FancyFOAM
Der Versuch einer umfassenden Dokumentation von OpenFOAM

Aljoscha N. Sander

19. Juni 2015

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

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, wass 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 Distrubtionen 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

[1] frei 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).

[2] Fun 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, das 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] DATE1 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

[3] u.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):

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

Durch die Inkompressibilität vereinfachst sich die Kontiuumsgleichung zu Gleichung 2.2:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2.2}$$

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 $u_j \frac{u_i}{x_j}$, 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 $\nu \frac{\partial^2 u_i}{\partial u_j^2}$, 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 (Newtonsch, Binghamsch, etc.) fest.

In den Dateien RASProperties, LESProperties und turbulenceProperties werden etwaige Turbulenzemodelle 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.

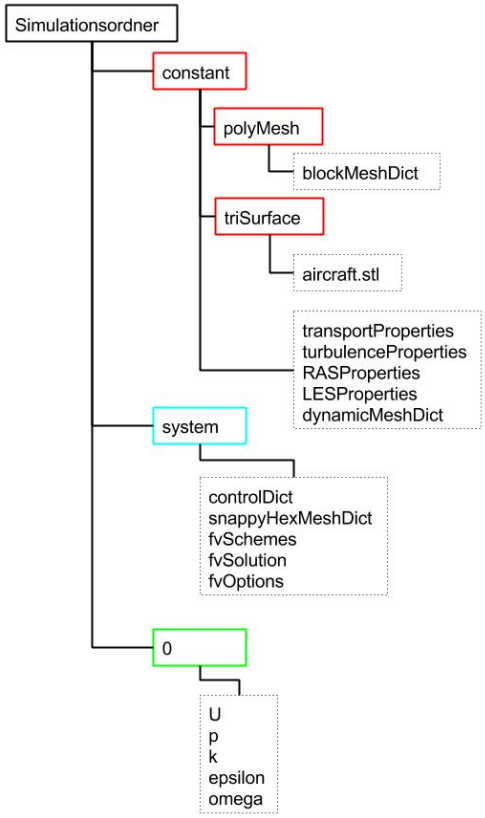

Abbildung 4.1: Die Ordnerstruktur innerhalb einer OpenFOAM Simulation. Der system ordner beinhaltet die Einstellungen der Lösungsprogramme, der 0 Ordner beinhaltet die Initialbedingungen und der constant Ordner das Gitter und die Randbedingungen

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 notwending, 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 .

anderen Parameter raussuchen

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

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 $u_j \frac{u_i}{x_j}$	`divSchemes`
(Druck)Gradient $-\frac{\frac{p}{\rho}+gk}{\partial x_i}$	`gradSchemes`
Diffusiver Term $\nu \frac{\partial^2 u_i}{\partial u_j^2}$	`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 Ralaxationsfaktoren 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 festegelegt; 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.

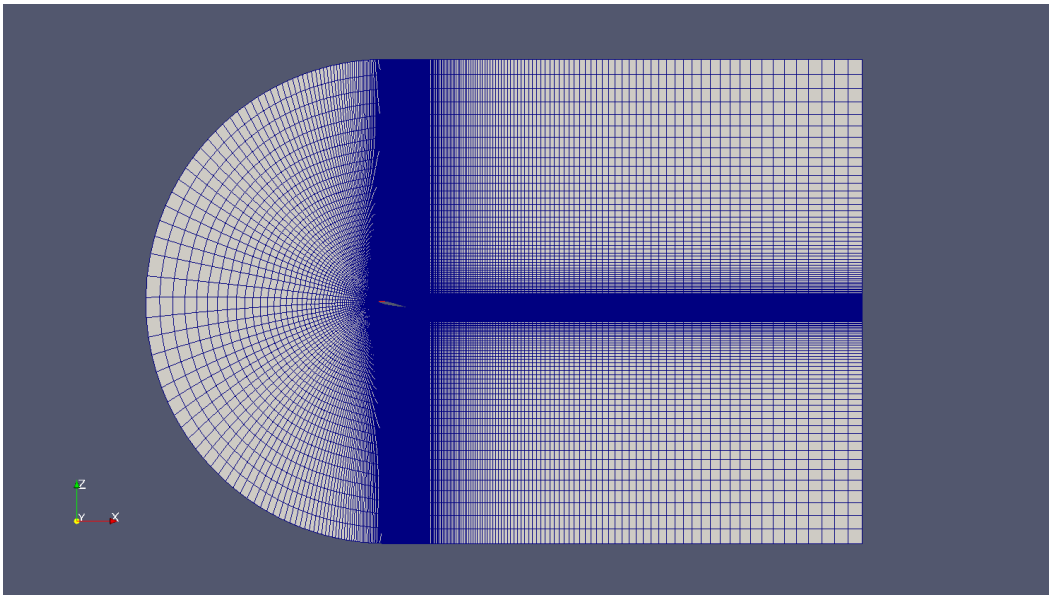

Abbildung 5.1: Ein mit *blockMesh* generiertes Gitter eines NACA 0012 Flügelprofils. Deutlich zu erkennen ist das Grading im Gitter.

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 richtig 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.

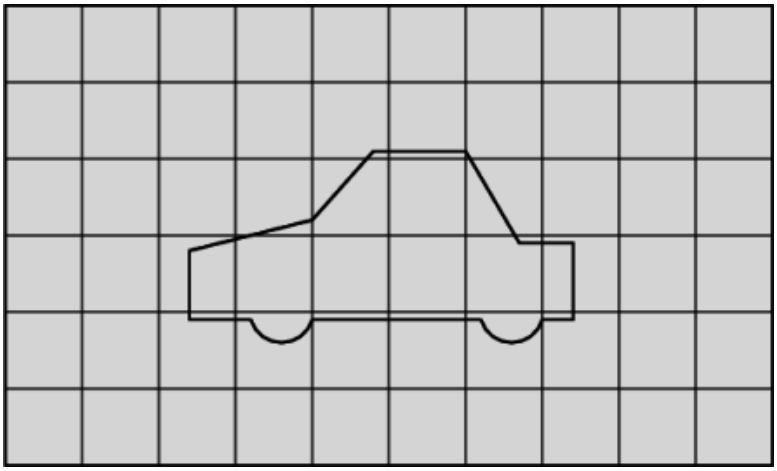

Abbildung 5.2: Geometrie innerhalb eines blockstrukturierten Gitters für die Gittergeneration mit snappyHexMesh. Das blockstrukturierte Gitter muss die Oberfläche überall dort umschließen, wo es zu einer Interaktion zwischen Fluid und Geometrie kommen soll.

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 blockstrukturietem 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 seinen hier die folgenden aufgezählt:

- `-parallel` soll *snappyHexMesh* parallel genutzt werden, so ist dieser Parameter unerlässlich.
- `-overwrite` *snappyHexMesh* schreibt alle drei Bearbeitungschritte 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 Geomtrie festgelegt (Listing 5.1):

Listing 5.1: Geometriedefinition innerhalb von *snappyHexMeshDict*

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

    verfeinerung
    {
        type searchableBox;
        min (-1.0 -0.7 0.0);
        max ( 8.0  0.7 2.5);
    }
};
```

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

- `closedTriSurfaceMesh`: benötigt als Input eine geschlossene Oberflächengeometrie
- `distributedTriSurfaceMesh`: raus finden was das ist
- `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` testen
- `searchablePlane` testen
- `searchablePlate` testen
- `searchableSphere`: generiert eine Kugel. Inputparameter sind *radius r* und *centre (x y z)*
- `searchableSurfaceCollection` dafuq?
- `searchableSurfaceWithGaps` dafuq?
- `triSurfaceMesh`: benötigt als Input eine Oberflächengeometrie.

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

Listing 5.2: *castellatedMeshControls* innerhalb von *snappyHexMeshDict*

```
castellatedMeshControls
{
  // maximal zulässige Anzahl von Zellen pro Prozessor
  // ACHTUNG: bricht Verfeinerung ab sobald Anzahl erreicht ist
    maxLocalCells 100000;

  // maximal zulässige Anzahl von Zellen über alle Prozessoren summiert
    // ACHTUNG: bricht Verfeinerung ab sobald Anzahl erreicht ist
    maxGlobalCells 2000000;

  // sollten weniger als diese Anzahl an Zellen verfeinert werden, wird abgebrochen
    minRefinementCells 10;

  // maximal zulässiges Ungleichgewicht an Zellanzahlen zwischen Prozessoren. Wird dieses Niveau überschritten, werden die Zellen neu
        verteilt
    maxLoadUnbalance 0.10;

  // Anzahl an Zellschichten zwischen den Verfeinerungen
    nCellsBetweenLevels 3;

  // wird das zusätzliche Tool surfaceFeatureExtract genutzt wird die davon generierte Geometrie hier eingebunden und das
        Verfeinerungslevel festegelegt
    features
    (
        {
            file "motorBike.eMesh";
            level 6;
        }
    );

  // die Oberflächen zu denen hin verfeinert werden soll, werden hier festgelegt. Dabei kann es sich sowohl um die vorher eingelenen
        Oberflächengeometrien aus dateien als auch um intern festgelegte Geomtrien, bsp searchableSphere handeln.
    refinementSurfaces
    {
        oberflaeche
        {
            level (min max);
        }
    }

  // Dieser Winkel zwischen zwei angrenzenden Zellen entscheidet darüber, ab wann das nächste Verfeinerungslevel an der Oberfläche
        eingesetzt werden soll. Start bei min.
    resolveFeatureAngle 45;

  // Soll ein ganzer Zellbereich innerhalb eines bestimmten Raumes auf ein bestimmtes Maß verfeinert werden, wird dies hier festgelegt
    refinementRegions
    {
        refinementBox
        {
            mode inside;
            levels ((1E15 4));
        }
    }

  // diese Koordinate entscheidet darüber welcher Teil des Gitters behalten wird; der innerhalb der Geometrie oder der ausserhalb.
    locationInMesh (3.0001 3.0001 0.43);

  \todo{was ist das?}
    allowFreeStandingZoneFaces true;
}
```

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):

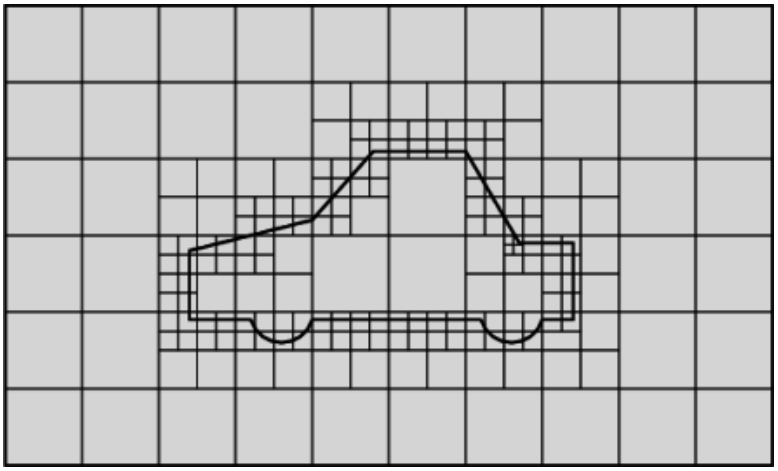

Abbildung 5.3: Die Zellverfeinerungen in der Umgebung der Oberflächengeometrie.

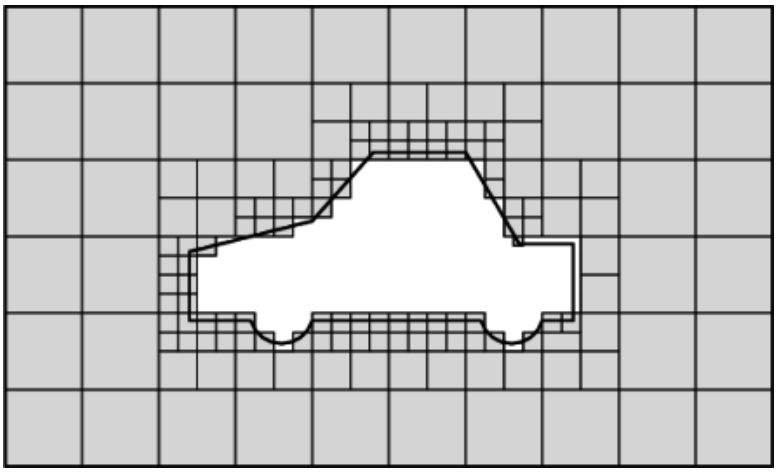

Abbildung 5.4: Das ungeglättete, verfeinerte Gitter in snappyHexMesh.

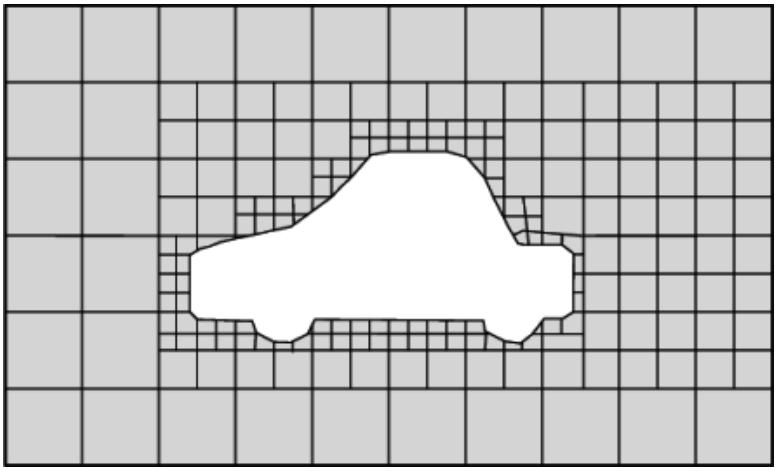

Abbildung 5.5: Das an der Oberfläche geglättete Gitter.

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*

```
snapControls
{
  // Dieser Parameter legt fest wie viele Iterationen (Wiederholungen) snappyHexMesh darauf verwenden darf die Oberfläche zu glätten.
    nSmoothPatch 3;

  // Dieser Faktor legt die Auflösungtoleranz fest; Die tatsächliche Toleranz entspricht der lokalen Kantenlänge multipliziert mit
       diesem Faktor.
    tolerance 2.0;

  // Lösungsiterationen des Gleichungslösers, welcher die Gitterpunkte verschiebt
    nSolveIter 30;

  // Relaxationsiterationen für den Gleichungslöser
    nRelaxIter 5;

  // Die folgenden Parameter gelten nur für den Fall das mit surfaceFeatureExtract gearbeitet wurde
    nFeatureSnapIter 10;

    implicitFeatureSnap false;

    explicitFeatureSnap true;

    multiRegionFeatureSnap false;
}
```

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*

```
addLayersControls
{
    // Hier wird angegeben, ob es sich um zum Grundgitter relative Größenangaben (true) oder um absolute Größenangaben (false) handelt.
    relativeSizes true;

    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 letzen 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 propritärer Gittergeneratoren, für welche OpenFOAM Konvertierungstools zur Verfügung stellt.

liste:

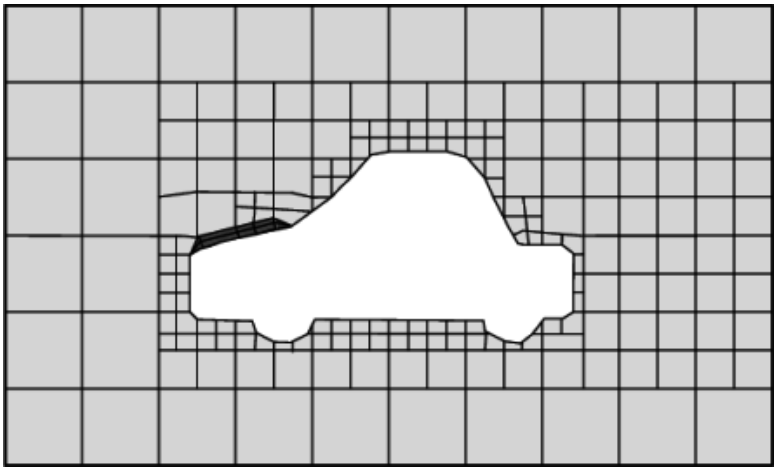

Abbildung 5.6: Die Abbildung zeigt das Hinzufügen von prismatischen Zellschichten an der Oberfläche der Geometrie.

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

Gläätungsalgorithmen DIC DICGaussSeidel FDIC GaussSeidel nonBlockingGaussSeidel symGaussSeidel

7 Inital 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*, *primitve type* und *derived type*. Dabei bedeutet *base type* einfachste geometrische Definitionen. Es gibt nut 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`
`directionMixed`	wie `valueFraction`, jedoch mit `valueFraction` als Tensor, sodass sich verschiedene Mischlevel in Normal- und Tangentialrichtung ergeben.	`refValue`, `refGradient`, `valueFraction`, `value`

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 Rechnugen

10.1 PIMPLE-Algorithmus: pimpleFoam

PIMPLE steht für *merged PIso siMPLE*.

10.2 PISO-Algorithmus: pisoFoam

10.3 ICO-Algorithmus: icoFoam

11 Bewegte Gitter

OpenFOAM bietet vielfältige Möglichkeiten Bewegungen in den Simulationen abzubilden. Im folgenden Kapitel sollen die wichtigsten Methoden kurz vorgestellt und erklärt werden. Zunächst muss unterschieden werden, ob es sich bei der zu simulierenden Strömung um eine quasistationäre Strömung handelt, oder um eine echt instationäre.

11.1 MRF

Für den Fall einer Quasistationären Bewegung, bspw. die Rotation einer gleichbleibenden Mixergeometrie oder Turbine bietet OpenFOAM 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:

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

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 Rechendomäne Ω in mehrere Bereiche Unterteilt. An den Schnittstellen dieser Bereich 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 eines AMI-Gitters einfügen

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 ield         | OpenFOAM: The Open Source CFD Toolbox           |
5   |  \\    /   O peration     | Version:  2.1.0                                 |
6   |   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
7   |    \\/     M anipulation  |                                                 |
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
        dabei, dass der erste Knoten innerhalb von OpenFOAM als Knoten 0, anstelle von Knoten 1 angesprochen wird
22  vertices
23  (
24      (-7.530630 0.050000 1.581494)   // Knoten 0
25  ...
26      (16.000000 -0.050000 -8.000000)   //Knoten 23
27  );
28
29  // Das gitter besteht insgesamt aus 6 Blöcken, welche im nachfolgenden Eintrag definiert werden. Wichtig zu beachten ist die Reihenfolge
        der angegebenen Kantenpunkte (vertices). Grundsätzlich ist egal in welcher Drehrichtung die Punkte angegeben werden, so lange
        die Drehrichtung beibehalten wird.
30  blocks
31  (
32      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)
33  ...
34      hex (19 20 23 22 7 8 11 10)   (150 100 1) simpleGrading (100.000000 500.000000 1)
35  );
36
37  // An dieser Stelle werden die Kanten des Gitters definiert. OpenFOAM stellt dafür verschiedene Verfahren zur Verfügung, unter anderem
        gerade Kanten (edges), Splines (spline) und Kreisbögen (arc)
38  edges
39  (
40  // Splineinterpolation entlang der angegebenen Koordinaten von Knoten 4 zu Knoten 5
41      spline 4 5
42          (
43              (0.000159 0.050000 0.000682)
44  ...
45              (0.301576 0.050000 -0.002021)
46          )
47
48  ...
49  // zur Übersicht wurden die restlichen Spline-Einträge gelöscht
50  ...
51  // Kreisbogen von Knoten 0 zu Knoten 1 mit dem Verbindungspunkt in den Klammern
52      arc 0 1 (-5.351755 0.050000 5.656854)
53  ...
54      arc 12 21 (-5.351755 -0.050000 -5.656854)
55  );
56
57  // die Randbedingungen
58  boundary
```

```
(
    inlet
    {
        type patch;
        faces
        (
            (1 0 12 13)
            (0 9 21 12)
        );
    }

    outlet
    {
        type patch;
        faces
        (
            (11 8 20 23)
            (8 3 15 20)
        );
    }

    topAndBottom
    {
        type patch;
        faces
        (
            (3 2 14 15)
            (2 1 13 14)
            (9 10 22 21)
            (10 11 23 22)
        );
    }

    airfoil
    {
        type wall;
        faces
        (
            (5 4 16 17)
            (7 5 17 19)
            (4 6 18 16)
            (6 7 19 18)
        );
    }
);

mergePatchPairs
(
);

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

B system

Listing B.1: Datei *system/fvOptions*

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

MRF1
{
    type            MRFSource;
    active          true;
    selectionMode   cellZone;
    cellZone        rotor;

    MRFSourceCoeffs
    {
        origin      (0 0 0);
        axis        (0 0 1);
        omega       104.72;
    }
}


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

Listing B.2: Datei *system/fvSchemes*

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

ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
    grad(U)         cellLimited Gauss linear 1;
}

divSchemes
{
    default         none;
    div(phi,U)      bounded Gauss linearUpwindV grad(U);
    div(phi,k)      bounded Gauss upwind;
    div(phi,omega)  bounded Gauss upwind;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p;
}
```

Listing B.3: Datei *system/fvSolution*

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

solvers
{
    p
    {
        solver           GAMG;
        tolerance        1e-7;
        relTol           0.01;
        smoother         GaussSeidel;
        nPreSweeps       0;
        nPostSweeps      2;
        cacheAgglomeration on;
        agglomerator     faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels      1;
    }

    U
    {
        solver           smoothSolver;
        smoother         GaussSeidel;
        tolerance        1e-8;
        relTol           0.1;
        nSweeps          1;
    }

    k
    {
        solver           smoothSolver;
        smoother         GaussSeidel;
        tolerance        1e-8;
        relTol           0.1;
        nSweeps          1;
    }

    omega
    {
        solver           smoothSolver;
        smoother         GaussSeidel;
        tolerance        1e-8;
        relTol           0.1;
        nSweeps          1;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
}

potentialFlow
{
    nNonOrthogonalCorrectors 10;
}

relaxationFactors
{
    fields
    {
        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*

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

libs
(
    "libOpenFOAM.so"
    "libincompressibleTurbulenceModel.so"
    "libincompressibleRASModels.so"
);

application     simpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         500;

deltaT          1;

writeControl    timeStep;

writeInterval   100;

purgeWrite      0;


//- Uncomment to have regular (every 2 hours of run time) restart files
//secondaryWriteControl    cpuTime; // runtime
//secondaryWriteInterval   7200;    // seconds
//secondaryPurgeWrite      1;       // keep all but last dump


writeFormat     binary;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

functions
{
    #include "readFields"
    #include "streamLines"
    #include "wallBoundedStreamLines"
    #include "cuttingPlane"
    #include "forceCoeffs"
}
```

C 0

Listing C.1: Datei *0/U*

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

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

internalField   uniform (0 0 0);

boundaryField
{
    movingWall
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }

    fixedWalls
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    frontAndBack
    {
        type            empty;
    }
}

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

Listing C.2: Datei *0/p*

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

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

internalField   uniform 0;

boundaryField
{
    movingWall
    {
        type            zeroGradient;
    }

    fixedWalls
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}

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

D Formeln zu *RANS-Randbedingungen*

Formelsammlung zum abschätzen der Initalwerte für RANS-basierte Tubulenzmodelle. Folgende Variablen haben feste Werte:

Tabelle D.1: Feste Variablen

Variable	Wert oder Formel
$T_u = \frac{u'}{U} = 0.001..0.1$	Turbulenzintesität
$C_\mu = 0.09$	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
$\varepsilon = \frac{C_\nu \cdot k^{1.5}}{0.03 \cdot L}$	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*

- □