL MUSCL01 Minmod OSPRE QUICK SFCD SuperBee UMIST biLinearFit blended clippedLinear cubic cubicUpwindFit downwind filteredLinear filteredLinear2 filteredLinear3 fixedBlended harmonic limitWith limitedCubic limitedCubic01

limitedGamma limitedLimitedCubic limitedLimitedLinear limitedLinear limitedLinear01 limitedMUSCL limitedVanLeer limiterBlended linear linearFit linearPureUpwindFit linearUpwind localBlended localMax localMin midPoint outletStabilised pointLinear quadraticFit quadraticLinearFit quadraticLinearUpwindFit quadraticUpwindFit reverseLinear skewCorrected upwind vanAlbada vanLeer vanLeer01 weighted

6.1.4 Diffusiver Term: **laplacianSchemes**

CoBlended Gamma Gamma01 LUST MUSCL MUSCL01 Minmod OSPRE QUICK SFCD SuperBee UMIST biLinearFit blended clippedLinear cubic cubicUpwindFit downwind filteredLinear filteredLinear2 filteredLinear3 fixedBlended harmonic limitWith limitedCubic limitedCubic01 limitedGamma limitedLimitedCubic limitedLimitedLinear limitedLinear limitedLinear01 limitedMUSCL limitedVanLeer limiterBlended linear linearFit linearPureUpwindFit linearUpwind localBlended localMax localMin midPoint outletStabilised pointLinear quadraticFit quadraticLinearFit quadraticLinearUpwindFit quadraticUpwindFit reverseLinear skewCorrected upwind vanAlbada vanLeer vanLeer01 weighted

6.1.5 Interpolation: **interpolationSchemes**

CoBlended Gamma Gamma01 LUST MUSCL MUSCL01 Minmod OSPRE QUICK SFCD SuperBee UMIST biLinearFit blended clippedLinear cubic cubicUpwindFit downwind filteredLinear filteredLinear2 filteredLinear3 fixedBlended harmonic limitWith limitedCubic limitedCubic01 limitedGamma limitedLimitedCubic limitedLimitedLinear limitedLinear limitedLinear01 limitedMUSCL limitedVanLeer limiterBlended linear linearFit linearPureUpwindFit linearUpwind localBlended localMax localMin midPoint outletStabilised pointLinear quadraticFit quadraticLinearFit quadraticLinearUpwindFit quadraticUpwindFit reverseLinear skewCorrected upwind vanAlbada vanLeer vanLeer01 weighted

6.2 Gleichungslöser: **fvSolution**

6.2.1 Symmetrische Matrizen

Gleichungslöser GAMG ICCG PCG smoothSolver

Glättungsalgorithmen DIC DICGaussSeidel FDIC GaussSeidel nonBlockingGaussSeidel symGaussSeidel

7 Initial und Randbedingungen

In diesem Bereich passieren die meisten Fehler; aus diesem Grund sollte hier mit besonderer Vorsicht und Konzentration gearbeitet werden.

Die Initial und Randbedingungen verteilen sich auf zwei Ordner: *0* und *constant*

Es müssen an allen Rändern Randbedingungen definiert werden.

Grundsätzlich unterscheidet OPENFOAM zwischen verschiedenen Patch-typen: *base type*, *primitive type* und *derived type*. Dabei bedeutet *base type* einfachste geometrische Definitionen. Es gibt nur zwei verschiedene Arten: *patch* und *wall*.

Aus dem *base type* ergeben sich die *primitive types* für die Größe Φ (siehe Tabelle 7.1):

Typ	Beschreibung	Festzulegende Werte
fixedValue	Wert für Φ muss festgelegt werden	value
fixedGradient	Der Gradient für Φ muss festgelegt werden	gradient
zeroGradient	Der Gradient von Φ ist null	-
calculated	Die Werte für Φ werden durch andere Größen bestimmt	-
mixed	Mischt fixedValue und fixedGradient abhängig vom Wert in valueFraction	refValue, refGradient, valueFraction, value
	wie valueFraction, jedoch mit valueFraction als Tensor, sodass sich verschiedene Mischlevel in Normal- und Tangentialrichtung ergeben.	refValue, refGradient, valueFraction, value
directionMixed		

Tabelle 7.1: Terme und ihre Bezeichnungen in OpenFOAM

Aus der Kombination dieser Typen lassen sich die sog. *derived types* herleiten. Da es insgesamt 67 dieser Randbedingungen gibt, werden nachfolgend nur die wichtigsten aufgeführt.

SRFFreestreamVelocity SRFVelocity activeBaffleVelocity activePressureForceBaffleVelocity advective atmBoundaryLayerInletVelocity calculated codedFixedValue codedMixed cyclic cyclicACMI cyclicAMI cyclicSlip cylindricalInletVelocity directionMixed empty externalCoupled fixedGradient fixedInternalValue fixedJump fixedJumpAMI fixedMean fixedNormalInletOutletVelocity fixedNormalSlip fixedValue flowRateInletVelocity fluxCorrectedVelocity

freestream inletOutlet interstitialInletVelocity kqRWallFunction mapped mappedField mappedFixedInternalValue mappedFixedPushedInternalValue mappedFlowRate mappedVelocityFlux mixed movingWallVelocity nonuniformTransformCyclic oscillatingFixedValue outletInlet outletMappedUniformInlet outletPhaseMeanVelocity partialSlip pressureDirectedInletOutletVelocity pressureDirectedInletVelocity pressureInletOutletParSlipVelocity pressureInletOutletVelocity pressureInletUniformVelocity pressureInletVelocity pressureNormalInletOutletVelocity processor processorCyclic rotatingPressureInletOutletVelocity rotatingWallVelocity sliced slip supersonicFreestream surfaceNormalFixedValue swirlFlowRateInletVelocity symmetry symmetryPlane timeVaryingMappedFixedValue translatingWallVelocity turbulentInlet uniformFixedGradient uniformFixedValue uniformInletOutlet uniformJump uniformJumpAMI variableHeightFlowRateInletVelocity waveTransmissive wedge zeroGradient

8 Pre-Processing

9 Stationäre Rechnungen

9.1 SIMPLE-Algorithmus: SIMPLEFOAM

SIMPLE steht für *Semi-Implicit Method for Pressure Linked Equations*. Dieses numerische Verfahren ist ein iterativer Algorithmus zum lösen der stationären Navier-Stokes-Gleichungen.

Die iterativen Schritte sind wie folgt:

1. Die Randbedingungen setzen
2. Die Gradienten von Druck und Geschwindigkeit berechnen
3. Die diskretisierte Impulsgleichung lösen um das innere Geschwindigkeitsfeld zu berechnen
4. Berechne den (unkorrigierten) Masseflux an den Zellflächen
5. Löse die Druckkorrekturgleichung für alle Zellen
6. Korrigiere das Druckfeld: $p^{k+1} = p^k + f_{relax} \cdot p$, mit f_{relax} = Relaxationsfaktor für p
7. Korrigiere den Druck an den Randbedingungen
8. Korrigiere den Massenfluss an den Zellflächen
9. Korrigiere die Geschwindigkeiten in den Zellen
10. Berechne die Dichte neu (Änderungen sind durch Druckänderungen möglich)

10 Instationäre Rechnungen

10.1 PIMPLE-Algorithmus: PIMPLEFOAM

PIMPLE steht für *merged PIs*o *si*MPL*E*.

10.2 PISO-Algorithmus: PISOFOAM

10.3 IC0-Algorithmus: ICOFOAM

11 Bewegte Gitter

OpenFOAM bietet vielfige Mglichkeiten Bewegungen in den Simulationen abzubilden. Im folgenden Kapitel sollen die wichtigsten Methoden kurz vorgestellt und erkl werden.

Zunst muss unterschieden werden, ob es sich bei der zu simulierenden Strmung um eine quasistation Strmung handelt, oder um eine echt instation.

Zunst gilt es den Typ von bewegtem Gitter festzulegen. Dies geschieht ber die Datei *constant/dynamicMeshDict*. Verfgbare Gittertypen sind:

11.1 MRF

Für den Fall einer Quasistationären Bewegung, bspw. die Rotation einer Gittertypen die Möglichkeit mit sog. MRF-Solvern zu rechnen. MRF steht für *Multi-Reference-Frame* und bedeutet, dass an Stelle eines echten bewegten Gitters in einem rotierenden Bezugssystem gerechnet wird. Dies birgt mehrere Vorteile, als auch Nachteile. Die Navier-Stokes-Gleichung verkompliziert sich durch die zusätzlichen Terme, welche durch das rotierende Koordinatensystem in die Gleichung einfließen:

Name	
dynamicInkJetFvMesh	
dynamicMotionSolverFvMesh	
dynamicRefineFvMesh	
movingConeTopoFvMesh	
multiSolidBodyMotionFvMesh	
rawTopoChangerFvMesh	
solidBodyMotionFvMesh	
staticFvMesh	

$$\frac{Du}{Dt} = -\frac{1}{\rho} \cdot \nabla p - 2(\Omega \times u) - (\Omega \times r) + \nu \nabla^2 u - gk \quad (11.1)$$

Gleichung 11.1 zeigt die Navier-Stokes Gleichung für ein rotierendes Bezugssystem.

Die Vorteile eines solchen Bezugssystems sind, dass keine Bewegungsgleichungen für die Gitterpunkte gelöst werden müssen. Des Weiteren ist auch keine räumliche Interpolation der simulierten Größen vom alten Gitter zum neuen Gitter nötig. Dadurch lassen sich sowohl potentielle Fehler, als auch Rechenzeit verringern. In der Standarddistribution (momentan: OpenFOAM 2.3.x) ist die Unterstützung für MRF grundsätzlich enthalten, unter anderem in *simpleFoam* und *pimpleFoam*.

Definiert wird die Rotation durch einen Eintrag in der Datei `system/fvOptions`, in der eine MRF-Option definiert wird. Listing B.1 listet einen Beispieleintrag aus der Datei `fvOptions`. Der Code ist aus einem der zu OpenFOAM gehörenden Tutorials entnommen.

11.2 AMI

AMI steht für *Arbitrary Mesh Interface*, was so viel wie beliebige-Gitter-Schnittstelle bedeutet. Im Grunde genommen wird die Rechendom Ω in mehrere Bereiche unterteilt. An den Schnittstellen dieser Bereiche werden die Simulationsgrößen interpoliert. Dadurch ist eine gleitende Bewegung der Gitter zueinander möglich, welche die Simulation von Bewegungen mit 6 Freiheitsgraden ermöglicht.

Um Simulationen mit dieser Technik durchzuführen sind mehrere Schritte nötig.

bild ei-
nes AMI-
Gitters
einsetzen

11.3 RBF

11.4 dynamicTopoFvMesh

Diese Bibliothek ermöglicht CFD-Simulationen mit den höchsten Freiheitsgraden. Dies wird durch eine dynamische Gittergenerierung zur Laufzeit der Simulationen gewährleistet. Der Code ist experimentell und die Simulationen schwer zu kontrollieren.

Simulationen mit dieser numerischen Technik basieren auf einer Kombination aus der Gitterbibliothek *Mesquite* und der eigentlichen Bibliothek *dynamicTopoFvMesh*.

Momentan ist noch ein Programmierfehler vorhanden, der das saubere durchführen einer dynamischen simulation verhindert, wenn der parameter `allowTableResize` auf `yes` gesetzt ist.

11.4.1 Mesquite

Diese Bibliothek ermöglicht das Glen eines numerischen Gitters. Dazu wird eine skalare Qualität, welche zu jedem Zeitschritt für jede einzelne Gitterzelle berechnet wird genutzt. Im Nachfolgenden sind diese aufgeführt.

Tabelle 11.2: Qualitätsmerkmale für *Mesquite*

Name	Formel
<i>UntangleBeta</i>	
<i>AspectRatioGamma</i>	
<i>InverseMeanRatio</i>	
<i>EdgeLengthQuality</i>	
<i>LocalSize</i>	
<i>VertexConditionNumber</i>	
<i>ConditionNumber</i>	
<i>MeanRatio</i>	
<i>EdgeLength</i>	

Tabelle 11.4: Optimierungsalgorithmen

Name
angularOscillatingDisplacement
angularOscillatingVelocity
calculated
cyclic
empty
fixedValue
generic
global
mixed
oscillatingDisplacement
oscillatingFixedValue
oscillatingVelocity
processor
slip
surfaceDisplacement
surfaceSlipDisplacement
symmetryPlane
timeVaryingUniformFixedValue
uniformFixedValue
value
wedge
zeroGradient

Tabelle 11.5: Constraints

Name	Beschreibung
axis	Nur translative Bewegung entlang einer Achse; keine Rotation, nur um diese Achse. Einziger benötigter Parameter ist die Ausrichtung der Achse <i>axis</i> (x y z)
line	Erlaubt translative Bewegungen nur in Richtung der festgelegten Linie. Rotation ist ebenfalls erlaubt. Benötigte Parameter sind: <i>CentreOfRotation</i> (x y z) und <i>orientation</i> (x y z)
orientation	
plane	
point	Setzt eine translative Begrenzung mit einem Punkt. Schlüsselwort ist <i>centreOfRotation</i> (x y z). Wird dies nicht gesetzt und es werden keine anderen constraints gesetzt, wird automatisch der Masseschwerpunkt als punktrestraint gesetzt.

Tabelle 11.6: Restraints

Name	Beschreibung
linearAxialAngularSpring	
linearDamper	
linearSpring	
sphericalAngularDamper	
sphericalAngularSpring	
tabulatedAxialAngularSpring	

11.5 *sixDoFRigidBodyMotionSolver*

Tabelle 11.3: Optimierungsalgorithmen

Name
<i>SteepestDescent</i>
<i>TrustRegion</i>
<i>NonSmoothDescent</i>
<i>ConjugateGradient</i>
<i>SmartLaplacian</i>
<i>QuasiNewton</i>
<i>FeasibleNewton</i>
<i>Laplacian</i>
<i>Randomize</i>

12 Mehrphasenrechnungen

13 Wärme

14 Post-Processing

- `execFlowFunctionObject`
- `vorticity`
- `yPlusRAS`
- `yPlusLES`
- `wallShearStress`
- `Q`
- `Lambda`

15 Paraview

16 Nachschlagewerke und Literatur

Anhang

A constant

Listing A.1: Datei *constant/polyMesh/blockMesh*

```
1
2 /*-----* C++ *-----*\
3 |=====|
4 | \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
5 | \ \ / O p e r a t i o n | Version: 2.1.0 |
6 | \ \ / A n d | Web: www.OpenFOAM.com |
7 | \ \ / M a n i p u l a t i o n | |
8 \*-----*/
9 FoamFile
10 {
11     version      2.0;
12     format       ascii;
13     class        dictionary;
14     object       blockMeshDict;
15 }
16 // ***** //
17
18 // globale Skalierungsgröße; gilt für das gesamte Gitter
19 convertToMeters 1.000000;
20
21 // Die Eckpunkte des Gitters. Im Falle dieser Geometrie die äußeren Kanten, sowie die Kanten des NACA 0012 profils. Zu beachten ist
22 // dabei, dass der erste Knoten innerhalb von OpenFOAM als Knoten 0, anstelle von Knoten 1 angesprochen wird
23 vertices
24 (
25     (-7.530630 0.050000 1.581494) // Knoten 0
26     ...
27     (16.000000 -0.050000 -8.000000) //Knoten 23
28 );
29
30 // Das gitter besteht insgesamt aus 6 Blöcken, welche im nachfolgenden Eintrag definiert werden. Wichtig zu beachten ist die Reihenfolge
31 // der angegebenen Kantenpunkte (vertices). Grundsätzlich ist egal in welcher Drehrichtung die Punkte angegeben werden, so lange
32 // die Drehrichtung beibehalten wird.
33 blocks
34 (
35     hex (4 5 1 0 16 17 13 12) (74 100 1) edgeGrading (1 0.020000 0.020000 1 500.000000 500.000000 500.000000 500.000000 1 1 1 1)
36     ...
37     hex (19 20 23 22 7 8 11 10) (150 100 1) simpleGrading (100.000000 500.000000 1)
38 );
39
40 // An dieser Stelle werden die Kanten des Gitters definiert. OpenFOAM stellt dafür verschiedene Verfahren zur Verfügung, unter anderem
41 // gerade Kanten (edges), Splines (spline) und Kreisbögen (arc)
42 edges
43 (
44     // Splineinterpolation entlang der angegebenen Koordinaten von Knoten 4 zu Knoten 5
45     spline 4 5
46     (
47         (0.000159 0.050000 0.000682)
48         ...
49         (0.301576 0.050000 -0.002021)
50     )
51     ...
52     // zur Übersicht wurden die restlichen Spline-Einträge gelöscht
53     ...
54     // Kreisbogen von Knoten 0 zu Knoten 1 mit dem Verbindungspunkt in den Klammern
55     arc 0 1 (-5.351755 0.050000 5.656854)
56     ...
57     arc 12 21 (-5.351755 -0.050000 -5.656854)
58 );
59
60 // die Randbedingungen
61 boundary
```

Anhang A constant

```
59  (
60    inlet
61    {
62      type patch;
63      faces
64      (
65        (1 0 12 13)
66        (0 9 21 12)
67      );
68    }
69
70    outlet
71    {
72      type patch;
73      faces
74      (
75        (11 8 20 23)
76        (8 3 15 20)
77      );
78    }
79
80    topAndBottom
81    {
82      type patch;
83      faces
84      (
85        (3 2 14 15)
86        (2 1 13 14)
87        (9 10 22 21)
88        (10 11 23 22)
89      );
90    }
91
92    airfoil
93    {
94      type wall;
95      faces
96      (
97        (5 4 16 17)
98        (7 5 17 19)
99        (4 6 18 16)
100       (6 7 19 18)
101      );
102    }
103  );
104
105  mergePatchPairs
106  (
107  );
108
109  // ***** //
```


B system

Listing B.1: Datei *system/fvOptions*

```
1  /*-----*- C++ -*-----*\
2  | ===== |
3  | \\ / F ield | OpenFOAM: The Open Source CFD Toolbox |
4  | \\ / O peration | Version: 2.3.0 |
5  | \\ / A nd | Web: www.OpenFOAM.org |
6  | \\ / M anipulation |
7  \*-----*/
8  FoamFile
9  {
10     version      2.0;
11     format        ascii;
12     class         dictionary;
13     location      "system";
14     object        fvOptions;
15 }
16 // ***** //
17
18 MRF1
19 {
20     type          MRFSource;
21     active         true;
22     selectionMode  cellZone;
23     cellZone       rotor;
24
25     MRFSourceCoeffs
26     {
27         origin     (0 0 0);
28         axis        (0 0 1);
29         omega       104.72;
30     }
31 }
32
33
34 // ***** //
```

Listing B.2: Datei *system/fvSchemes*

```

1  /*-----* C++ *-----*\
2  |=====|
3  | \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
4  | \ \ / O p e r a t i o n | Version: 2.3.0 |
5  | \ \ / A n d | Web: www.OpenFOAM.org |
6  | \ \ / M a n i p u l a t i o n | |
7  \*-----*/
8  FoamFile
9  {
10     version      2.0;
11     format        ascii;
12     class         dictionary;
13     object        fvSchemes;
14 }
15 // ***** //
16
17 ddtSchemes
18 {
19     default        steadyState;
20 }
21
22 gradSchemes
23 {
24     default        Gauss linear;
25     grad(U)        cellLimited Gauss linear 1;
26 }
27
28 divSchemes
29 {
30     default        none;
31     div(phi,U)     bounded Gauss linearUpwindV grad(U);
32     div(phi,k)     bounded Gauss upwind;
33     div(phi,omega) bounded Gauss upwind;
34     div((nuEff*dev(T(grad(U)))) Gauss linear;
35 }
36
37 laplacianSchemes
38 {
39     default        Gauss linear corrected;
40 }
41
42 interpolationSchemes
43 {
44     default        linear;
45 }
46
47 snGradSchemes
48 {
49     default        corrected;
50 }
51
52 fluxRequired
53 {
54     default        no;
55     p;
56 }

```

Listing B.3: Datei `system/fvSolution`

```

1  /*-----* C++ -----*/
2  |=====|
3  | \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
4  | \ \ / O p e r a t i o n | Version: 2.3.0 |
5  | \ \ / A n d | Web: www.OpenFOAM.org |
6  | \ \ / M a n i p u l a t i o n | |
7  /*-----*/
8  FoamFile
9  {
10     version      2.0;
11     format        ascii;
12     class         dictionary;
13     object        fvSolution;
14 }
15 // *****
16
17 solvers
18 {
19     p
20     {
21         solver      GAMG;
22         tolerance    1e-7;
23         relTol       0.01;
24         smoother     GaussSeidel;
25         nPreSweeps    0;
26         nPostSweeps   2;
27         cacheAgglomeration on;
28         agglomerator  faceAreaPair;
29         nCellsInCoarsestLevel 10;
30         mergeLevels   1;
31     }
32
33     U
34     {
35         solver      smoothSolver;
36         smoother     GaussSeidel;
37         tolerance    1e-8;
38         relTol       0.1;
39         nSweeps       1;
40     }
41
42     k
43     {
44         solver      smoothSolver;
45         smoother     GaussSeidel;
46         tolerance    1e-8;
47         relTol       0.1;
48         nSweeps       1;
49     }
50
51     omega
52     {
53         solver      smoothSolver;
54         smoother     GaussSeidel;
55         tolerance    1e-8;
56         relTol       0.1;
57         nSweeps       1;
58     }
59 }
60
61 SIMPLE
62 {
63     nNonOrthogonalCorrectors 0;
64 }
65
66 potentialFlow
67 {
68     nNonOrthogonalCorrectors 10;
69 }
70
71 relaxationFactors
72 {
73     fields
74     {
75         p            0.3;

```

```
76     }
77     equations
78     {
79         U          0.7;
80         k          0.7;
81         omega      0.7;
82     }
83 }
84
85 cache
86 {
87     grad(U);
88 }
```

Listing B.4: Datei *system/controlDict*

```

1  /*-----* C++ *-----*\
2  |=====|
3  | \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
4  | \ \ / O p e r a t i o n | Version: 2.3.0 |
5  | \ \ / A n d | Web: www.OpenFOAM.org |
6  | \ \ / M a n i p u l a t i o n | |
7  \*-----*/
8  FoamFile
9  {
10     version      2.0;
11     format        ascii;
12     class         dictionary;
13     object        controlDict;
14 }
15 // *****
16
17 libs
18 (
19     "libOpenFOAM.so"
20     "libincompressibleTurbulenceModel.so"
21     "libincompressibleRASModels.so"
22 );
23
24 application      simpleFoam;
25
26 startFrom         latestTime;
27
28 startTime         0;
29
30 stopAt            endTime;
31
32 endTime          500;
33
34 deltaT           1;
35
36 writeControl      timeStep;
37
38 writeInterval     100;
39
40 purgeWrite        0;
41
42
43 //- Uncomment to have regular (every 2 hours of run time) restart files
44 //secondaryWriteControl    cpuTime; // runtime
45 //secondaryWriteInterval   7200;    // seconds
46 //secondaryPurgeWrite      1;       // keep all but last dump
47
48
49 writeFormat       binary;
50
51 writePrecision    6;
52
53 writeCompression  uncompressed;
54
55 timeFormat        general;
56
57 timePrecision     6;
58
59 runTimeModifiable true;
60
61 functions
62 {
63     #include "readFields"
64     #include "streamLines"
65     #include "wallBoundedStreamLines"
66     #include "cuttingPlane"
67     #include "forceCoeffs"
68 }

```

C 0

Listing C.1: Datei 0/U

```

1  /*-----*- C++ -*------*\
2  | ===== |
3  | \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
4  | \ \ / O p e r a t i o n | Version: 2.3.0 |
5  | \ \ / A n d | Web: www.OpenFOAM.org |
6  | \ \ / M a n i p u l a t i o n | |
7  \*-----*/
8  FoamFile
9  {
10     version      2.0;
11     format        ascii;
12     class         volVectorField;
13     object        U;
14 }
15 // ***** //
16
17 dimensions      [0 1 -1 0 0 0 0];
18
19 internalField    uniform (0 0 0);
20
21 boundaryField
22 {
23     movingWall
24     {
25         type      fixedValue;
26         value      uniform (1 0 0);
27     }
28
29     fixedWalls
30     {
31         type      fixedValue;
32         value      uniform (0 0 0);
33     }
34
35     frontAndBack
36     {
37         type      empty;
38     }
39 }
40
41 // ***** //

```

Listing C.2: Datei 0/p

```

1  /*-----* C++ *-----*\
2  | ===== |
3  | \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
4  | \ \ / O p e r a t i o n | Version: 2.3.0 |
5  | \ \ / A n d | Web: www.OpenFOAM.org |
6  | \ \ / M a n i p u l a t i o n | |
7  \*-----*/
8  FoamFile
9  {
10     version      2.0;
11     format        ascii;
12     class         volScalarField;
13     object        p;
14 }
15 // ***** //
16
17 dimensions      [0 2 -2 0 0 0 0];
18
19 internalField    uniform 0;
20
21 boundaryField
22 {
23     movingWall
24     {
25         type      zeroGradient;
26     }
27
28     fixedWalls
29     {
30         type      zeroGradient;
31     }
32
33     frontAndBack
34     {
35         type      empty;
36     }
37 }
38
39 // ***** //

```

D Formeln zu *RANS-Randbedingungen*

Formelsammlung zum abschätzen der Initialwerte für RANS-basierte Turbulenzmodelle. Folgende Variablen haben feste Werte:

Tabelle D.1: Feste Variablen

Variable	Wert oder Formel
$T_u = \frac{u'}{U} = 0.001..0.1$	Turbulenzintensität
$C_\mu = 0.09$	Standardvariable in vielen RANS-Modellen
L	Turbulente Längenskala
u'	Turbulente Schwankungsgeschwindigkeit

Tabelle D.2: Empirische Formeln zur Abschätzung der turbulenten Viskosität $\tilde{\nu}$

Formel	Anmerkung
$\tilde{\nu} = 1.2247 \cdot (U \cdot I \cdot L)$	Freie Umströmung

Tabelle D.3: Empirische Formeln zur Abschätzung der turbulenten kinetischen Energie k

Formel	Anmerkung
$k = 1.5 \cdot (T_u \cdot U_{in})^2$	Freie Umströmung
$k = T_u^2 \cdot \bar{U}_{in}^2$	Abgewandelte Form

Tabelle D.4: Empirische Formeln zur Abschätzung der Dissipationsrate ε

Formel	Anmerkung
$\varepsilon = \frac{C_v \cdot k^{1.5}}{0.03 \cdot L}$	Freie Anströmung
$\varepsilon = \frac{C_\mu^{0.75} \cdot k^{1.5}}{0.03 \cdot L}$	Abgewandelte Form

Tabelle D.5: Empirische Formeln zur Abschätzung der spezifischen Dissipationsrate ω

Formel	Anmerkung
$\omega = \frac{\sqrt{k}}{L}$	mit L = turbulente Längenskale
$\omega = \frac{\varepsilon}{k}$	Standard

E Checkliste zu *snappyHexMesh*

Diese Checkliste orientiert sich an der allgemeinen Struktur der *snappyHexMesh*-Kontrolldatei *snappyHexMeshDict*.

- ☐ Prismatische Zellschichten an der Oberfläche? -> auf true setzen
- ☐ Name der Inputdatei (normalerweise stl-datei) festgelegt?
- ☐ maxLocalCells > gewünschte Anzahl der Zellen?
- ☐ maxLocalCells < 80% RAM?
- ☐ maxGlobalCells < 80% RAM?
- ☐ refinementBox(en) festgelegt?
- ☐ maxLoadUnbalance < 0.11?
- ☐ nCellsBetweenLayers < 3 (für sehr große Gitter)?
- ☐ nCellsBetweenLayers > 3 (für kleine Gitter und 2D)?
- ☐ features in Benutzung -> refinementLevel == min. surfaceRefinement?
- ☐ surfaceRefinement?
- ☐ refinementBox? -> in refinementRegions festlegen?
- ☐ meshSelection nicht auf Fläche?
- ☐ gitter zu wellig? surfaceFeatureExtract benutzen und nSmoothPatch hoch
- ☐ surfaceLayers? -> Name des Patches, nicht geometrie!
- ☐ es werden keine Layer generiert? nGrow auf 0 oder 1

F Checkliste zu *pimpleFoam*

Diese Checkliste listet die wichtigsten Einstellungen von *pimpleFoam*

□