Institut für Thermische Strömungsmaschinen
Prof. Dr.-Ing. H.-J. Bauer, Ord.

Implementation of a Nearest-Neighbor Search for Particle Simulation in Sparse Computational Domains

Energietechnisches Praktikum
von
Jens Peifle, B.Sc.

Betreuer: **Marc Keller, M.Sc.**

Februar 2019

Ich versichere, die Arbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet zu haben. Die wörtlich oder inhaltlich übernommenen Stellen sind als solche kentlich gemacht. Die Satzung des Karlsruher Instituts für Technologie (KIT) zur Sicherung guter wissenschaftlicher Praxis wurde von mir in der aktuellen Fassung beachtet.

Karlsruhe, February 4, 2019

Contents

List of Symbols

Symbol	Unit	Description
Latin symbols		
F_{xy}		Faktor xy
H	m	Höhe
c_p	$\mathrm{J}/(\mathrm{kg\,K})$	Spezifische Wärmekapazität
Indices		
r		r-Koordinate
x		x-Koordinate
y		y-Koordinate
Abbreviations		
CFD		Computational Fluid Dynamics
ANN		Approximate Nearest Neighbor (name of C++ library)
frNN		fixed-radius nearest neighbor
CSV		Comma-Separated Value (file type)

1 Introduction

Turbofan engines are widely used in commercial aircraft. New, more stringent requirements with regards to emissions, fuel consumption and noise pollution are putting pressure on aircraft engine manufacturers to increase the efficiency of their products. From a technical point of view, there are two options: increase the thermal efficiency of the turbine stage or increase the bypass ratio of the engine. The bypass ratio is the amount of air that passes around the turbine core rather than through it. Generally, a higher bypass ratio results in a quieter, more efficient engine. To increase the bypass ratio, the diameter of the engine's fan blades must be increased. However, increasing the fan diameter leads to an increase in the circumferential velocity of the fan blade tips. At very high speeds, large increases in losses and noise emissions are observed.

In conventional turbofans, the fan and the low-pressure compressor and turbine which drives the fan are attached to single shaft. In this configuration, the speed of the compressor stage is limited by the blade tip speeds to the fan. With the addition of a gearbox between the fan and the compressor and turbine, the fan speed can be greatly reduced while the compressor and turbine can rotate much faster. Both components can therefore operate at their optimal speeds, greatly increasing efficiency and reducing noise.

However, geared turbofans are not without drawbacks. In addition to increased complexity and manufacturing cost, a significant amount of energy is lost as heat within the gearbox. The cooling and lubrication of the gearbox are key challenges. These functions are realized with oil jets arranged around the gears. The interaction between the oil jets and the gear surfaces determines the cooling and lubrication performance as well as the further propagation of the oil within the gearbox.

Therefore, this interaction is a current focus of research at the Insitute of Thermal Turbomachinery at the Karlsruhe Institute of Technology. Experimental investigations in this area are difficult due to the small time scale and inaccessible location of the interactions. However, Computational Fluid Dynamics (CFD) methods offer possibilities for detailed investigation. One such method is Smoothed Particle Hydrodynamics (SPH), a particle-based method that is well-suited to modeling free surface flows and moving boundaries. SPH, like other approaches to fluid dynamics modeling, is very computationally expensive.

It is desirable to reduce the required computation time as much as possible. For this purpose, this work examines an approach to increasing the computational efficiency of the SPH solver and therefore reducing the required computation time.

2 Motivation & Objective

2.1 Motivation

Smoothed-particle hydrodynamics is a computational method which can be used to simulate mechanics of solids and fluids. It is mesh-free and employs the Lagrangian approach, which makes it well-suited for complex problems with free surface flows and moving boundaries.

Compared to mesh-based methods, SPH requires a very large number of particles to ensure an equivalent resolution. However, in applications where there is relatively little high-density fluid (e.g. oil) in a computational space filled with low-density fluid (e.g. air), the low-density phase cam be completely omitted. This is results in a sparse computational domain, in which the amount of particles is small compared to the size of the domain.

During an SPH-method simulation, particles interact locally within a characteristic radius ("smoothing length"). In other words, each particle's behavior is influenced only by the particles surrounding it within a certain range. Therefore, for each particle p_i in the domain, all points within a certain cut-off radius r around the particle must be determined. This type of search is called *fixed-radius near neighbor* (frNN) search and is defined in section 2.2

In the in-house code currently in use at the Insitute for Thermal Turbomachinery, the fixed-radius near neighbors search is implemented using cell linked-lists (CLL, also cell lists). In sparsely-filled computational domains as seen in simulations using the SPH method, other methods for frNN search may yield a performance advantage over the CLL method, in the form of reduced run time or decreased memory use.

2.2 Problem

The fixed-radius near-neighbor search in this case is specified as follows:

For each point p_i within the computational domain, find all neighbors p_j that lie less than a cut-off radius $r = 3d_x$, away from the particle, where d_x is the mean particle spacing. This includes the particle $p_1$itself. The result is a set of interactions $p_i \leftrightarrow p_j$, where $p_j \leftrightarrow p_i$ is considered identical to $p_i \leftrightarrow p_j$ and is not repeated.

2.3 Objective

The objective of this work is to compare alternative methods for solving the described fixed-radius near neighbor search problem to the existing solution using CLL. For this purpose, a benchmarking framework is created in which the a comparison can take place independently of the rest of the SPH code. Alternative solutions are researched and the most promising methods are implemented within the framework along with the CLL method. The benchmarks measure process runtime and memory use, which are then used to compare the search methods.

3 Neighbor Search Methods

The following sections describe tools that can be used to solve the fixed-radius near neighbor search problem described in section 2.2. These are implemented in C++ 11 and outfitted with timing code in order to measure their execution times.

3.1 Cell Linked-Lists Method

The cell-linked lists method (also called linked-cell method or cell-lists method) divides the calculation domain into cells of edge lengths equal to or greater than the cutoff radius of the interaction search. Therefore, to find the neighbors of a particle, only the cells adjacent to the cell containing the particle must be searched. In the 3D case, the potential neighbors of a particle are found within the 27 cells directly surrounding the particle (**?**).

The particles themselves are first sorted into two lists, $first$ and *next*. The list $first$ contains the indices of the first particle in the cell for each cell, i.e. $first[i]$ is the first particle in the i-th cell. The list *next* links a particle i to the index of the next particle j in the same cell, i.e. $next[i] = j$. If there are no further particles in the cell, *next* contains -1. In this manner, by starting with the first particle and following the links until the next index is -1, all particles in a cell can be visited in an efficient manner.

To create particle interaction lists for the full domain, the search loops through each cell pair. A cell pair is the current cell and itself, or the current cell and an adjacent cell. For each particle in the current cell, the distance to each particle in the other cell is calculated. If the distance is smaller than the cutoff radius, the interaction is added to the interaction pair lists. For increased efficiency, there is no need to check "backwards" into previous adjacent cells, as any interactions between the current cell and the previous cell have already been found.

3.2 The *ANN* Library

The ANN library, written by David M. Mount and Sinul Arya (**??**), implements k-nearest neighbor search using methods based on orthogonal space decomposition. The particles in the search domain are spatially sorted into special data structures. ANN supports kd-trees and box-decomposition trees (bd-tress). The bd-tree includes more decomposition methods than the kd-tree and is more robust for highly clustered data sets.

ANN supports exact and approximate nearest neighbor searching and a number and a number of distance metrics. In this case, the ANN library is used to perform exact fixed-radius NN searching using the Euclidean (L_2) distance metric. Exact searches can be performed by setting the tolerance ε to 0. For fixed-radius searching, ANN provides a procedure *annkFRSearch*, which returns the k closest points that lie within the radius bound. Because ANN statically allocates its arrays, the *annFRSearch* function must be called twice to find all points within the radius: once to find the number of points within the radius, and a second time to actually find the points.

The procedure as implemented in this work is therefore as follows: First, the search tree structure is initialized using the all data points (particles) in the search domain. Then the *annkFRSearch* procedure is called twice for each data point. Initially, a search is performed using *annkFRSearch* with $k = 0$ to find the number of points k' that lie within the radius bound. Then, the appropriate arrays of size k' are allocated, and filled by calling the procedure a second time with $k = k'$. At this point, the neighbors and therefore the interactions of the current point are known. These must then be inserted into the interaction lists, making sure not to duplicate any interactions.

If there is a known upper bound on the possible number of points within the search radius, it would be possible to pre-allocate arrays that are as large as the upper bound, and therefore need to perform the search only once. It is assumed that this would come at the cost of higher memory use, but it was not tested in the scope of this work. For more detailed information about the ANN libraby, see the ANN Programming Manual (**??**).

3.3 The *nanoflann* Library

nanoflann is a header-only library for C++11 for KD-Tree structures. It does not support approximate nearest neighbors search. It includes a number of optimizations for increased efficiency. For instance, there are no provisions for choosing between or adding custom NN-search algorithms, and by using STL containers for the output data, there is no need to call the fixed-radius search method twice, as is the case with ANN. **??**.

These optimizations could lead to performance advantages of *nanoflann* over the *ANN* library. It is also able to work with dynamic point clouds without rebuilding the entire kd-tree index. For these reasons, it could be promising for the final application and is mentioned here and included in the source files. However, a comparison to the ANN library is not included in this report.

4 Project Structure and Requirements

This chapter describes the structure of the project. First the overall organization and then the individual components are explained. Following that, the prerequisites and processes for building and/or running the code used in various parts of the project are briefly described.

4.1 Requirements

This project requires a Linux system with *gcc*, *bash*, and a recent version of Python 3 with the *numpy* package to run the benchmarks. Additional Python packages are required to view and execute the analysis notebooks. *Jupyter Notebook*, *matplotlib*, *pandas*, and *seaborne* can be installed using the python package installer *pip*. Note: Jupyter requires at least Python 3.3.

4.2 Directory Structure

Figure 4.1 shows the file structure of the project. First of all, the documentation (i.e. this pdf) is in the *docs* folder, and the latex source files are in *latex*. The *ann_1.2.2* directory contains the neighbor search C++ libraries that is examined. The *nanoflann* directory is an additional neighbor search libraray (see section **??**. The folders *src* and *bin* contain the C++ source files (described in section 4.3) and their compiled binaries, respectively. The *scripts* directory is home to the test case generation scripts (described in section 4.4) and test execution scripts (see chapter 5). The *test* directory contains the working directories of the test runs and the result data. Finally, the folder *analysis* contains the iPython notebooks that were used in the examination and analysis of the results. These are described in more detail in section 4.5.

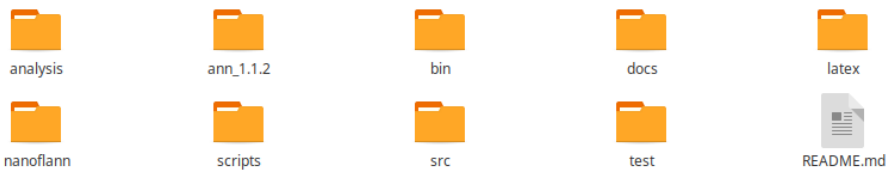

Figure 4.1: The root-level structure of the project repository.

4.3 Search Methods

The nearest neighbor search methods are implemented in C++ 11. The *src* directory contains the source files. The linked-cell method is implemented in *fr_cellLinkedList.cpp*. For the ANN method, there are multiple variants. First, *fr_ann_query.cpp* will only query a query single point for its neighbors. For the full nearest neighbor search, a method that builds the interaction pair lists (*fr_ann.cpp*) and a method that skips this step (*fr_ann_nolist.cpp*) are provided. This was necessary due to the extremely long time required for generating the interaction pair lists.

The *src* directory also contains a Makefile which can be used to build any or all of the source files. The resulting executable files are placed in the *bin* directory upon a successful build. Finally, source files for the Nanoflann library are provided (*fr_nanoflann.cpp* and *utils.h*) for possible future work evaluating this additional nearest neighbor search method. For more details about Nanoflann, see section 3.3.

4.4 Test Case Scripts

To be able to evaluate the performance of the different nearest neighbor search algorithms, a number of different test cases are required. These take the form of a list of data points which are processed by the search method executable. The test cases differ by the distribution of the data points in the search domain and are described in detail in section 5.1.

To generate the test cases, different scripts are used. Python was chosen as the scripting language due to familiarity and the availability of easy-to-use plotting libraries for visualization of the data point distributions. In the *scripts* directory, test case scripts are provided for four different distribution types examined in this work. With the exception of *test3d_full*, which simply generates a filled domain, each of the scripts take certain parameters which control the fill and spacing of the points. Note that due to the method used to generate the distributions, the fill factor passed to the script does not specify the fill directly - some experimentation is necessary to get the desired fill.

The test case scripts are meant to be run within the benchmarking framework, as they also generate a statistics file for later use in the analysis. However, the scripts can also be run separately. Copy the script and the file *config.py* anywhere and execute the test case script with the Python interpreter. The data points are written to *data.pts* and can be visualized in 2D or 3D with scripts *plot_2d.py* and *plot_3d.py*.

4.5 Jupyter Notebooks for Analysis

To examine and compare the test results, Python is again used in the form of Jupyter Notebooks found in the *analysis* directory. To access the notebooks, make sure to have the appropriate packages installed (see section 4.1). The Jupyter Notebook server can be started from the project root with *jupyternotebook* and notebooks are viewed and executed via a web browser.

The notebook files themselves are generally self explanatory. The overall structure of the analysis is as follows. In the data-preparation notebook, data is read from the results files in the *tests* directory, cleaned and labeled, and then written to a CSV file. The following notebooks read the nicely-formatted data from the CSV file and generate various tables and plots. In other words, if the data in the *tests* directory changes, the data-preparation notebook must be rerun in order to update the CSV file.

5 Test Cases & Benchmarking Methodology

This chapter first describes the test cases and shows how the data points are distributed for each case. Then, the benchmarking process and the tracked values are explained.

5.1 Test Cases

As in the SPH simulation, particles can be distributed within the domain in a variety of different ways. They can be arranged in many small clusters, or large groups. The cluster edges could be horizontal and vertical (i.e. parallel to the domain bounds) or at an angle. Depending on the contents of the simulation domain, there can be many points in the domain or only relatively few. The test cases are designed to cover these different varieties and allow for a comparison between them.

The test cases are differentiated by two important parameters, *fill* and *filltype*. The distribution and orientation of the points is determined by the fill type. Specifically, there are four different fill types: *full*, *clusters*, *corners*, and *diagonal*. Each are generated using the appropriate test case script as described in 4.4.

The second important parameter is the $fill$. This is defined as the fraction of actual points in the domain over the number of points in a fully-filled domain. The test case *full*, consisting of a fully-filled domain, has by definition a fill of 1.0. For a domain size of 0.005x0.01x0.005 and a particle spacing of 10^{-4}, the *full* test case contains 250000 points distributed on a regular grid. The number of points in the other test cases is therefore always a fraction of 250000.

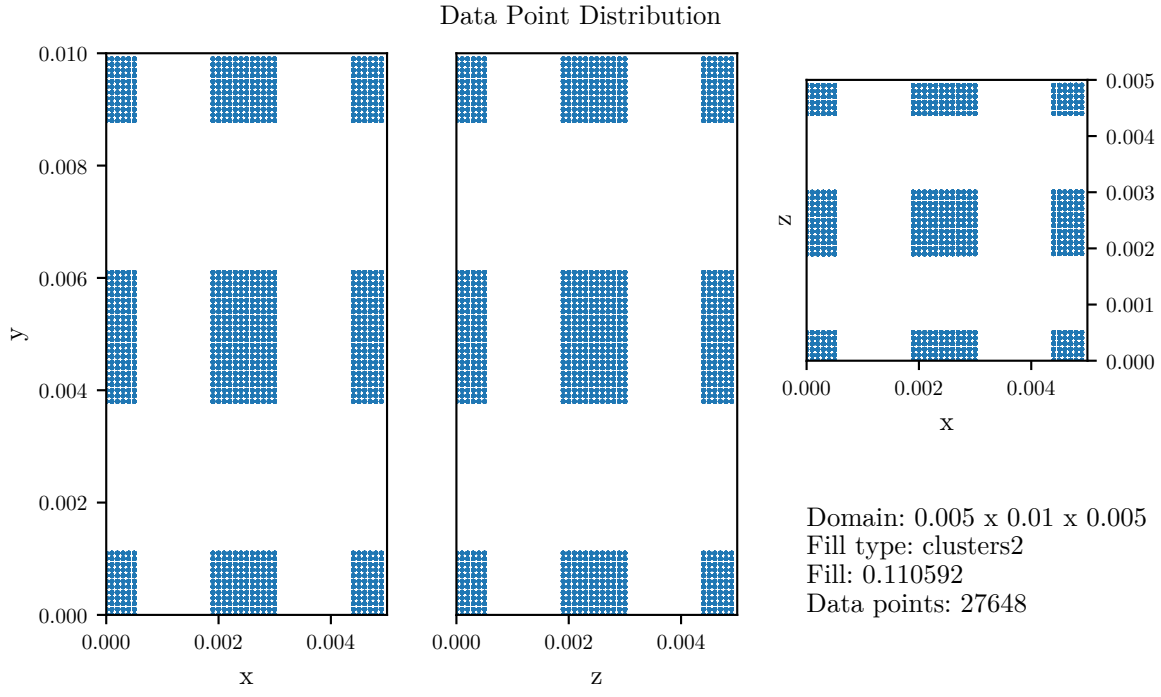

Figure 5.1: Data points distribtion for the clusters2 test case with 11% fill.

The *clusters* test case is made up of rectangular, box-shaped clusters of points. Within the clusters, the points are distributed on a regular grid. The *clusters* cases are denoted *clustersN*, where the cluster size parameter *N* specifies the number of clusters distributed throughout the space, i.e. a higher value of *N* leads to more, but smaller clusters. The *clusters* test cases are examined for fill levels of 11% and 51% with varying cluster sizes. The cluster number parameter *N* is set as an argument to the test case generation script. Exemplary *clusters* test cases for the same fill but different *N* values can be see in figures 5.1 and 5.2.

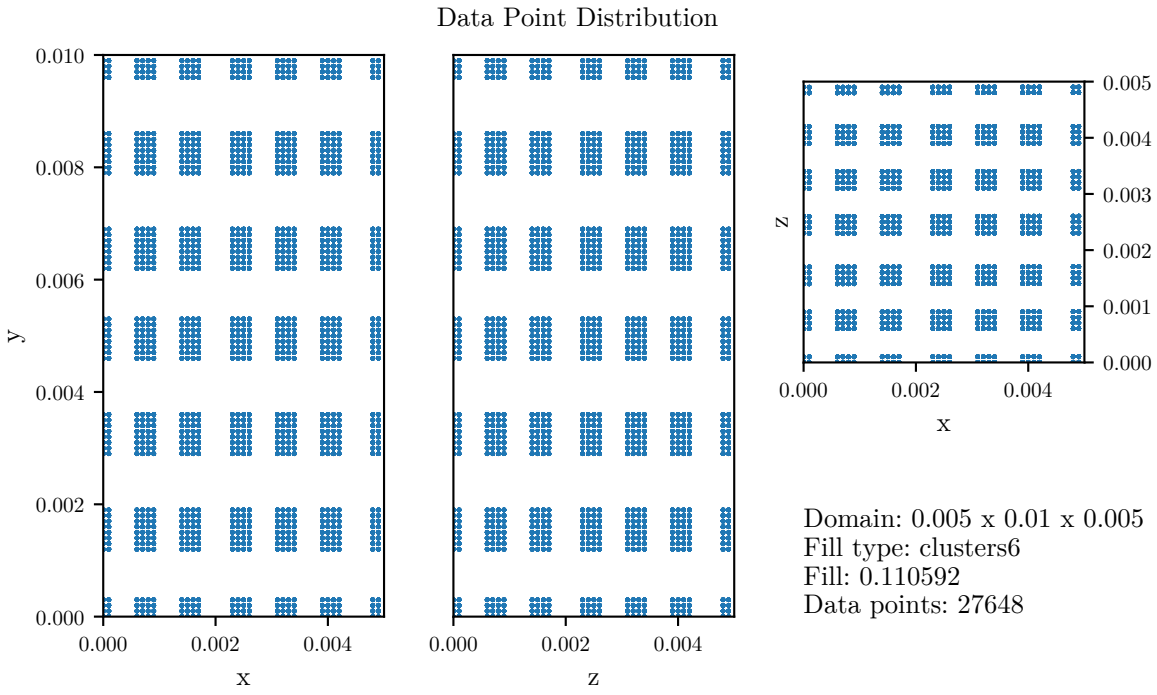

Figure 5.2: Data points distribtion for the clusters6 test case with 11% fill.

The *corners* test case represent large groups of particles. The points are distributed regularly within two rectangular boxes extending from opposite corners of the domain towards the point in the center. This is visualized in figure 5.3. The maximum possible fill with this fill type is achieved when both boxes extend from their corners to the center. In this case two quarters of the domain are filled, leading to a fill of 50%. The *corners* test case is examined for 2%, 11%, and 14% fill.

The *diagonals* test case is the only case in which the boundaries of the clusters are not parallel to the boundaries of the domain. The distribution is created by filling in the test domain from all eight corners up to a plane angled at 45 degrees from the domain boundaries. See figure 5.4 for a 2D view of the *diagonals* test case with 11% fill. This distribution is however best viewed in 3D using the *plot_3d* script.

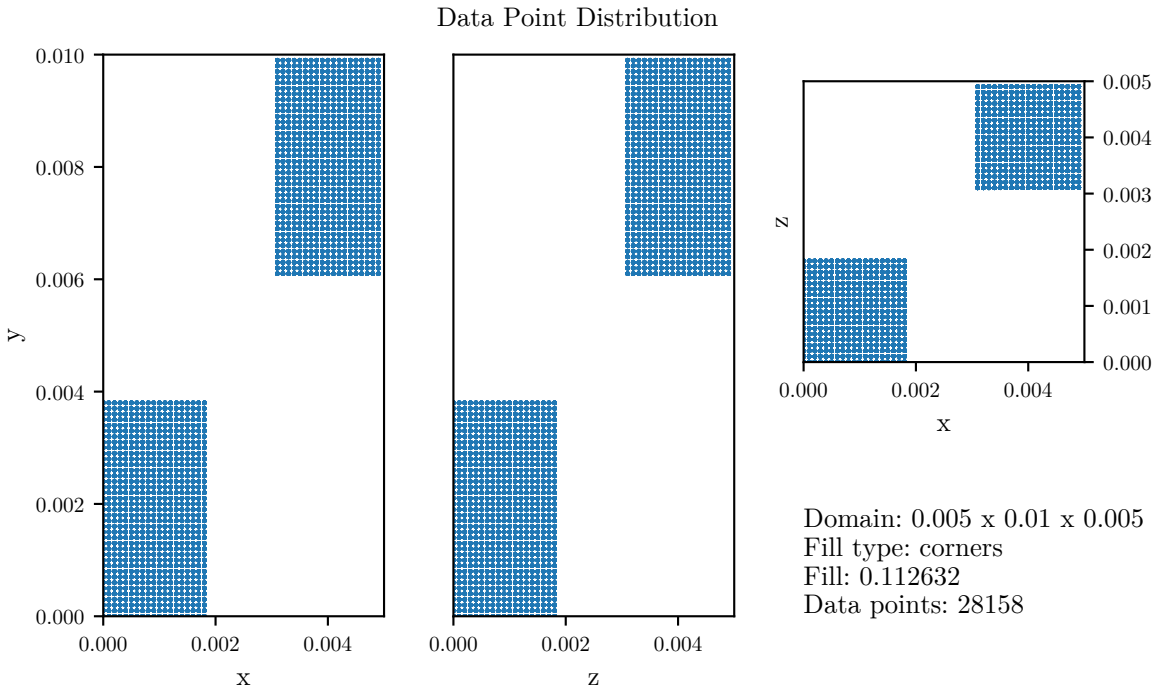

Figure 5.3: Data points distribution for the corners test case with 11% fill.

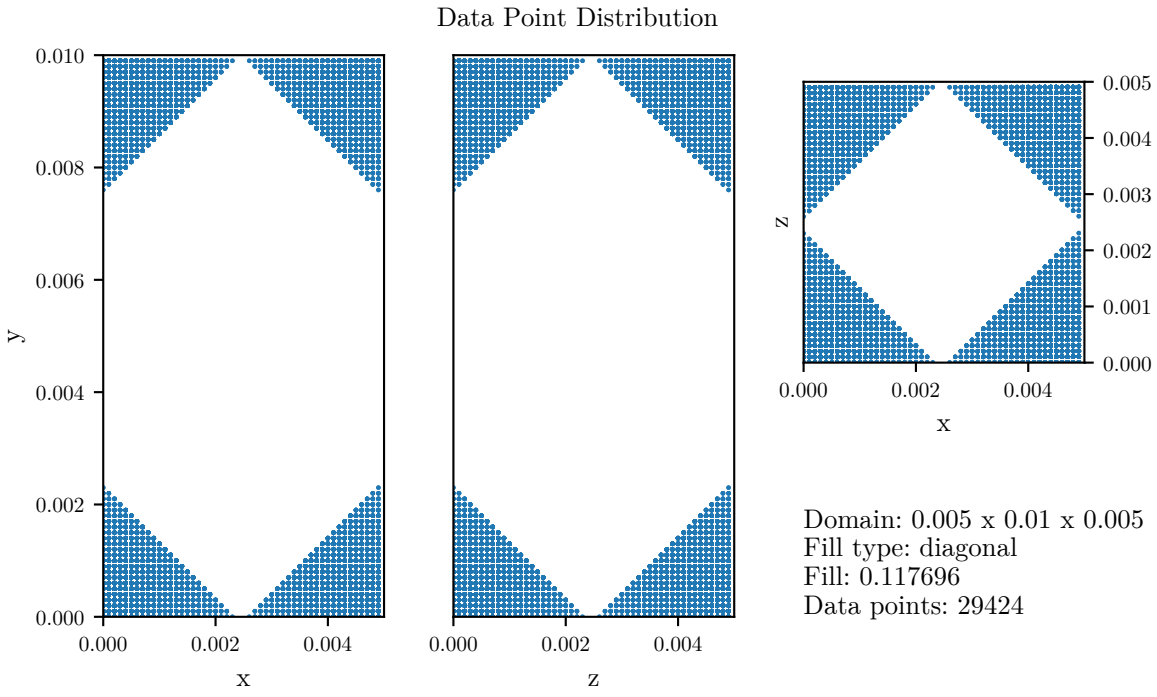

Figure 5.4: Data points distribtion for the diagonal test case.

5.2 Benchmarking

5.2.1 Tracked values

In order to compare the performance of the search methods, two characteristic values are tracked. The first is the run time of the search: how much time it takes for the program to solve the frNN search problem and compile the results into a useful data format. This is achieved by using simple timers within the C++ implementation of the search methods. For all search methods, the total time, *ttotal*, is tracked. This total time does not include reading or writing data. For the ANN search, the time is additionally split up into the time spent determining the number of neighbors, *tksearch*, the time spent retrieving the neighbors from the kdtree structure *tfrsearch*, and the time spent processing the interaction pair lists, *tprocessing*. The second tracked value is the memory use of the search method executable. This is measured by the *time* program, which outputs the maximum resident set size of the process. The resident set size is the space occupied by the process in the main memory (RAM).

5.2.2 Benchmarking Process

The benchmark is run via bash scripts for each search method, which define the test cases to run and also take care of tracking the memory use of the search processes. After the test case scripts and search method executables are copied to the test directory, the process as shown in figure 5.5 is executed for each test case.

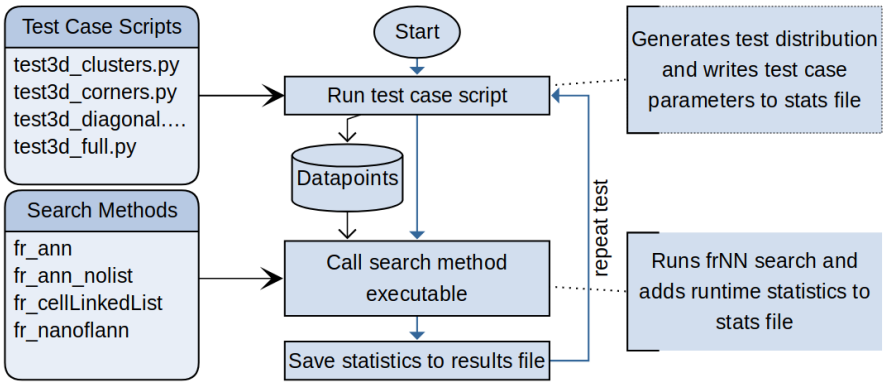

Figure 5.5: The program flow for a test run with a certain test case and search method.

First, the test case script is called, which writes the data points and statistics file. The search method executable ist then executed using the /itshape time command to track the memory use of the process. The search method executable adds its own time results to the statistics file and once it is finished the memory use results are also added. This is repeated multiple times to get a statistically valid sample. The script then moves on to the next test case, repeating the described process.

6 Results and Discussion

The results of the benchmarks were aggregated by mean over at least five repeated runs. The aggregated results can be found in table 6.1 at the end of this chapter. Take special note of the extremely long runtimes for the ANN runs which include the interaction list processing. What follows is a graphical analysis of the benchmark results. First comes an overview of the results, followed by a more detailed breakdown of the search methods into the componetns and comparison over different fill types and fill percentages.

Figure 6.1 shows the total runtime of the frNN search over various fill types. Each color represents a different a search method. The cell linked-list method (*CLL*) is in blue, the ANN library with interaction list processing in green, and the ANN library but without list processing (ANN-NoList, or *ANN-NL*) in orange. Note that there are no results for ANN for some test cases as the processing times for these cases exceeded reasonable values. The scattering in the x-axis does not have a meaning and is only to prevent overlapping of data points.

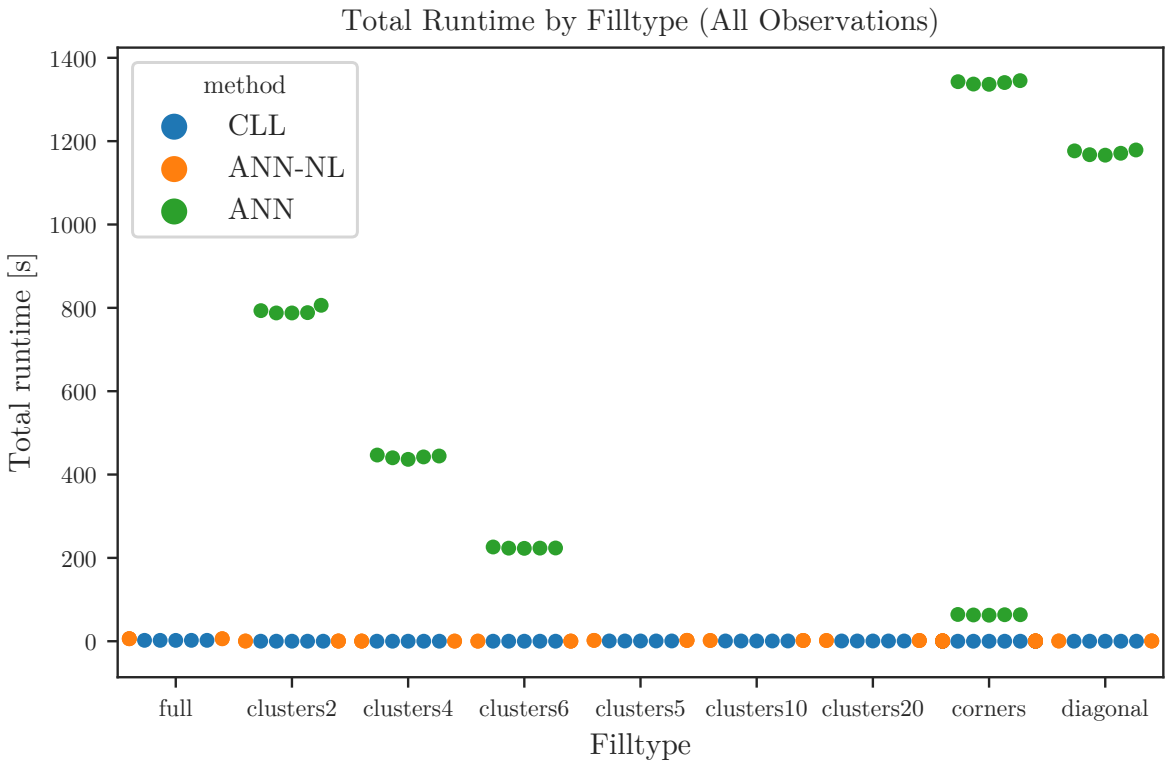

Figure 6.1: Mean total runtime grouped by fill type.

It is immediately apparent that the ANN runs take much longer than the other methods. In figure 6.2, ANN has been removed from the plot and only the CLL and ANN-NL runs are shown. With a more reasonable y-axis scale, we see that runtimes for ANN-NL are still significantly longer than CLL runtimes in almost all cases. In other words, the CLL method is faster and outputs the results in the correct format, while the ANN search itself takes longer and does not include the processing step.

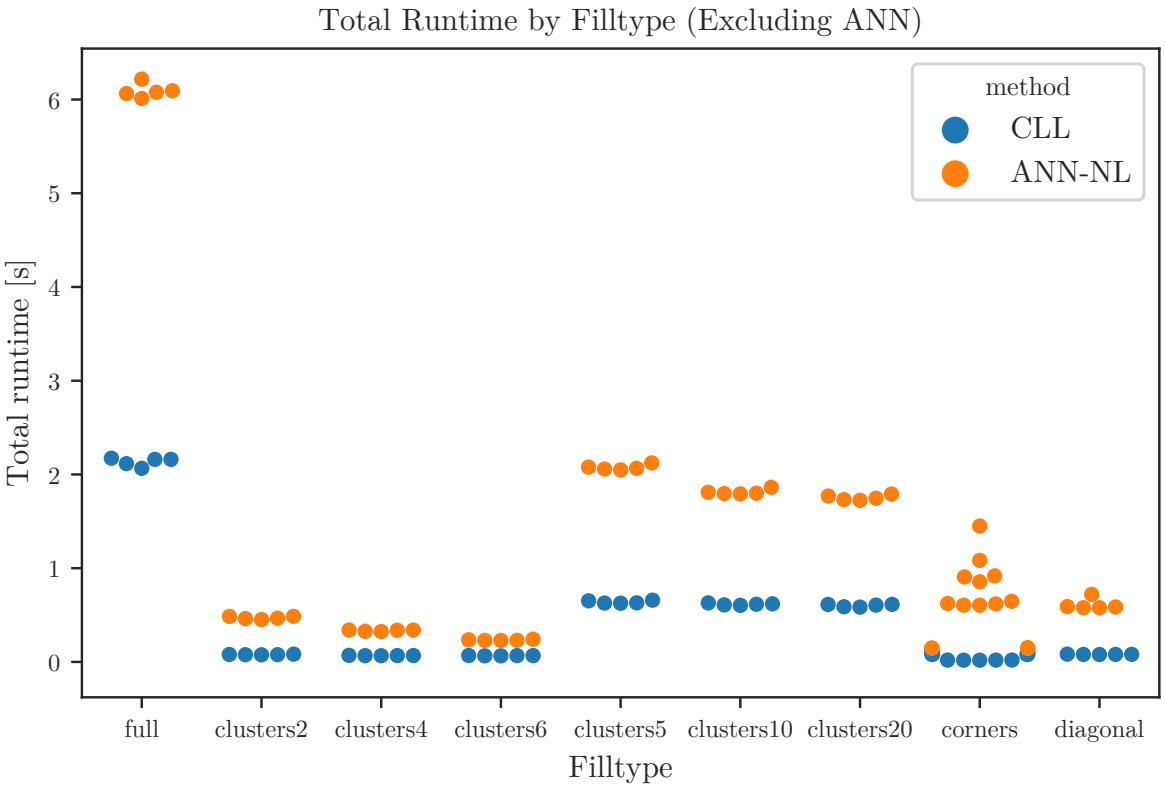

Figure 6.2: Total process runtime grouped by fill type, excluding ANN

Comparing memory use in figure 6.3, the ANN and ANN-NL methods are similar, with ANN-NL slightly below ANN. This is not surprising, as the memory requirements for the list generation are small compared to the kd-tree structure. However, the CLL method uses about one-half of the amount of memory that the ANN methods use.

In figure /refFIG:runtimeann the total runtime of the ANN search method is shown, grouped by fill type. Note that the runtimes are averaged over different fills

Table 6.1: All benchmarking results aggregated by mean.

fill	filltype	method	Datapts	Memory	$t_{frsearch}$	$t_{tksearch}$	$t_{processing}$	t_{total}
2	corners	ANN	7200	16.02	0.09	0.13	63.17	63.40
		ANN-NL	7200	13.93	0.07	0.07	0.00	0.15
		CLL	7200	8.84	0.00	0.00	0.00	0.02
11	clusters2	ANN	27648	42.46	0.32	0.53	791.79	792.70
		ANN-NL	27648	34.55	0.20	0.22	0.00	0.47
		CLL	27648	19.18	0.00	0.00	0.00	0.08
	clusters4	ANN	27648	34.30	0.22	0.41	441.36	442.04
		ANN-NL	27648	28.62	0.13	0.15	0.00	0.33
		CLL	27648	13.50	0.00	0.00	0.00	0.07
	clusters6	ANN	27648	27.53	0.14	0.26	223.46	223.92
		ANN-NL	27648	23.46	0.08	0.11	0.00	0.23
		CLL	27648	13.03	0.00	0.00	0.00	0.07
	corners	ANN	28158	51.01	0.45	0.70	1339.18	1340.39
		ANN-NL	28158	41.14	0.28	0.28	0.00	0.62
		CLL	28158	19.25	0.00	0.00	0.00	0.08
	diagonal	ANN	29424	48.75	0.41	0.67	1170.92	1172.07
		ANN-NL	29424	39.53	0.28	0.28	0.00	0.61
		CLL	29424	19.27	0.00	0.00	0.00	0.08
14	corners	ANN-NL	37044	52.37	0.48	0.47	0.00	1.04
		CLL	37044	20.88	0.00	0.00	0.00	0.10
51	clusters10	ANN-NL	128000	111.68	0.71	0.89	0.00	1.81
		CLL	128000	41.08	0.00	0.00	0.00	0.62
	clusters20	ANN-NL	128000	107.62	0.67	0.87	0.00	1.75
		CLL	128000	40.04	0.00	0.00	0.00	0.60
	clusters5	ANN-NL	128000	130.79	0.87	0.99	0.00	2.07
		CLL	128000	61.42	0.00	0.00	0.00	0.64
100	full	ANN-NL	250000	347.17	2.88	2.80	0.01	6.09
		CLL	250000	116.75	0.00	0.00	0.00	2.14

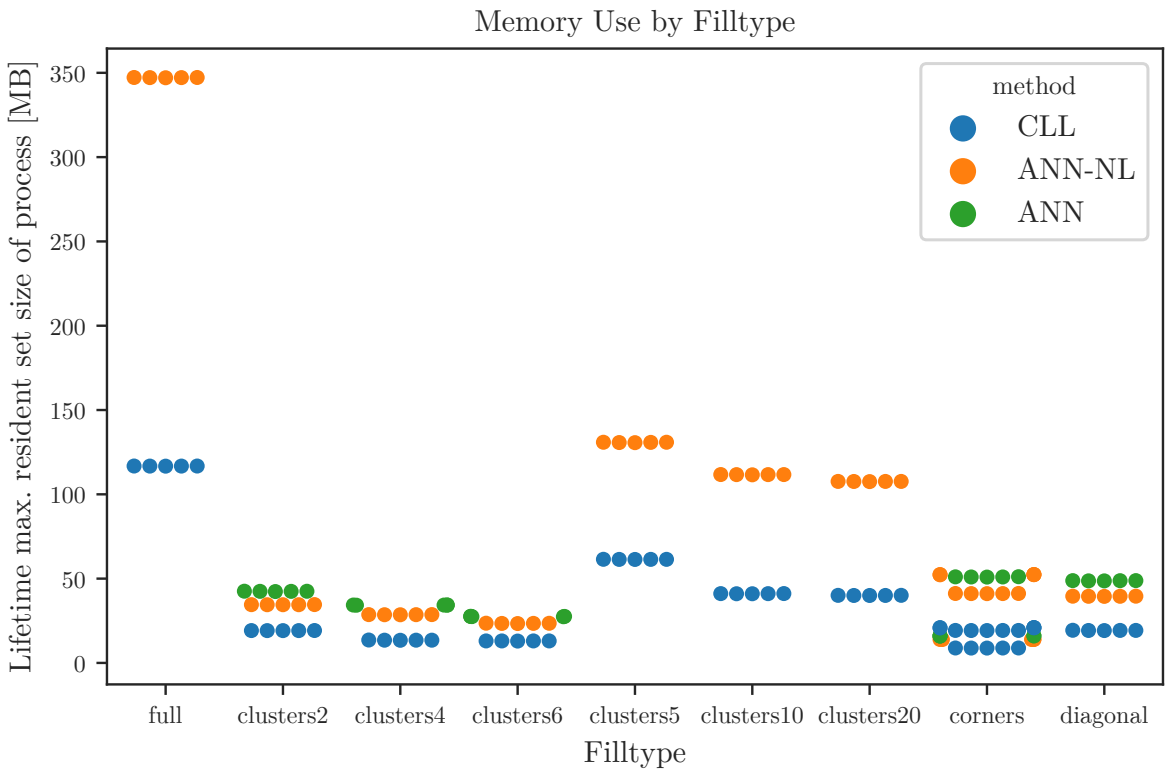

Figure 6.3: Mean maximum memory use grouped by filltype.

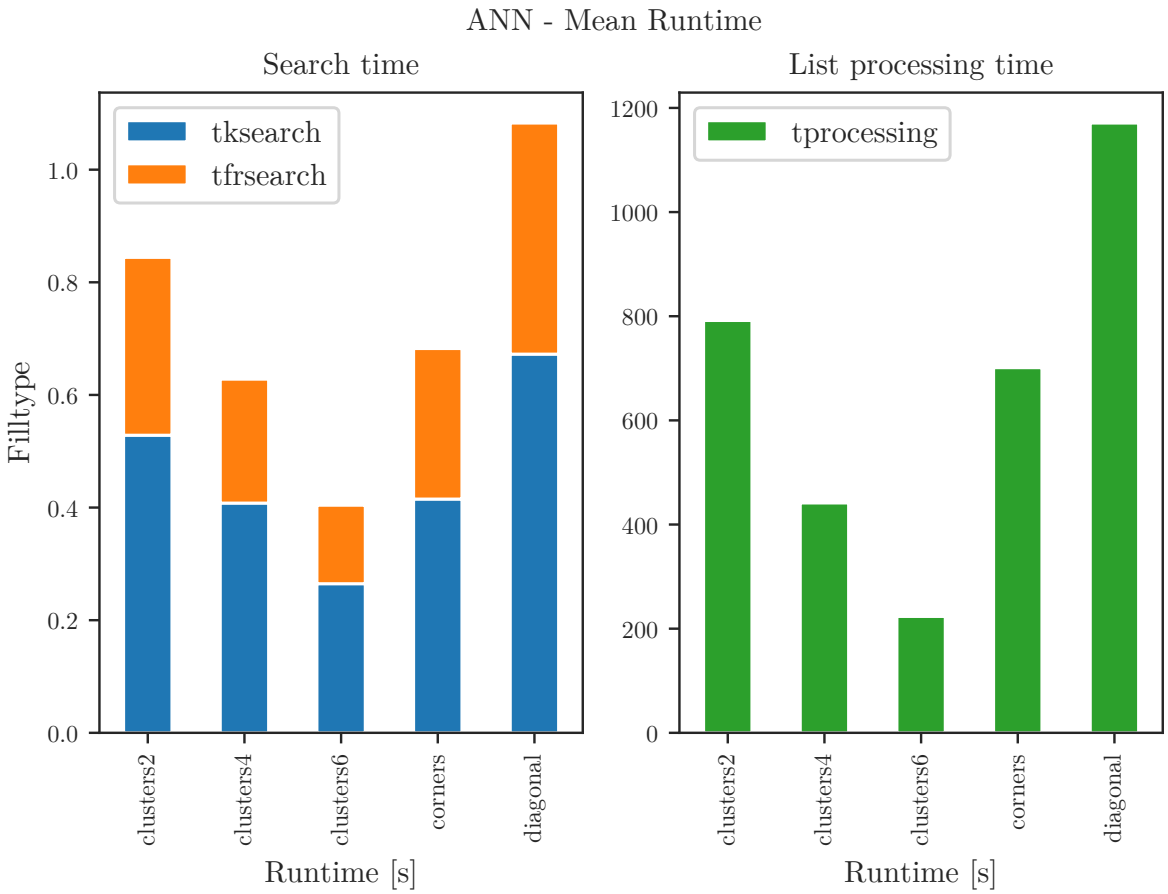

Figure 6.4: FIXME!

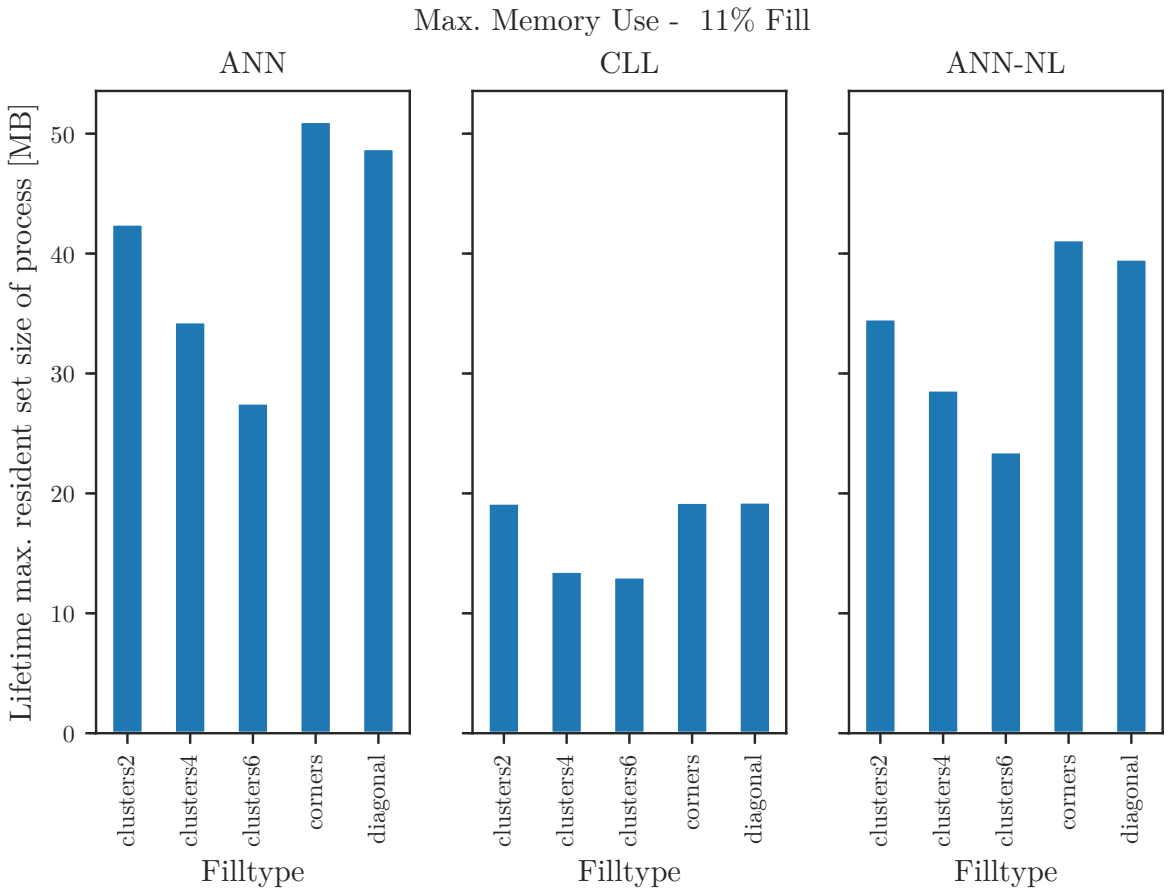

Figure 6.5: FIXME!

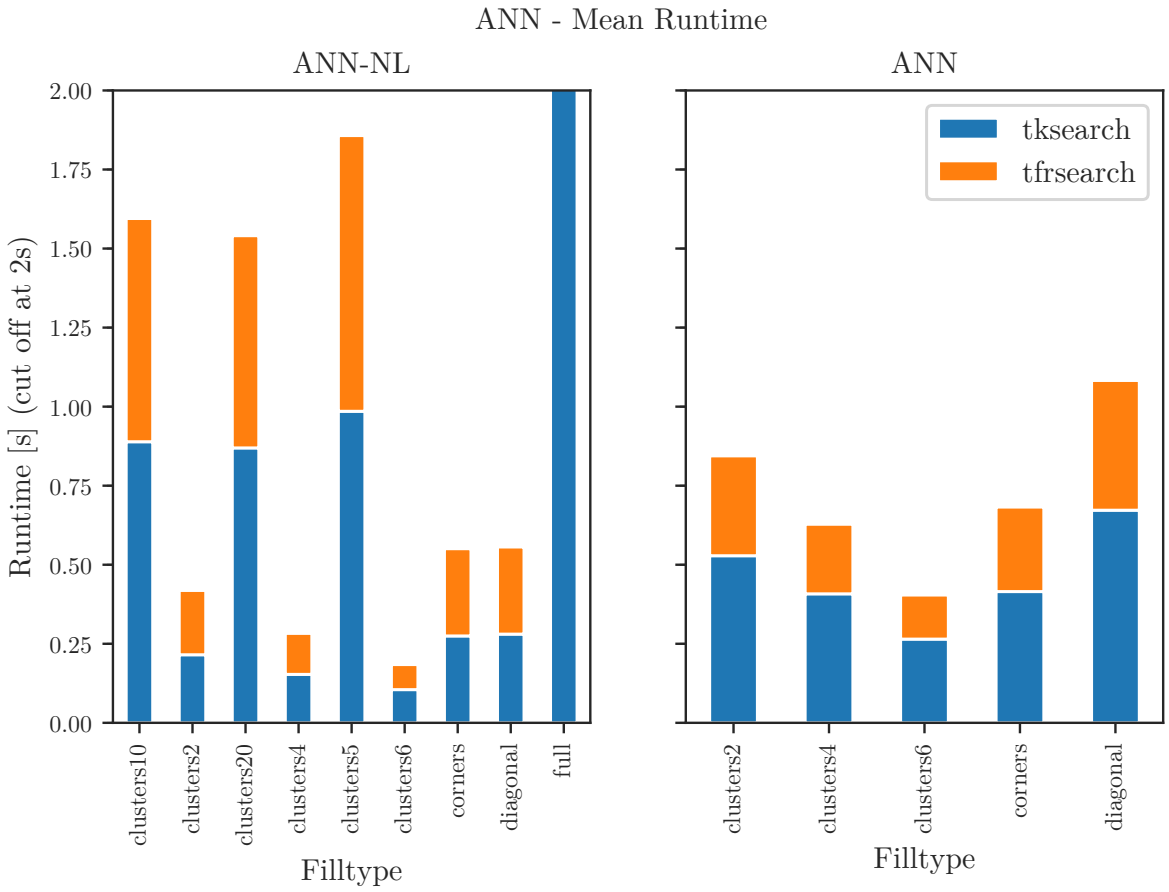

Figure 6.6: FIXME!

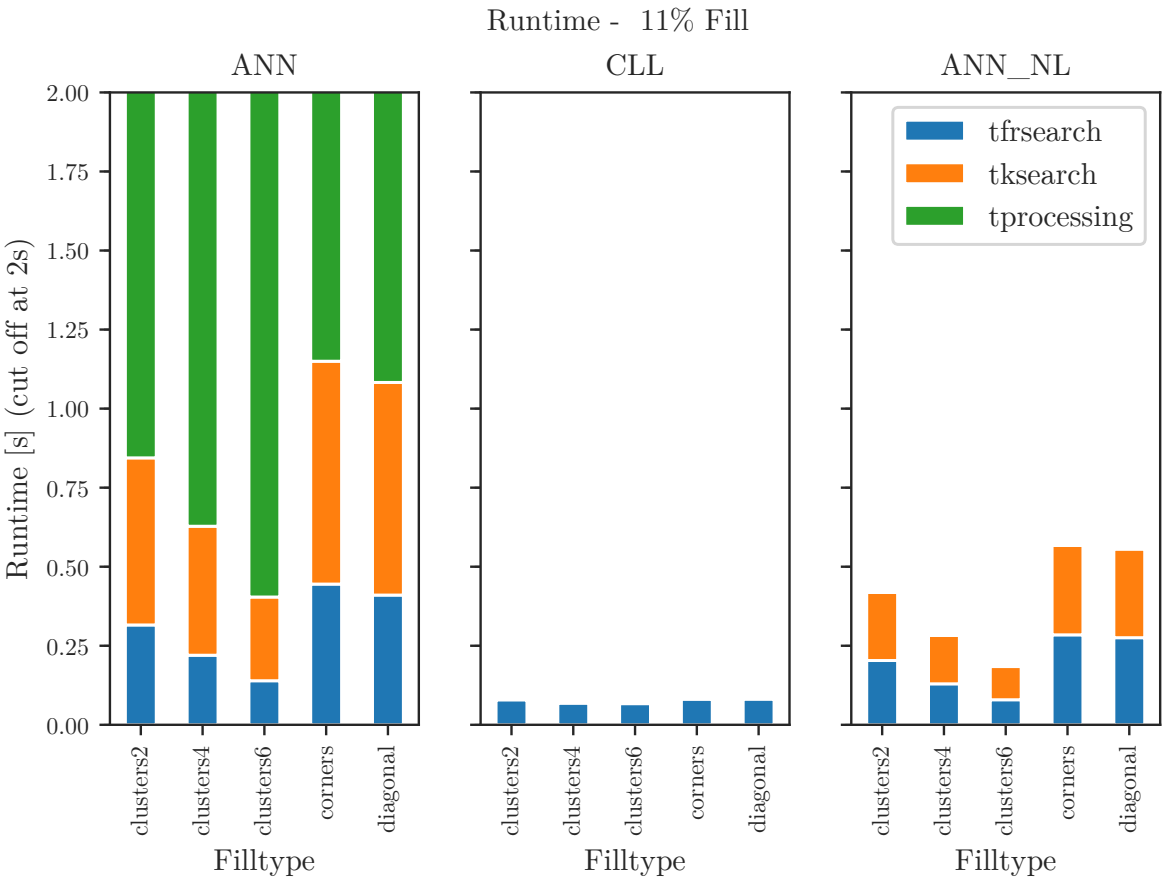

Figure 6.7: FIXME!

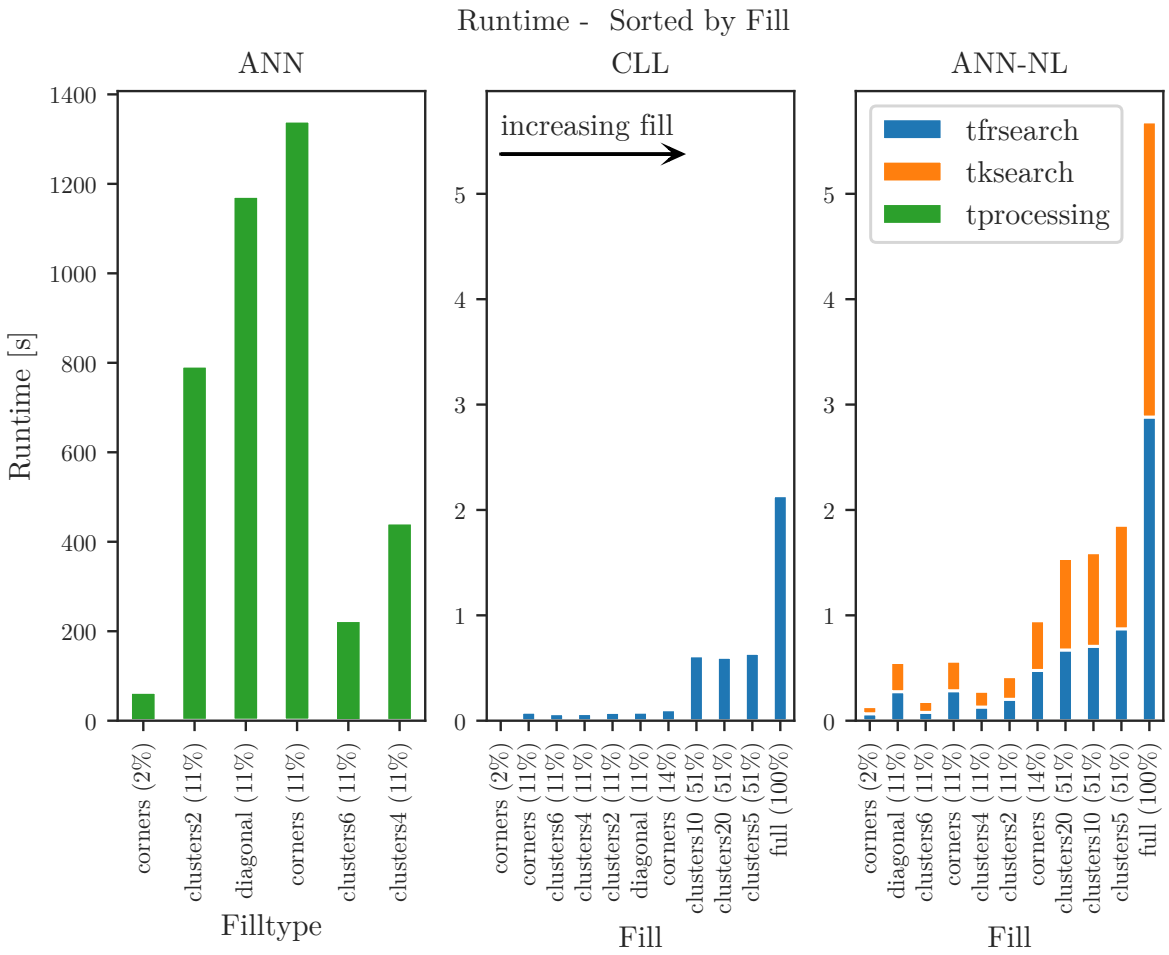

Figure 6.8: FIXME!

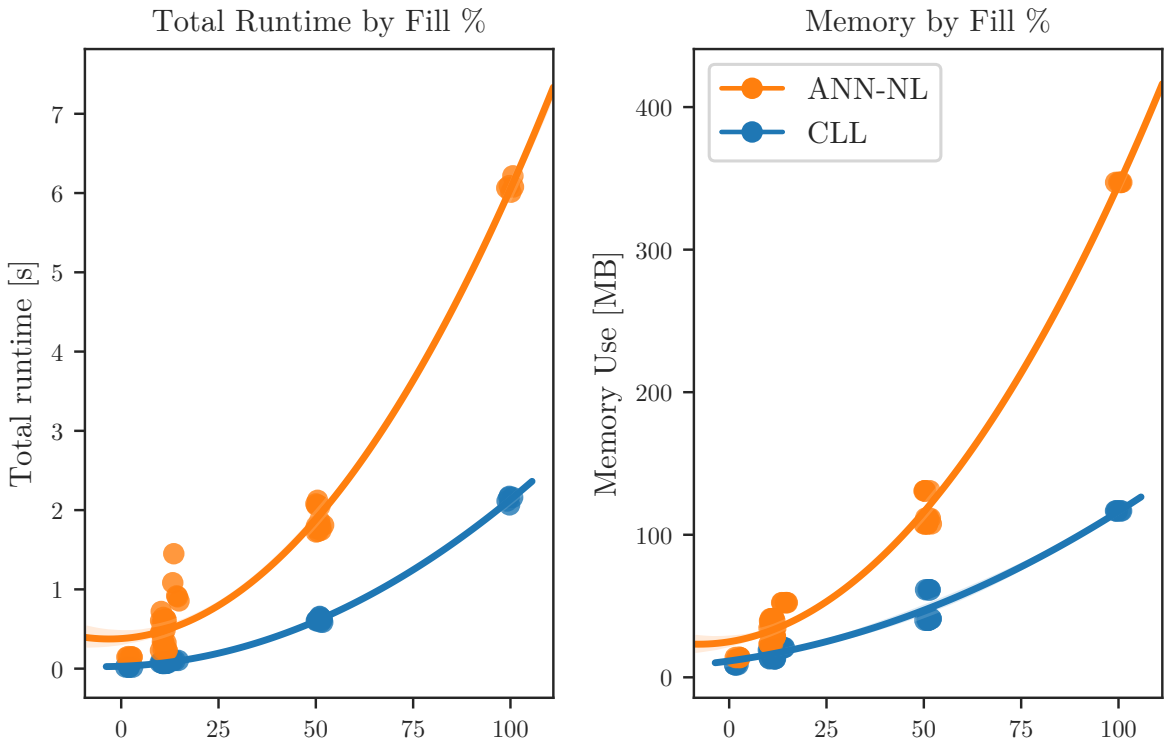

Figure 6.9: FIXME!

7 Summary and Outlook

Die Zusammenfassung ermöglicht Außenstehenden einen Überblick über den Inhalt der Arbeit und enthält insbesondere die Zielsetzung, die verwendete Methodik, Verfahren und Ansätze sowie die erzielten Ergebnisse. Daran schließt sich ein Ausblick an, der mögliche Verbesserungen bzw. weitere Arbeitsschritte aufzählt. Die Zusammenfassung ist das am häufigsten gelesene Kapitel (größt mögliche Sorgfalt!) und umfasst maximal zwei bis drei Seiten.

8 Abschlussarbeiten am ITS

Diese Vorlage dient zur Erstellung wissenschaftlicher Abschlussarbeiten am Institut für Thermische Strömungsmaschinen (ITS) und stellt eine Zusammenfassung der Meinungen aller wissenschaftlicher Mitarbeiter dar. Zu manchen Punkten kann der Betreuer eigene Meinungen haben. Es existiert eine äquivalente Word-Vorlage, deren Benutzung mit dem Betreuer abzusprechen ist.

In diesem Kapitel wird auf den allgemeinen Aufbau einer Abschlussarbeit eingegangen. In Kapitel 9 wird im Detail auf einzelne Aspekte der sprachlichen Gestaltung, Aufbau von Gliederung, Diagrammen, Abbildungen sowie Gleichungen eingegangen.

8.1 Allgemeines

Die Arbeit ist in deutscher Sprache anzufertigen und wird einseitig gedruckt. Der Gesamtumfang beträgt in der Regel für Bachelorarbeiten 30 – 60 und für Masterarbeiten 60 – 80 Seiten. Diese Angaben sind lediglich als Richtlinie zu werten und erfolgen immer in Absprache mit dem Betreuer. Vor Beginn des Schreibens ist die Gliederung mit dem Betreuer abzusprechen. Um den Korrekturaufwand möglichst gering zu halten, ist es sinnvoll, eine Vorkorrektur an einem Teil der Arbeit durchzuführen, bevor die gesamte Arbeit zur Korrektur abgegeben wird. Für die Einarbeitung in das Thema sollten ungefähr zwei Wochen eingeplant und spätestens sechs Wochen vor Abgabe mit dem Schreiben begonnen werden.

Grundsätzlicher Aufbau einer Abschlussarbeit:

- ITS Deckblatt
- Leerseite
- Gegebenenfalls ein Sperrvermerk
- Aufgabenstellung
- Unterschriebene eidesstattliche Erklärung
- Danksagung (freiwillig!)
- Evtl. Kurzfassung/Abstract
- Inhaltsverzeichnis
- Abbildungsverzeichnis
- Tabellenverzeichnis
- Symbolverzeichnis
- **Hauptteil** (siehe folgende Unterkapitel)
- Literaturverzeichnis
- Anhang

Im Folgenden ist beschrieben, aus welchen Bausteinen der Hauptteil einer Abschlussarbeit besteht, welche Aufgabe diese Bausteine haben und wie sie mit Inhalt zu füllen sind. Um den Leser an einem „roten Faden“ durch die Arbeit zu führen, ist es hilfreich kurze Zusammenfassungen am Ende der Kapitel bzw. Überleitungen zum nachfolgenden Kapitel zu liefern.

8.2 Einleitung

Die Einleitung hat die Funktion zum Thema hinzuführen und liefert Antworten, warum das Thema der Arbeit aus technischer, wissenschaftlicher und evtl. wirtschaftlicher Sicht von Bedeutung ist. Mit Ende der Einleitung muss das Thema der Arbeit klar beschrieben sein. Daher ist der Anfang sehr allgemein gehalten, während zum Schluss das Problem eingegrenzt wird. Hier gilt: Möglichst keine externe Quellen!

8.3 Stand der Forschung

Im Stand der Forschung wird aufgezeigt, welche Arbeiten es zu diesem bzw. ähnlichen Themen gibt und welche Lücke diese Arbeit füllen soll. Hierbei sind vor allem sämtliche Vorgängerarbeiten des ITS ausfindig zu machen. Als Quellen sind bevorzugt Zeitschriftenartikel und Konferenzbeiträge statt Dissertationen anzugeben. Im letzten Abschnitt des Kapitels wird eine konkrete Zielsetzung der Arbeit und die Vorgehensweise erarbeitet.

8.4 Grundlagen

Grundlagen, die zum Verständnis der weiteren Kapitel notwendig sind, werden beschrieben. Daher ist dieses Kapitel möglichst knapp zu gestalten, es handelt sich nicht um ein Lehrbuch! Um Platz zu sparen, können Verweise auf weiterführende Lehrbücher verwendet werden. Zum Beispiel:

> „Im Rahmen dieser Arbeit kommt nur das k-ε-Turbulenzmodell zum Einsatz. Deshalb wird an dieser Stelle auf andere Turbulenzmodelle nicht weiter eingegangen. Diese werden ausführlich in Autor XY et al. (2017) erläutert.“

Aussagen in diesem Kapitel werden mit Quellenangaben versehen, wenn es sich nicht um allgemeines Wissen im Ingenieurwesen handelt. Herleitungen komplexer Formeln sind nur mit Bedacht zu verwenden. Um zu verhindern, dass der Leser gezwungen ist im Text vor- und zurückzuspringen, ist auf einen logischen Aufbau des Kapitels zu achten. Dies kann zum Beispiel nach Teilgebieten, chronologisch oder nach Komplexität gestaltet sein. Verweise auf noch kommende Textstellen sind zu vermeiden. Besser: „In Abschnitt 8.2 wurde gezeigt, dass ... “.

8.5 Material und Methoden

Hier wird der experimentelle Aufbau bzw. das verwendete numerische Modell beschrieben und die Vorgehensweise der Arbeit dargelegt. Es soll kein Benutzerhandbuch zur eingesetzten Software enthalten! Die folgenden exemplarischen Gliederungen können als Orientierungshilfe dienen:

Beispiel für numerische Arbeit:

- Geometrie
- Netz
- Anfangs- und Randbedingungen
- Verwendete Solver, Zeitschrittweite, Konvergenzkriterien, ...
- Einlaufrechnung, Mitteilungsdauer, ...

Beispiel für experimentelle Arbeit:

- Versuchsaufbau und Geometrie
- Eingesetzte Messtechnik
- Angabe der Messgenauigkeit
- Vorgehensweise bei der Versuchsdurchführung

8.6 Ergebnisse und Diskussion

In diesem Kapitel werden die Ergebnisse beschrieben, diskutiert und ggf. mit anderen Daten aus der Literatur verglichen. Die Diskussion soll klären, ob die Daten sinnvoll sind, die Ergebnisse so zu erwarten waren und ob ungewöhnliche Beobachtungen sichtbar sind. Es ist auf eine übersichtliche Struktur zu achten, untersuchte Fälle sind konsistent und logisch zu benennen. Die Darstellung der Daten ist an die verwendete Messgenauigkeit anzupassen (z.B. ist eine Angabe von 314,394 575 K in den meisten Fällen nicht sinnvoll).

8.7 Zusammenfassung und Ausblick

Die Zusammenfassung ermöglicht Außenstehenden einen Überblick über den Inhalt der Arbeit und enthält insbesondere die Zielsetzung, die verwendete Methodik, Verfahren und Ansätze sowie die erzielten Ergebnisse. Daran schließt sich ein Ausblick an, der mögliche Verbesserungen bzw. weitere Arbeitsschritte aufzählt. Die Zusammenfassung ist das am häufigsten gelesene Kapitel (größt mögliche Sorgfalt!) und umfasst maximal zwei bis drei Seiten.

8.8 Anhang

Stören umfangreiche Ergebnisse oder Daten den Lesefluss, werden sie nur auszugsweise in den Kern der Arbeit übernommen. Der Rest kommt in den Anhang. Programmcode kommt nach Absprache mit dem Betreuer auch in den Anhang. Aktive Verweise auf den Anhang sind zu vermeiden. Nicht: „In Abbildung A.12 ist eine Skizze des Versuchsaufbaus dargestellt. Die Probe ist unterhalb des Messkopfes eingespannt und ... “. In diesem Fall ist die Abbildung zu wichtig für den Anhang und wird stattdessen besser im entsprechenden Kapitel platziert.

9 Weitere Hinweise

In diesem Kapitel wird auf einzelne Aspekte im Detail eingegangen. Zunächst wird die sprachliche Gestaltung behandelt, im weiteren Verlauf der Aufbau der Gliederung. Schließlich werden einige Besonderheiten zu Diagrammen, Abbildungen sowie Gleichungen erläutert.

9.1 Sprachliche Gestaltung

Das Ziel einer Abschlussarbeit ist, den untersuchten Sachverhalt möglichst einfach und leicht verständlich darzustellen. Komplizierte Satz- bzw. Wortstrukturen, die den Leser beeindrucken sollen, haben hier nichts zu suchen. Es ist sprachlich zwischen Gegenständen, beobachtbaren physikalischen Phänomenen, messbaren physikalische Größen und dem Wert der physikalischen Größen zu unterscheiden. Eine korrekte Formulierung ist beispielsweise:

> „Es findet Wärmeübertragung vom Fluid auf die Rohrleitung statt. Die Leitung ändert daher ihre Temperatur. Diese beträgt 815 °C“.

Falsch ist hingegen:

> „Die Leitung hat 815 °C“ oder „Die Wärmeübertragung wurde gemessen“ und niemals: „Die Leitung wurde gemessen“.

Ein Gegenstand hat genau einen Namen. Synonyme sind zu vermeiden, falls nicht völlig klar ist, dass mit beiden Begriffen dasselbe gemeint ist. Es sind aussagekräftige und eindeutige Beschreibungen zu verwenden. Zum Beispiel können Begriffe wie stromauf oder stromab eine eindeutigere Beschreibung liefern als horizontal oder vertikal. Auch rechts oder links sowie oben oder unten werden besser durch funktionsbezogene Lagebezeichnungen ersetzt. Beispielsweise:

> „Der Wirbel befindet sich stromab der rückspringenden Stufe im Strömungskanal. Zur Messung des Geschwindigkeitsfeldes im Wirbel wird eine Sonde quer zur Strömungsrichtung traversiert.“

Sätze, die sich über mehr als drei Zeilen erstrecken, stören den Lesefluss und werden besser in einzelne Sätze unterteilt. Zur Strukturierung von Sätzen zu Gedanken können Absätze eingesetzt werden. Sätze werden im Passiv konstruiert (kein „man“, „ich“, „wir“). Füllwörter, wie z.B. „jedoch, hierfür, ...“, sind zu vermeiden. Diese beinhalten keine Information und sind daher nicht notwendig. Lange Listen oder Aufzählungen sind ebenfalls zu vermeiden.

Eine korrekte Rechtschreibung und Interpunktion ist verpflichtend (Rechtschreibreform vom August 2006). Daher immer ein Rechtschreibprogramm nutzen! Als Dezimaltrennzeichen wird das Komma verwendet (1,3 km). Bis auf Vorgehensweise und Zusammenfassung werden alle Kapitel im Präsens verfasst.

9.2 Gliederung

9.2.1 überflüssige Überschrift

Auf einen Punkt x.y.1. folgt immer ein Punkt x.y.2. Wenn nicht, so ist der Punkt x.y.1 unnötig. Die Untergliederung auf der nächst höheren Ebene ist dann ausreichend. Empfehlung: Nach einer Kapitel- (Abschnitts-)Überschrift folgt i.d.R. Text und nicht etwa eine weitere Überschrift oder eine Abbildung. Gliedert sich das Kapitel X in X.1 und X.2, so könnte zu Beginn das Kapitels X z.B. stehen, was im Kap. X besprochen werden soll und wie sich dies auf die Unterkapitel aufteilt, warum die Themen des Kapitels X wichtig sind oder/und wie diese Themen mit dem Rest der Arbeit zusammenhängen. Die Gliederung ist so zu strukturieren, dass sie nicht mehr als drei Unterebenen (3.2.1, 3.2.1.1) enthält, vor allem nicht im Inhaltsverzeichnis. Treffende Überschriften wählen. Nicht „Messaufbau“, sondern „Laseroptische Methode zur Untersuchung des Streuverhaltens“.

9.3 Literaturverweise

Alle zur Durchführung der Arbeit und zu deren Nutzen herangezogenen Quellen sind im Text zu beschreiben oder zumindest zu erwähnen und in einem Literaturverzeichnis am Ende des Hauptdokumentes aufzulisten. Außerdem sollten sie dem Betreuer in einer geeigneten Form verfügbar gemacht werden. Ist eine Quelle nur schwierig oder kurz verfügbar (z.B. weil ein Buch nur für kurze Zeit ausgeliehen werden darf), so ist eine Kopie der für die Arbeit wesentlichen Passagen anzufertigen. Internetquellen (auch Wikipedia) sind nicht als seriöse Quelle geeignet und mit adäquaten Fachliteraturquellen zu ersetzen. Falls die Angabe im Ausnahmefall notwendig ist, wird die Quelle in Form einer html-Datei oder eines Screenshots festgehalten. Direkte Zitate der Form:

> „In modernen Hochleistungstriebwerken betragen die Kühlluftmassenströme bereits über 30% des gesamten Massenstroms der Kernmaschine.“ (**?**),

sind zu vermeiden.

9.4 Abbildungen und Tabellen

Abbildungen werden im Text erläutert: Abbildung 9.1 lassen sich die physikalischen und geometrischen Eigenschaften des berechneten Gebiets entnehmen. Daher ist die Abbildung so platziert, dass sie nach dem Text erscheint, in dem diese beschrieben werden. Ebenso wird die Aussage der Abbildung erläutert: Die Fluid begrenzenden Flächen sind blau dargestellt, ... Passiv konstruieren: Nicht „Das Bild 2.4 zeigt“, sondern „In Bild 2.4 ist gezeigt“.

Abbildungen helfen dem Autor, seine Erkenntnisse zu erläutern oder zu belegen. Diese Verbindung zwischen Abbildung und Aussage muss deutlich werden, sonst ist die Abbildung zu streichen. Alle Grafiken, Abbildungen und Tabellen müssen im Text referenziert und besprochen werden.

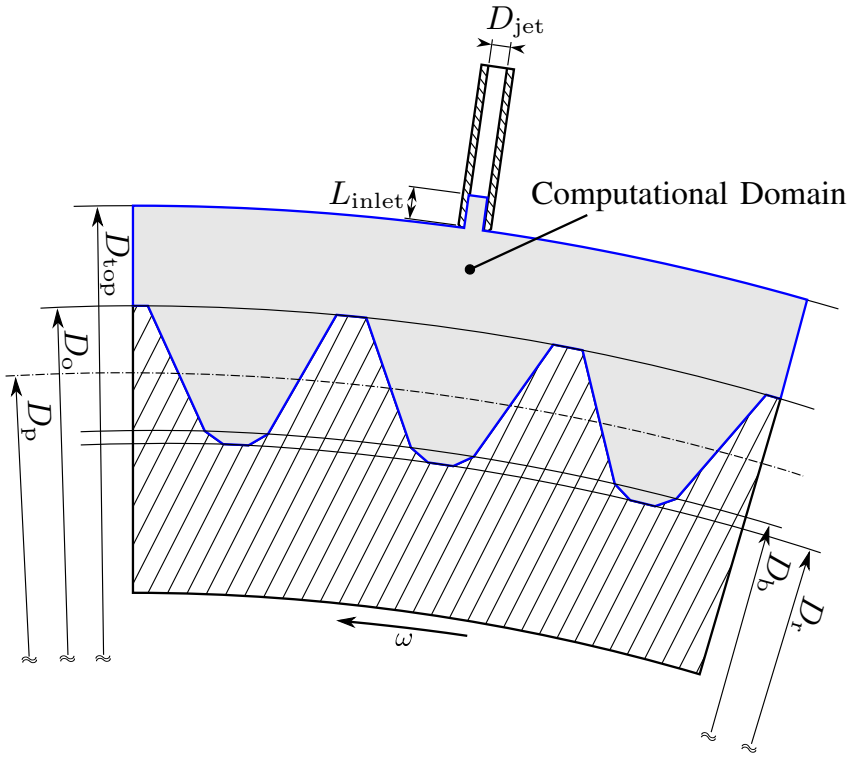

Figure 9.1: Querschnitt durch das berechnete Segment des Zahnrads, **?**

Die Referenz ist durch eine fortlaufende Nummerierung der Objekte eindeutig vorzunehmen. Dabei werden Objekte gleichen Typs wie z.B. Abbildungen, Gleichungen und Tabellen jeweils gesondert nummeriert. Auf ein einheitliches Erscheinungsbild und Lesbarkeit achten (Textgröße, Auflösung, Größe angepasst an DIN A4, Vektorgrafiken wenn möglich). Abbildungen, die nicht selbst erstellt wurden, müssen ebenfalls zitiert werden. Wenn sie verändert wurden: „Abbildung nach **?**“. Im Gegensatz zu Abbildungen besitzen Tabellen keine Unter- sondern Überschriften (siehe Tabelle **??**). Die Unterschriften sind möglichst aussagekräftig und eindeutig zu gestalten. Weitere Tabellenbeispiele werden von Markus Püschel[1] aufgezeigt.

[1] `https://www.inf.ethz.ch/personal/markusp/teaching/guides/guide-tables.pdf`

9.5 Gleichungen

Alle Formelzeichen, die in einer Gleichung auftreten und nicht schon im vorangehenden Text definiert wurden, sind zu benennen und zu erläutern. In der Literatur etablierte Formelzeichen zur Beschreibung von physikalischen Größen wählen. Ein Formelzeichen beschreibt genau eine physikalische Größe. Jede physikalische Größe wird mit genau einem Formelzeichen beschrieben. Der Übersichtlichkeit halber mit Indizes und eingängigen Namen arbeiten. Gleichungen werden zentriert, nummeriert und sind Teil eines Satzes (besitzen also Satzzeichen). Zum Beispiel beschreibt der Impulssatz der Strömungsmechanik,

$$\sum \vec{F} = \frac{d\vec{I}}{dt} = \underbrace{\frac{\partial}{\partial t} \iiint \rho \, \vec{w} \, dV}_{\text{instationär}} + \underbrace{\iint \rho \, \vec{w} \, (\vec{w} \cdot \vec{n}) \, dA}_{\text{stationär}}, \tag{9.1}$$

die zeitliche Änderung des Impulses $\frac{d\vec{I}}{dt}$ innerhalb eines Kontrollvolumens V, mit der Dichte ρ, dem Geschwindigkeitsvektor $\vec{w}$, der Oberfläche A und dem äußeren Oberflächennormalen-Einheitsvektor $\vec{n}$. Die Änderung des Impulses entspricht der Summe der äußeren Kräfte $\sum \vec{F}$.

Formelzeichen werden immer zusammen mit einer Bezeichnung erwähnt. Die Angabe des Werts einer physikalischen Größe erfolgt immer mit der entsprechenden Einheit:

$\Delta T = 452{,}69\,\mathrm{K}$.

Zwischen Größe und Einheit, wie 2,0 mm, wird ein halbes Leerzeichen gesetzt (umbruchgeschütztes Leerzeichen verwenden bzw. LaTeX-Paket siunitx). Einheiten und Operatoren, wie „log“, werden nicht kursiv gesetzt. Innerhalb der Arbeit auf eine einheitliche Darstellung für Vektoren, Tensoren etc. achten und SI-Einheiten verwenden.

Appendix

A.1 Anhang 1

Table A.1: Simulationsergebnisse, die den Lesefluss stören.

	$w = 8$			$w = 16$		
	$t = 0$	$t = 1$	$t = 2$	$t = 0$	$t = 1$	$t = 2$
$dir = 1$						
c	0,0790	0,1692	0,2945	0,3670	0,7187	3,1815
c	-0,8651	50,0476	5,9384	-9,0714	297,0923	46,2143
c	124,2756	-50,9612	-14,2721	128,2265	-630,5455	-381,0930
$dir = 0$						
c	0,0357	1,2473	0,2119	0,3593	-0,2755	2,1764
c	-17,9048	-37,1111	8,8591	-30,7381	-9,5952	-3,0000
c	105,5518	232,1160	-94,7351	100,2497	141,2778	-259,7326

A.2 Anhang 2

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adip-

iscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.