Technische Hochschule Ingolstadt

Simulation eines Wirbelrohrs in OpenFOAM

Zwischenstand

Vinzenz Götz
00107303

Bericht SHK

Prof. Dr. Konrad Költzsch

20.08.2021

Abstract

Der folgende Bericht fasst den Zwischenstand der Simulation eines Wirbelrohres zusammen. Das (Ranque-Hilsch-)Wirbelrohr wurde von G. Ranque erfunden [Ran34] und enthält keine beweglichen Teile.

In einem Wirbelrohr wird ein tangential eingespeister Luftstrom in zwei Ströme, einer warm und der Andere kalt, aufgeteilt. Die Temperaturdifferenz zwischen den Ausgangsströmen beträgt je nach Bauart bis zu 30 K [Bru69]

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einführung

1.1. Inspiration

Die Inspiration für das Projekt Wirbelrohr wurde in einem Youtube-Video von “This Old Tony“ [Ton16] gefunden. In dem Video wird ein Wirbelrohr oder “Vortex Tube“ gebaut und demonstriert. Eine Weile später wurde das Projekt zusammen mit Herrn Kölztzsch gestartet.

1.2. Prinzipielle Funktion

Das Wirbelrohr funktioniert indem tangential Druckluft in das Rohr geleitet wird. Dort bildet die Luft gezwungenermaßen einen Wirbel, der sich mit hoher Geschwindigkeit dreht. So wirbelnd wandert die Strömung in Richtung des warmen Auslasses, welcher sich am weitesten vom Einlass entfernt befindet. Auf dem Weg durch das Rohr dreht der Wirbel um und wandert in der Mitte wieder entgegengesetzt zu den äußeren Schichten in Richtung des kalten Auslasses, welcher sich neben dem Einlass befindet.

1.3. Geometrie

Die verwendete Geometrie wurde im Laufe des Projekts weitereintwickelt. Es entstanden hierbei eine Handvoll von Ausführungen.

1.3.1. Geometrie von This Old Tony

This Old Tony stellte neben seinem Video zum Wirbelrohr auch die von ihm verwendete Geometrie zur verfügung. Diese ist in Abbildungen 1.1 und 1.2 dargestellt.

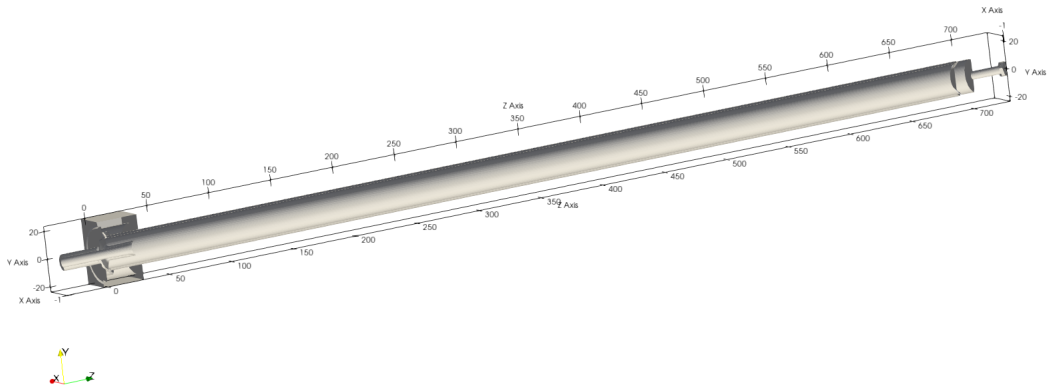

Abbildung 1.1.: Schnitt durch die erste verwendete Geometrie

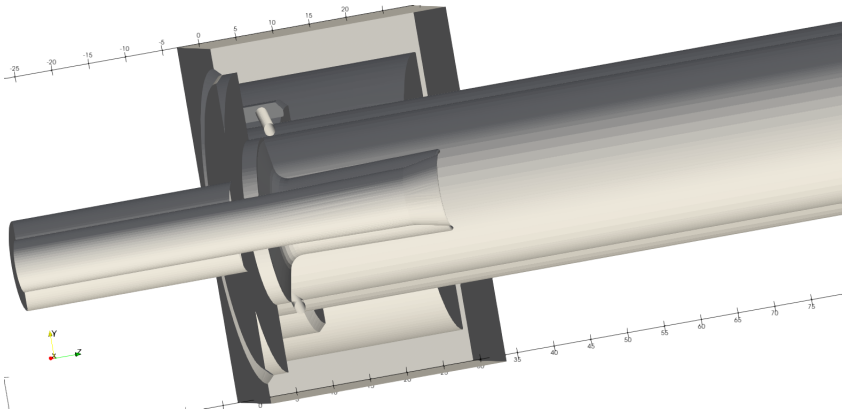

Abbildung 1.2.: Nahaufnahme der Vorkammer mit Durchgang zur Wirbelkammer

Hier ergaben sich bei der Simulation unter Anderem das Problem, dass die tangentialen Verbindungen zwischen “Vorkammer“ und “Wirbelkammmer“ so eng waren, dass diese nicht mehr vernetzt wurden und die Simulation abbrach weil OpenFOAM den Einlass-Patch nicht fand. Dieses Problem kommt allzu oft vor und sollte bei Fehlern wie patch not foundïn Betracht gezogen werden. Dadurch musste das Refinement im Kammerbereich sehr weit erhöht werden, was die Anzahl der Zellen sehr weit erhöhte.

1.3.2. Gröbere Geometrie

Durch weitere Recherche wurde eine Vereinfachung der Geometrie erzielt. Diese wurde einmal nur konzeptionell und darauffolgend auch in den Maßen übernommen. In Abbildung 1.3 ist diese zu sehen.

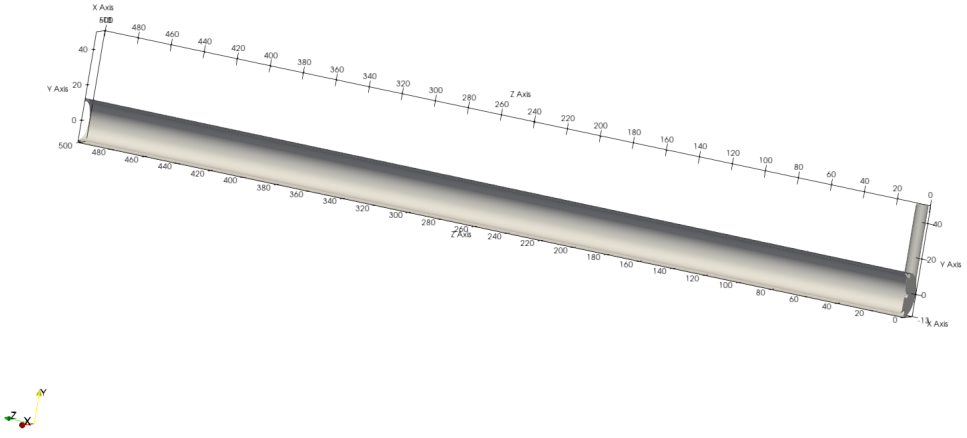

Abbildung 1.3.: Schnitt durch die grobe Geometrie

1.3.3. Grobe Geometrie von Bruun

Um die Ergebnisse mit Literaturdaten abgleichen zu können wurde daraufhin die von Bruun [Bru69] verwendete Geometrie in CATIA modelliert und in OpenFOAM eingesetzt. Bruuns Geometrie besitzt im Gegensatz zur vorherigen Geometrie eine Vorkammer,

von der aus die Luft durch Schlitze in die eigentliche Wirbelkamer gelangen kann. In Abbildungen 1.4 und 1.5 ist dies zu erkennen.

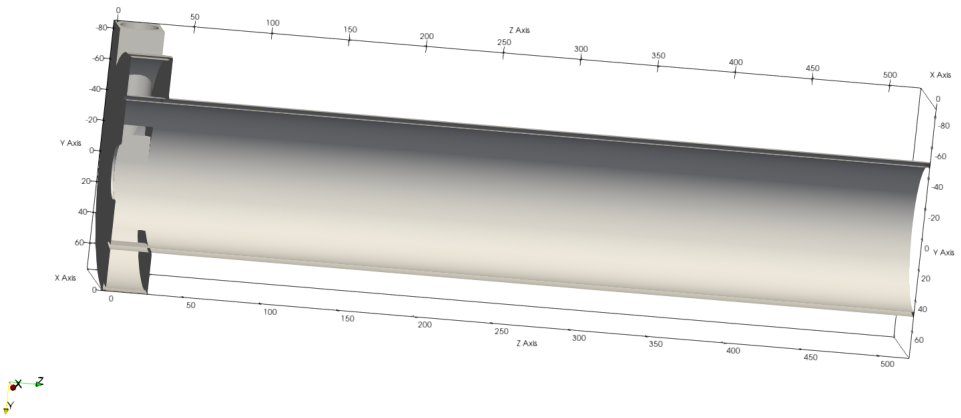

Abbildung 1.4.: Schnitt durch Bruuns Geometrie

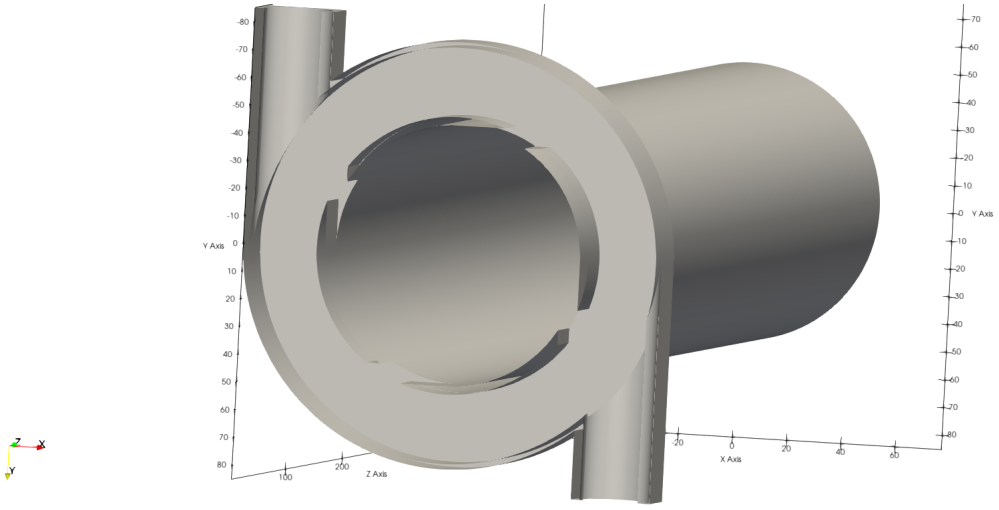

Abbildung 1.5.: Nahaufnahme der Vor- und Wirbelakmmer

1.3.4. Gekauftes Rohr

Die letzte Iteration der Geometrie ist ein in CATIA nachgebautes Wirbelrohr, welches sich im Labor C026 befindet. An diesem lassen sich die Simulationsdaten selbst verifizieren. Diese Geometrie, die in Abbildung 1.6 zu sehen ist, besitzt eine Vorkammer und einen konisch geformten Auslass an der kalten Seite des Rohrs.

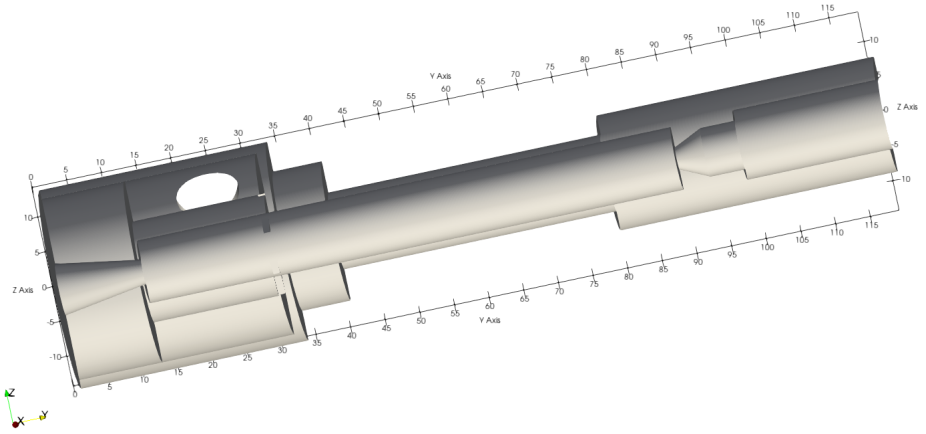

Abbildung 1.6.: Schnitt durch ein Modell des im Labor vorhandenen Rohres

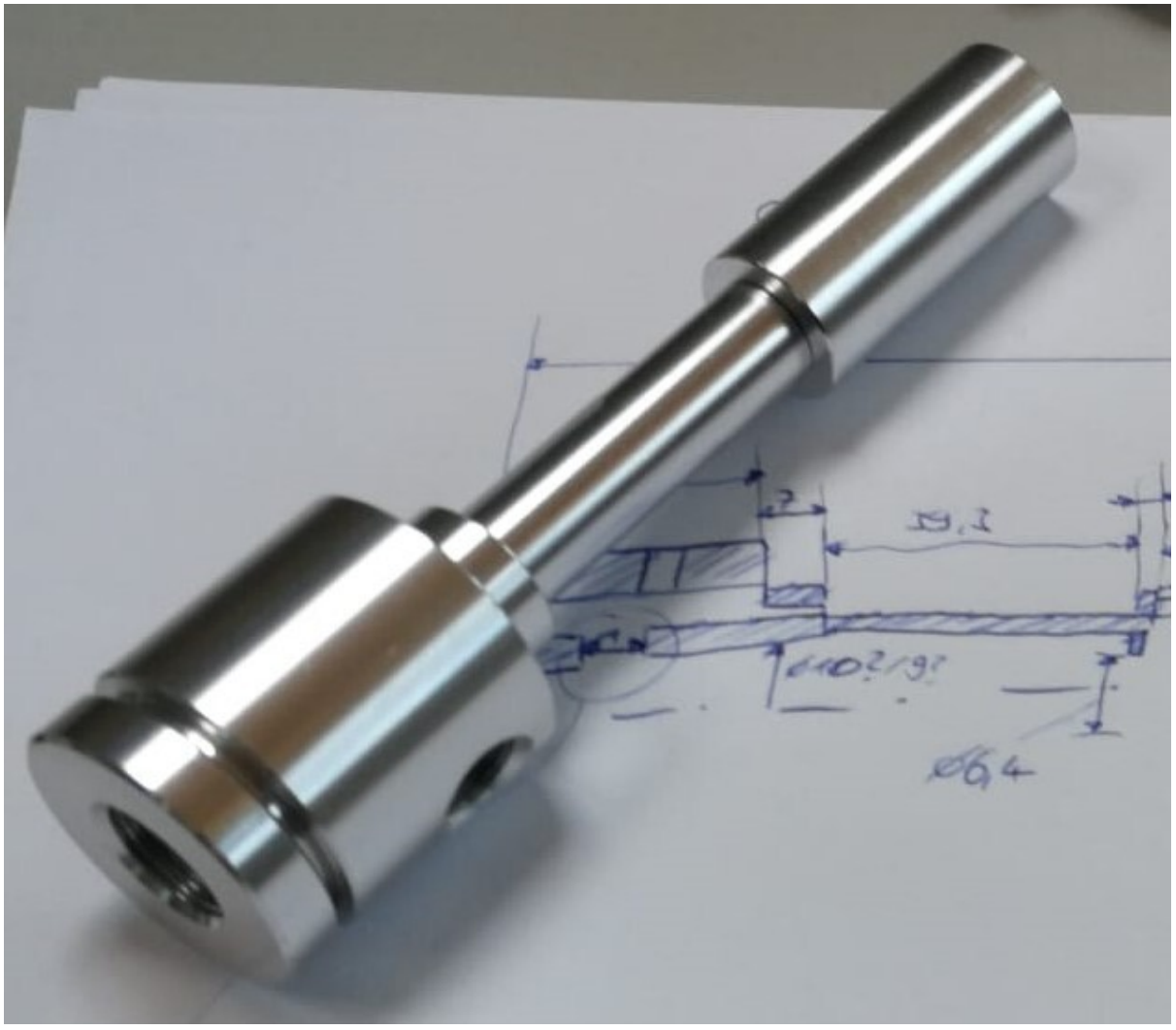

Abbildung 1.7.: Aufnahme des Rohrs im Labor

2. Simulation

Die Simulation wurde mit unterschiedlichen Solvern getestet wovon einige mehr oder weniger gut funktionierten.

2.1. Solver

2.1.1. rhoSimpleFoam

Die Simulation des Rohres wurde anfangs mit dem rhoSimpleFoam Solver vorgenommen. Dieser ist ein stationärer Solver, welcher die Kompressibilität des Mediums berücksichtigt. Die Ergebnisse waren nicht wie erhofft und brachten keine nennenswerte Temperaturdifferenz hervor.

2.1.2. rhoCentralFoam

rhoCentralFoam ist ein instationärer Solver, welcher zwar Turbulenz aber keine Unstetigkeiten, also Verdichtungsstöße darstellen kann.

2.1.3. rhoPimpleFoam

rhoPimpleFoam ist ebenfalls ein instationärer Solver, kann jedoch Verdichtungsstöße darstellen. Dem Solver fehlt jedoch die Fähigkeit Turbulenz abzubilden.

2.2. Randbedingungen

Bisher wurden als Einlassrandbedingungen Geschwindigkeit und Druck als fest gewählt. Geschwindigkeit als erste Randbedingung funktionierte nicht. Daraufhin wurde eine Druckrandbedingung implementiert. Diese funktionierte nur mit einem “Ramp-Up“, bei dem die ersten Iterationen der Simulation annähernd Umgebungsdruck am Einlass herrscht.

Die Auslassrandbedingungen waren einerseits inletOutlet für die Geschwindigkeit und eine Druckrandbedingung von Umgebungsdruck. In den letzten Versionen der Simulation wurden ebenfalls waveTransmissive Randbedingungen verwendet, wobei diese aber nie für Druck UND Geschwindigkeit angewendet wurden.

2.3. Bisheige Ergebnisse

Die bisherigen Ergebnisse sind nicht zufriedenstellend. Die Temperaturdifferenz an den Auslässen geht nach langer Simulation gegen 0 K.

2.4. Auswertung

Die Auswertung der Simulationsdaten wurde mittels eines Matlab-Skripts bewerkstelligt. Dieses findet sich im Anhang 5.1 und 5.2. Die referenzierten Funktionen zur Datengewinnung aus den .dat-Dateien lassen sich von Matlab automatisch erstellen und mit ein paar Veränderungen verwenden. Sie werden nicht aufgeführt.

3. Versuche

Mit bisherigen, einfachen Untersuchungen am Wirbelrohr wurde eine maximale Frequenz von $f_{max} = 7.6kHz$ ermittelt (siehe Abbildung 3.1). Somit ist nach Shannonschem Abtasttheorem eine Simulationsfrequenz von $2 \cdot f_{max} = 15.2kHz$ notwendig um Aliasing zu vermeiden. Das heißt der Zeitschritt muss $\Delta t = \frac{1}{2 \cdot f_{max}} = 6.58 \cdot 10^{-5} s$ was in allen bisherigen Simulationen erreicht wurde.

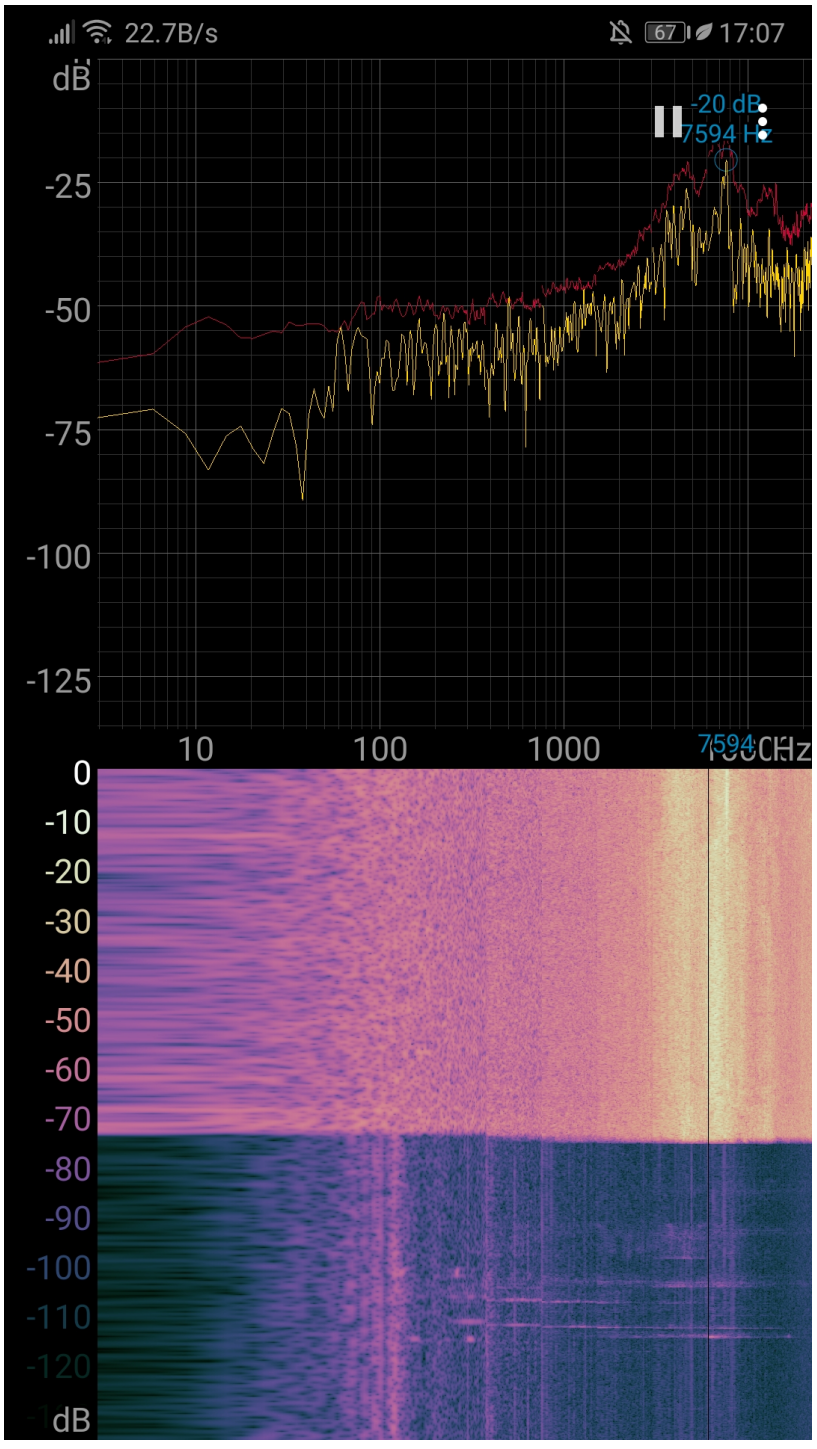

Abbildung 3.1.: Frequenz-Spike des Wirbelrohres im Betrieb

4. Probleme

4.1. Schlechte Vernetzung

Ein Problem der Simulation, welches beim Umbau häufig Kopfzerbrechen bereitet ist die schlechte Vernetzung des Rechenraumes. Kleine Durchgänge werden oft als nicht vorhanden wahrgenommen und dadurch nicht vernetzt. Die dahinterliegenden Patches werden dann von OpenFOAM nicht als existent interpretiert und führen so zu Fehlern.

4.2. Non Reflective Boundary Conditions

Durch weitere Recherche, unter Anderem in der Präsentation von F. Piscaglia [Pis], wurde ein Fehler in den bisherigen Simulationen klar. Reflektierende Randbedingungen verursachen eine stehende Welle, die Einfluss auf die Ergebnisse nehmen kann. Bisher war maximal eine der beiden Radnbedingungen Geschwindigkeit oder Druck eine nicht-reflektierende Randbedingung. Diese RAndbedingung war waveTransmissive.

4.3. Turbulenzmodellierung

Die Turbulenzmodellierung, welche in rhoCentralFoam nicht enthalten ist wurde bisher außer Acht gelassen. rhoCentralFoam liefert somit keine physikalisch korrekten Ergebnisse.

5. Weiteres Vorgehen

5.1. Versuche

In Zukunft werden in Versuchen am Rohr im Labor Temperaturen und Drücke gemessen. Die Temperaturen werden mit den Massenstromverhältnissen zwischen warmer und kalter Seite korreliert.

5.2. Simulation

Die Simulation wird künftig mit nicht-Reflektierenden Randbedingungen, also waveTransmissive, an beiden Auslässen und für Druck UND Geschwidigkeit ablaufen.

Als Solver wird in Zukunft rhoPimpleFoam verwendet werden, da bisher in der Simulation und in der Literatur keine Verdichtungsstöße auftraten und so das Turbulenzverhalten einen größeren Einfluss auf die Simulation hat.

Vorstellbar ist auch selbst nicht-reflektierende Randbedingungen für OpenFOAM zu programmieren oder den Solver abzuändern, was jedoch erhebliche Arbeit bedeutet.

Andere Solver, wie kommerzielle oder proprietäre Solver anderer akademischer Einrichtungen sind weitere Möglichkeiten andere Ergebnisse zu erzielen. An diese heranzukommen stellt jedoch die größte Schwierigkeit dar.

Literatur

[Ran34] G. Ranque. *Method And Apparatus For Obtaining From A Fluid Under Pressure Two Currents Of Fluids At Different Temperatures*. englisch. 1934.

[Bru69] H. H. Bruun. „EXPERIMENTAL INVESTIGATION OF THE ENERGY SEPARATION IN VORTEx TUBES“. englisch. In: *Journal Mechanical Engineering Science* 11.6 (1969). DOI: https://doi.org/10.1243\%2FJMES_JOUR_1969_011_070_02.

[Ton16] This Old Tony. *Building a Vortex Tube*. englisch. 2016. DOI: https://www.youtube.com/watch?v=Hn8hDY4bvpI.

[Pis] F. Piscaglia. *Develpment of NSCBC for compressible Navier-Stokes equations in OpenFOAM®: Subsonic Non-Reflecting Outflow*. englisch. DOI: http://www.engines.polimi.it.

Anhang

A. Abbildungen

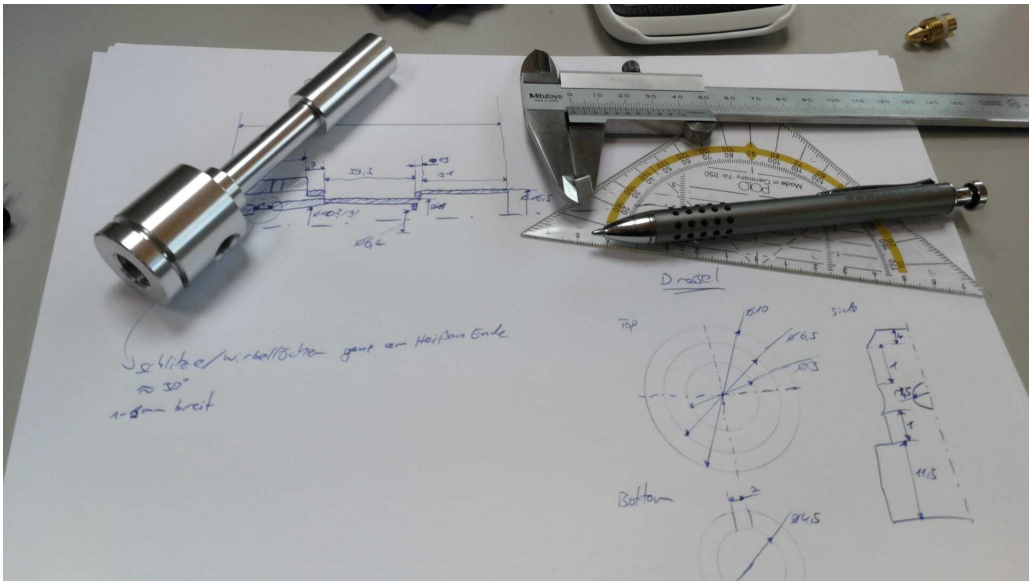

Abbildung A.1.: Im Prozess entstandenes Bild. Eventuell für Abschlussbericht verwendbar

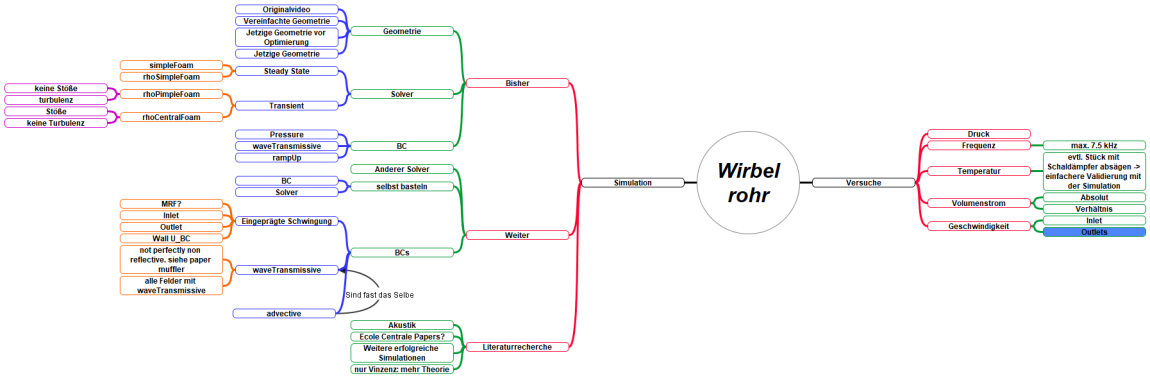

Abbildung A.2.: Mindmap über das Projekt Wirbelrohr

B. Matlab Skripte

Code 5.1: Matlab Skript zuVerifizierung der Simulation anhand von Massenströmen

```
day = input('input␣your␣day␣>>>␣␣');

path = "C:\Users\Public\openfoam\vortextuberesults\"+day+"\postProcessing
    >> \";
suffix = "\0\surfaceFieldValue.dat";

[Time1, in1] = importmass(path+"massflow_inlet1"+suffix, [5,inf]);
[Time2, in2] = importmass(path+"massflow_inlet2"+suffix, [5,inf]);
[Time3, out1] = importmass(path+"massflow_outlet"+suffix, [5,inf]);
[Time4, out2] = importmass(path+"massflow_outlet2"+suffix, [5,inf]);

[Timec, Tc] = importtemp(path+"outlet_average"+suffix, [5,inf]);
[Timew, Tw] = importtemp(path+"outlet2_average"+suffix, [5,inf]);

subplot(2,1,1)
hold on

if length(Time1) > length(Time2)
    Time1 = Time2;
else
    Time2 = Time1;
end

if length(Time3) > length(Time4)
    Time3 = Time4;
else
    Time4 = Time3;
end

[in1, in2] = matrixverlaengerung(in1, in2);
[Time1, Time2] = matrixverlaengerung(Time1, Time2);
[Time1, in1] = matrixverlaengerung(Time1, in1);

[out1, out2] = matrixverlaengerung(out1, out2);
[Time3, Time4] = matrixverlaengerung(Time3, Time4);
[Time3, out1] = matrixverlaengerung(Time3, out1);
[Time4, out2] = matrixverlaengerung(Time4, out2);

inges = -in1-in2;
outges = out1+out2;
```

```
[Time3, outges] = matrixverlaengerung(Time3, outges);
[Time1, inges] = matrixverlaengerung(Time1, inges);

plot(Time1,inges,Time3,outges,Time3,out1,Time4,out2)
legend('inlet gesamt','outlet gesamt','outlet 1','outlet 2')
title('Massenstrme')
xlabel('Zeit [s]')
ylabel('Massenstrom [kg/s]')


subplot(2,1,2)
hold on

[Tc, Tw] = matrixverlaengerung(Tc, Tw);
[Timec, Timew] = matrixverlaengerung(Timec, Timew);
delta = Tw-Tc;
[Timec, delta] = matrixverlaengerung(Timec, delta);


plot(Timec,Tc,'b',Timew,Tw,'r',Timec,delta+280*ones(length(delta),1))
legend('T cold','T warm','deltaT +280K')
title('Temperaturen')
xlabel('Zeit [s]')
ylabel('Temperatur [K]')

if input('waiting...') == 1
   return
end

[inges, outges] = matrixverlaengerung(inges, outges);
[Time1, outges] = matrixverlaengerung(Time1, outges);

subplot(1,1,1)
plot(Time1,abs((inges-outges)./inges)*100)
title('Massenstromabweichung')
xlabel('Zeit [s]')
ylabel('Abweichung [%]')
```

Code 5.2: Matlab Skript zum Vergleich verschiedener Simulationsergebnisse

```
suffix = "\0\surfaceFieldValue.dat";
caseno = input('input␣the␣casenumbers␣␣␣>>>␣␣␣␣');
linestats = ["r","b","g","m","c","k","y"];

for k = 1:length(caseno)

    path = "C:\Users\Public\openfoam\studie\"+caseno(k)+"\postProcessing
        >> \";
    [Time1, in1] = importmass(path+"massflow_inlet1"+suffix, [5,inf]);
    [Time2, in2] = importmass(path+"massflow_inlet2"+suffix, [5,inf]);
    [Time3, out1] = importmass(path+"massflow_outlet"+suffix, [5,inf]);
    [Time4, out2] = importmass(path+"massflow_outlet2"+suffix, [5,inf]);
    p = plot(Time1,abs(in1),linestats(k)+'-',Time2,abs(in2),linestats(k)+'
        >> --',Time3,out1,linestats(k)+':',Time4,out2,linestats(k)+'-.');
    for n = 1:4
       p(n).LineWidth = 1;
    end
    hold on
end


N = length(caseno)*4;
Legend=cell(N,1);
for iter=1:length(caseno)
   Legend{4*(iter-1)+1}=strcat("Inlet1 ", caseno(iter));
   Legend{4*(iter-1)+2}=strcat("Inlet2 ", caseno(iter));
   Legend{4*(iter-1)+3}=strcat("kaltes Outlet ", caseno(iter));
   Legend{4*(iter-1)+4}=strcat("warmes Outlet ", caseno(iter));
end
lgd = legend(Legend);
lgd.FontSize = 16; set(gca,'FontSize',16);
title(input('gimme␣a␣title␣>>>␣␣␣'),'FontSize',20)
xlabel(input('label␣my␣x-Axis␣>>>␣␣␣'),'FontSize',16)
ylabel(input('label␣my␣y-Axis␣>>>␣␣␣'),'FontSize',16)
axis([0 Time1(end) 0 0.19]);
yticks(0:0.01:0.19)
grid on
hold off
```

```
input('waiting...')

%%
for k = 1:length(caseno)

    path = "C:\Users\Public\openfoam\studie\"+caseno(k)+"\postProcessing
        >> \";
    [Timec, Tc] = importtemp(path+"outlet_average"+suffix, [5,inf]);
    [Timew, Tw] = importtemp(path+"outlet2_average"+suffix, [5,inf]);
    p = plot(Timew,Tw,linestats(k)+'-',Timec,Tc,linestats(k)+'--');
    for n = 1:2
       p(n).LineWidth = 1;
    end
    hold on
end


N = length(caseno)*2;
Legend=cell(N,1);
for iter=1:length(caseno)
   Legend{2*(iter-1)+1}=strcat("Temperatur warmes Outlet ", caseno(iter));
   Legend{2*(iter-1)+2}=strcat("Temperatur kaltes Outlet ", caseno(iter));
end
lgd = legend(Legend);
lgd.FontSize = 16;
set(gca,'FontSize',16)
title(input('gimme␣a␣title␣>>>␣␣␣'),'FontSize',20)
xlabel(input('label␣my␣x-Axis␣>>>␣␣␣'),'FontSize',16)
ylabel(input('label␣my␣y-Axis␣>>>␣␣␣'),'FontSize',16)
grid on
yticks(275:5:400)
lim = axis;
xlim([0 Timec(end)]);

input('waiting...')

hold off

%%
for k = 1:length(caseno)

    path = "C:\Users\Public\openfoam\studie\"+caseno(k)+"\postProcessing
```

```
      >> \";
    [Timec, Tc] = importtemp(path+"outlet_average"+suffix, [5,inf]);
    [Timew, Tw] = importtemp(path+"outlet2_average"+suffix, [5,inf]);
    Tcsurrogate = zeros(length(Tw),1);
    if length(Tc) >= length(Tcsurrogate)
        for h = 1:length(Tcsurrogate)
           Tcsurrogate(h) = Tc(h);
        end
    end
    if length(Tc) < length(Tcsurrogate)
        for h = 1:length(Tc)
           Tcsurrogate(h) = Tc(h);
        end
        for i = length(Tc):length(Tcsurrogate)
            Tcsurrogate(i) = Tc(h-1);
        end
    end

    p = plot(Timew,Tw-Tcsurrogate,linestats(k)+'-');
    for n = 1:1
       p(n).LineWidth = 1;
    end
    hold on
end

N = length(caseno);
Legend=cell(N,1);
for iter=1:length(caseno)
   Legend{(iter-1)+1}=strcat("Temperaturdifferenz zwischen den Auslssen ",
      >>  caseno(iter));
end
lgd = legend(Legend);
lgd.FontSize = 16;
set(gca,'FontSize',16)
title(input('gimme␣a␣title␣>>>␣␣␣'),'FontSize',20)
xlabel(input('label␣my␣x-Axis␣>>>␣␣␣'),'FontSize',16)
ylabel(input('label␣my␣y-Axis␣>>>␣␣␣'),'FontSize',16)
ylim([0 50])
xlim([0 Timec(end)]);
yticks(0:5:50)
grid on
hold off
```

```
input('waiting...')

hold off

%%
for k = 1:length(caseno)

    path = "C:\Users\Public\openfoam\studie\"+caseno(k)+"\postProcessing
        >> \";
    [Timec, Tc] = importtemp(path+"outlet_average"+suffix, [5,inf]);
    deltat = zeros(length(Timec)-1,1);
    for h = 2:length(Timec)
       deltat(h) = Timec(h) - Timec(h-1);
    end

    p = plot(deltat,linestats(k)+'-');
    for n = 1:1
       p(n).LineWidth = 1;
    end
    hold on
end

N = length(caseno);
Legend=cell(N,1);
for iter=1:length(caseno)
   Legend{(iter-1)+1}=strcat("delta t pro Zeitschritt ", caseno(iter));
end
lgd = legend(Legend);
lgd.FontSize = 16;
set(gca,'FontSize',16)
title(input('gimme␣a␣title␣>>>␣␣␣'),'FontSize',20)
xlabel(input('label␣my␣x-Axis␣>>>␣␣␣'),'FontSize',16)
ylabel(input('label␣my␣y-Axis␣>>>␣␣␣'),'FontSize',16)

grid on
hold off

input('end...')
hold off
clc
clear

close();
```