Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Student Research Project

Moritz Burmester

Streamlining Aircraft CFD Analysis with Automated Parametric Geometry and Mesh Generation

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Moritz Burmester

Streamlining Aircraft CFD Analysis with Automated Parametric Geometry and Mesh Generation

Studienarbeit eingereicht im Rahmen des Bachelorstudiums

im Studiengang Flugzeugbau
am Department Fahrzeugtechnik und Flugzeugbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Erstprüfer/in: Prof. Dr. Detlef Schulze
Abgabedatum: Datum

Abstract

Moritz Burmester

Keywords

AeroFlow, Computational Fluid Dynamics, Ansys Fluent, CPACS, TiGL, Matlab, AMR, CAD, STEP, GUI

Abstract

In the field of computational fluid dynamics, preprocessing work can be extremely labor-intensive. This work involves reading geometric parameters, building CAD models, generating volume meshes, and refining the mesh to provide accurate results without requiring extensive computational effort. This process can be time-consuming and requires significant technical expertise. The work presented in this document provides a solution to the problem by automating the preprocessing work with minimal user intervention. This toolbox is called AeroFlow. The software reads the geometric parameters of an aircraft and automatically builds a CAD model, generates the volume meshes, and solves the flow field while using adaptive mesh refinement. This reduces the time and technical expertise required for preprocessing and allows for faster setup of simulations. The ability to easily adjust the critical parameters provides additional convenience and speed in the simulation setup process. Overall, AeroFlow greatly improves the efficiency of the preprocessing work in aeronautical engineering, reducing the time and resources required to set up simulations allowing for faster and more accurate results.

Contents

List of Figures

List of Abbreviations

AMR	Adaptive Mesh Refinement
CAD	Computer Aided Design
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
CPACS	Common Parametric Aircraft Configuration Schema
GUI	Graphical User Interface
UAV	Unmanned Aireal Vehicle

Chapter 1

Introduction

1.0.1 Motivation

Accurate and reliable prediction of aerodynamic behavior is a fundamental part in aircraft development. While low fidelity models using Vortex Lattice Methods for example can produce many datapoints in a short timeframe, their accuracy is limited by their simplifications. Computational Fluid Dynamics however are time consuming, but can provide more accurate solutions and further insight into flow phenomena. With increasing computational power, such high fidelity methods can be used earlier in the design phase than ever before. The most time consuming processes today are the steps requiring human intervention. Traditionally the preprocessing is done manually for each and every simulation to be run. Automating these steps enables engineers to get results more quickly and to better explore an entire parameter space early in the design process.

1.0.2 Objectives

The main objective of the AeroFlow toolbox is to automate preprocessing steps as good as possible. The goal is to have a digital wind tunnel with a sensible basic setup and minimal user input required. Real world application requires that the basic setup can be modified to the desired operating conditions easily. To enable a further layer of customizability, the toolkit shall avoid the use of proprietary software interconnection. This has three reasons:

1. Proprietary connections limit the set of parameters that can be accessed by the tool. (As in the case of the Ansys Workbench)

2. The underlying software providing functionality such as mesh generation can be replaced more easily. The currently implemented commercial software can be changed to opensource codes for example.

3. Standardization and Extensibiltiy: Proprietary connections may be elegant for certain tasks but require specific skills by the user. A script based scheme throughout the toolkit makes it more accessible and coherent. This will be increasingly relevant if capabilities are to be added using additional software.

The initial version of this work is not intended to provide an optimized simulation setup for low subsonic flight. Nevertheless the provided basecase shall resemble a useful starting point for this kind of flow.

Chapter 2

State of the Art

2.1 Challenges in Preprocessing

External aerodynamic analysis often require a comparison of multiple similar geometries and operating conditions. Therefore, this section will briefly introduce the challenges specific to a study of this nature. The general principles and problems of CFD lie beyond the scope of this document.

2.1.1 Recurring Tasks

Throughout all preprocessing steps most of the tasks will be exactly the same for each simulation. Once a base case has been created, a parametric study will most often require only one single parameter change. If the preprocessing and parameter change are not automated, the user will spend most of his time waiting for his workstation to finish tasks or load graphical user interfaces.

2.1.2 Geometry Creation

The geometry creation must be reliable, consistent and of appropriate complexity. In an automated workflow it is important to produce error free CAD geometry every time. Otherways there is not only the risk of wasting CPU time on wrong input geometry, but also of working with misleading result data, should the error not be visible to the user. A further common pitfall is an inconsistent naming scheme for geometry elements. Most meshing software applies cell refinements based on named surface patches or bodies. In automated studies it is therefore important to stick to a naming scheme for consistent (initial) mesh quality. To avoid bad mesh quality later on and to limit the required cell count, geometry should only include relevant geometric features. While cleaning up production CAD models is common practice for single simulation runs, studies on various parametric geometries require an approach tailored to this task.

2.1.3 Meshing

Meshing is the most critical phase of preprocessing, as mesh quality is largely responsible for the quality of the final results. It is usually an iterative process starting from a coarse mesh.

Geometry representation

The first step is to produce a mesh representing the input geometry reasonably well. Spatial discretization has to be refined in areas of small geometric features and high curvature

such as leading and trailing edges of aircraft wings. This will help achieve cell quality requirements and proper representation of the flow domains geometry.

Resolution of Flow Features

While resolution of the geometry surfaces can be visually judged before an initial simulation has been run on the mesh, the resolution of flow features in the domain is more complicated. Depending on geometry and angle of attack for example, volume mesh refinements can be applied based on experience. This usually involves defining geometric primitives as areas for a specified mesh refinement. This however will lead to an unnecessarily high cell count, or increasingly complex refinement setups. The issue is further complicated when a large set of geometries and angles of attack is to be analysed. Each of these simulation runs pose different requirements on the mesh. A traditional approach would imply running a simulation on the initial mesh and manually refining areas of high field gradients. This is a labour intensive process. One possible solution is the use of adaptive mesh refinement. This will incorporate a coarse base mesh at the start of the solving phase. Depending on the developing flow field, cells matching a predefined criterion will be marked for refinement. As the solution progresses, the mesh will improve in quality. This way, solving time can be drastically reduced while also reducing manual work after initial setup. For more information on AMR, the reader is referred to: [3] and [8]

2.2 Literature Overview

For general guidelines on producing mesh for low subsonic flow, [3] provides useful advice. Proposed solutions to the challenge of automated proprocessing can be found in [4] and [10]. AeroFlow is heavily influenced by these implementations. The following figure shows the overall workflow as implemented by DLR.

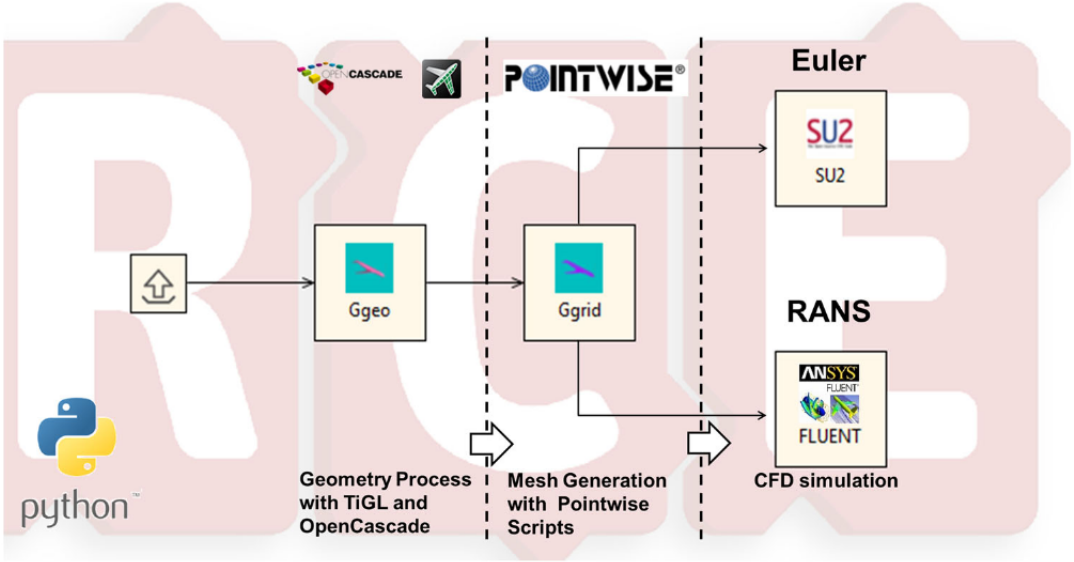

Figure 2.1: The workflow implemented by DLR. Source: [4]

Chapter 3

Developed Concept and Implementation

3.1 Overview

A workflow using Matlab, TiGL, Ansys Fluent and Ansys Fluent Meshing was developed for AeroFlow. The general flow of information and the different stages of work are illustrated in Figure 3.2. This section will give a short introduction to the tools and then explain the general method of connenction between them.

3.2 The Tools used

3.2.1 Controlling the Workflow: Matlab

Matlab is used througout the workflow to edit input scripts for other software, trigger execution of each program and to move and delete files. One exception is the use of python code (by DLR) to trigger CAD geometry creation. The execution of the python script however will also be controlled by Matlab.

3.2.2 Geometry Definition: CPACS

CPACS is a text based geometry definition scheme developed by DLR [1]. It is the format used to define the geometry and can be read by the geometry creation module TiGL (see below). A CPACS definition file contains all the geometric parameters such as fuselage shape and diameter or wingspan, profile section and positioning. The basic concept is as follows:

1. Define a cross section shape
2. Scale the cross section shape as needed
3. Place the cross section in the intended position
4. Place a second cross section in a different position
5. Create a lofted surface between the cross sections

This basic concept is used for the creation of geometry of the model. CPACS is not limited to this kind of geometry definition only. Control surfaces and propeller disks are just some examples of what the geometry library of of CPACS is capable of. For further information it is highly recommended to look at the user guide [1] as well as [4] and [10]. In the default configuration of the AeroFlow, the CPACS file will automatically be created based

on a proprietary UAV definition provided by [7]. This behavior can be changed to directly incorporate CPACS definition from external sources. (See section 3.3.5)

3.2.3 Geometry Creation: TiGL

TiGL is a tool developed by DLR [9]. In the scope of this work it will generate a STEP model from the aforementioned CPACS definition. It also provides a GUI that can be used to check the aircraft configuration in real time. Changes in the CPACS definition file can be directly visualized when loaded into TiGL Viewer. This is useful for manual generation of geometric variants. Furthermore this is one way to edit profile definitions. More information can be found in the user documentation [1]. In the default configuration TiGL will export a STEP file. This part of the AeroFlow workflow can be skipped to directly incorporate CAD files from external sources.

3.2.4 Meshing: Ansys Fluent Meshing

Ansys Fluent Meshing will read geometry as STEP files and generate an initial mesh based on the specified mesh parameters. This mesh will only resolve the aircraft surface properly. The rest of the domain will be treated by Adaptive Mesh Refinement is Ansys Fluent. More information on mesh generation strategies can be found in [8]. The initial mesh is produced with the *Poly-Hexcore* approach. A comparison of the different options can be found in [5].

3.2.5 Solution: Ansys Fluent

General

Ansys Fluent will read a template simulation case and replace its mesh for the current geometric variant. Boundary conditions and simulation control will be set according to the specified values.

Adaptive Mesh Refinement

A key element in the context of this work is Adaptive Mesh Refinement. The default setup contains refinement based on Turbulent Kinetic Energy and the Pressure Hessian. Note that the implemented thresholds are just a baseline and should be subject to investigation for production use of the toolbox. For more information the user is referred to Ansys Fluent documentation and the file:

`Run_[...]/AeroFlow/CFD/Solver/CFD_Basecase.cas.h5`

Result Output

Numeric data and a standardized .html report will be exported and moved to:

`Run_[...]/AeroFlow/Output/AoA_[...]deg`

These results can be accessed as soon as the corresponding angle of attack simulation is finished. The report contains information about the simulation setup, mesh statistics, numerical results for aerodynamic coefficients and multiple images. A 3D viewer is also included. Customizing the reports before conducting studies is recommended.

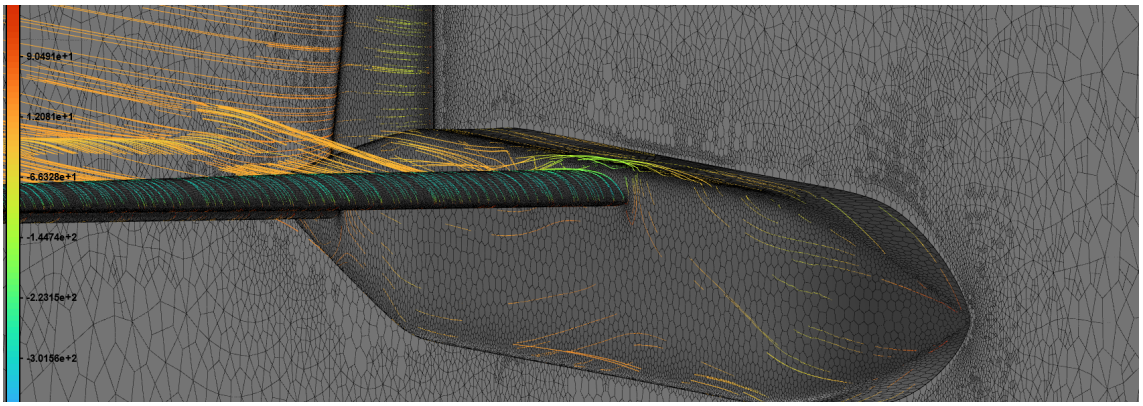

Figure 3.1: The 3D viewer in the postprocessing report

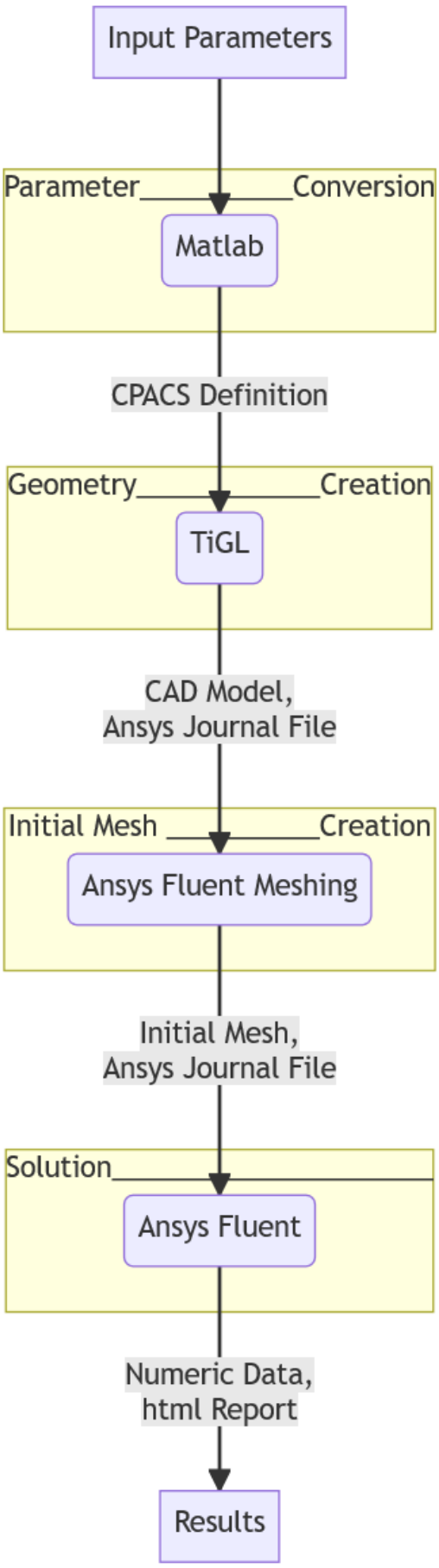

Figure 3.2: The overall flow of information in AeroFlow

3.3 Connecting the Tools

This section will explain how parameters are implemented and how specific software is made aware of them.

3.3.1 Input Interface

To change the default simulation setup to reflect a geometry, mesh setup or operating point of interest, input parameters for AeroFlow need to be specified.

Geometric Parameters

In the default setup, geometric parameters will be specified in a proprietary Matlab `.mat` file. (To specify a geometry in a different way, see section 3.3.5) The proprietary default file is:

`Run_[...]/AeroFlow/Input/myAircraftParameters.mat`

To use a specific geometry, replace this file with the design candidate of interest. An alternative way is to specify a different file to be loaded in:

`Run_[...]/AeroFlow/CFD/Geometry/step_1loadParam.m`

However specifying a different file *location* might cause issues with Matlab path management. The implemented setup includes fuselage, wing, vertical stabilizer and horizontal stabilizer. For a detailed look at all available parameters the user is referred to the file:

`Run_[...]/AeroFlow/CFD/Geometry/candidateScheme.cpacs.xml`

Mesh Parameters

Mesh parameters such as domain extents and resolutions in different areas are specified in:

`Run_[...]/AeroFlow/Input/meshingParameters.mat`

For setup and testing, it is recommended to use a coarse set of parameters. An example is provided.

Solver Parameters

Solver parameters are treated analogous to meshing parameters. Note that the default setup is not optimized. To get a proper understanding of the effect of all parameters, it is recommended to try different configurations and compare their effect on results. Solver parameters also include the angles of attack that shall be analyzed for the current geometric variant. Ansys Fluent will run a simulation for every angle of attack, starting from the initial mesh.

3.3.2 Script Parametrization

The next section will explain, how all the input scripts for third party software will be parametrized based on the specified input parameters. Every script containing geometric, meshing or solver parameters is based on a template file with placeholders. For example, the file:

`Run_[...]/AeroFlow/CFD/Mesh/fluentMeshingTemplate.jou`

This file contains keywords written in capital letters. Consider the matlab file:

`Run_[...]/AeroFlow/CFD/Mesh/step1_editMeshParamInJournal.m`

This script will read the template file, replace the keyword placeholders with their appropriate parameter values and save it as:

`Run_[...]/AeroFlow/CFD/Mesh/fluentMeshing.jou`

Note that the word *template* is now not part of the filename anymore. This basic procedure will be applied to all instances where a template file is used. The range of parameters in these scripts is only limited by the third party softwares. Technically AeroFlow can be expanded to the full extent of geometry, meshing and solver capability. The dafault scripts may serve as an example of implementation.

Special Treatment of Geometric Parameters

As the default way of providing geometric information uses different definitions than the CPACS standard, appropriate values are calculated by

`Run_[...]/AeroFlow/CFD/Geometry/step2_convertData.m`

Note that the formulae can be understood by looking at the proprietary input file. Naming variables was intentionally not done to avoid conflicting definitions after conversion. The values are subsequently written to

`Run_[...]/AeroFlow/CFD/Geometry/candidate.cpacs.xml`

The template file in this case is:

`Run_[...]/AeroFlow/CFD/Geometry/candidateScheme.cpacs.xml`

3.3.3 Script Based Execution

All software used in the workflow will be triggered from Matlab via a system command. These commands will not only start the software. They will also include the name of an input script for the corresponding program. For example, Fluent Meshing will be started and told to run the commands contained in ”fluentMeshing.jou” as follows:

```
system('fluent 3d -meshing -t4 -hidden -i "fluentMeshing.jou"')
```

The exact commands for every instance of a software being triggered can be found in

the Matlab scripts. For the syntax of each of these commands, the reader is referred to the documentation of the corresponding software. This basic concept applies to all steps in the workflow.

3.3.4 The Overall Workflow

To understand how all the tools are connected, it is recommended to go through the controlling Matlab scripts, starting at:

`Run_[...]/AeroFlow/allRun.m`

This file contains multiple run commands, corresponding to geometry creation, meshing and solution. Each of the subsequent Matlab scripts will themselves point to a deeper level of Matlab scripts. These contain the the actual commands that will modify files. The resulting tree like structure can be better understood by following the `run()` commands down the hierarchy.

3.3.5 Workflow Customization

It may be necessary to adapt AeroFlow for specific usecases. A few options for providing the input geometry have been considered in the development phase of the toolbox. This section will guide the reader through the basic options and necessary changes.

Skip Automatic Creation of CPACS Definition from Proprietary Source

If the user wants to provide a CPACS file from an external source, the paramater conversion from proprietary input can be deactivated. This can also be useful, if manual geometry variation shall be used. Manual creation of a CPACS definition can be done by simply editing a `candidate.cpacs.xml` file in a text editor. Note that the specified geometry variant can be simultaneously visualized using the TiGL Viewer. To deactivate parameter conversion, it is necessary to mark the command lines for *steps 1 to 3* as comments in:

`Run_[...]/AeroFlow/CFD/Geometry/runGeometryCreation.m`

Furthermore, a `candidate.cpacs.xml` must be placed in directory:

`Run_[...]/AeroFlow/CFD/Geometry/`

Note that this is not a validated setup.

Skip CAD Creation from CPACS Definition

If the user wants to provide the input geometry as CAD file directly, `runGeometryCreation.m` has to be deactivated completely. To get surface based refinements in the initial meshing steps, the implemented naming conventions need to be used. Corresponding information can be found in

`Run_[...]/AeroFlow/CFD/Mesh/fluentMeshingTemplate.jou (from line 85 on)`

Changes to this script may be necessary. Refer to Ansys Fluent Meshing documentation for further information.

3.3.6 Simulation Scheduler: Studyrunner

Exploring a parameter space requires a series of many simulations. All preprocessing steps have been automated in AeroFlow and even angle of attack variations are dealt with automatically. Yet if different geometric variations shall be compared, the user normally needs to start the next simulation run manually. To solve this problem, a task scheduler can be used. An advanced setup might include the use of software like DAKOTA [2]. To make basic functionallity more accessible, AeroFlow includes `studyRunner.m`. This is a simple Matlab script that will to through each `Run_[...]`-directory and start the `allRun.m` scripts, as soon as the previous case is finished. Note that intelligent error handling is not yet implemented. If one case setup causes a Matlab error, the scheduler will not resume. Care should be taken to make sure everything is set up correctly.

3.4 Limitations of the implemented CPACS Definition

3.4.1 Limited Profile Catalogue

CPACS definitions can contain a large set of parameters beyond the basic version used in this work. It is important to know that every piece of information available for geometry creation has to be specified inside the `candidate.cpacs.xml` file. This includes the cross sections for every geometry. In its current implementation, the included catalogue only contains square and circular sections for the fuselage and two different airfoil sections for the lifting surfaces. If the user wants to work with different shapes, these need to be added. To do this, it is recommended to look at `candidateScheme.cpacs.xml` and the CPACS documentation [1].

3.4.2 Profile Converter

CPACS files require a special format for profile definitions. To help converting commonly available coordinate data, a dedicated tool has been added:

`/additionalTools/profileConverter.m`

This script will read common coordinate files and rearrange the values as needed. It will then export files containing the strings required for the CPACS definition. Note that all relevant paths for input and output need to be specified manually before execution.

3.4.3 Trailing Edge Treatment

CFD Meshes will be of bad quality at sharp edges. To adress this issue, all trailing edges should have a thickness of at least 1% of the chord length as recommended in [3]. All used profile section shapes in the catalogue should be checked for compliance. Sharp edges can be changed by manually editing the point coordinates. To visually check the effect, TiGL Viewer can be used next to a text editor. An alternative way is to find a source of proper profile coordinates.

Chapter 4

Installation Instructions

The toolbox requires working installations of the following third party applications:

1. Matlab (tested with version 22b)
2. Python (tested with versions 3.9, 3.10, 3.11)
3. TiGL (tested with version 3.2.3)
4. Ansys Fluent Meshing and Soution (tested with version 22R2)

4.1 Notes on Third Party Installations

Installing the third party is a standart practice, so the reader is referred to the corresponding software documentation for further information. Only one aspect requires further attention: Python, TiGL and Ansys Fluent must be accessible from the command line.

1. For Python, this requirement is usually met by a standart installation.
2. For TiGL, the user needs to check the box for adding TiGL to the windows path variable during installation.
3. For Ansys Fluent, it is recommended to execute the following script, if the software is not already accessible from a terminal:

   ```
   [path to your Ansys installation]\v222\fluent\ntbin\win64\setenv.exe
   ```

   Note that this path can vary depending on your installation directory and version.

4.2 Obtaining the AeroFlow Toolbox

The tolbox is available on github via the following repository:

`https://github.com/MoritzBur/AeroFlow`

For instructions on using the toolbox, the reader is referred to the video tutorial series.

Chapter 5

Results and Review

5.1 Results and Review

5.1.1 Capabilities

AeroFlow is a framework that enables engineers to do state of the art CFD analysis with greatly reduced manual labour required. Through automation of recurring tasks and the use of adaptive mesh refinement, the user can focus more on the analysis of results. While it is possible to use only default geometry features and low subsonic operating conditions, the tool is built to leverage the full capabilities of its underlying software components. The toolbox does *not* offer a graphical user interface or means of checking the users input for plausability. This means that the implemented tool can not replace user expertise in the tools involved or CFD in general.

5.1.2 Use Cases

Finding a sensible Basecase

In conjunction with the provided task scheduler it is possible to investigate and compare the effects of different case setups. This might include comparisons of different turbulence models or AMR thresholds for example. This way the development of a sensible base case itself can be massively sped up. The user can specify different case setups, let all the simulations run and come back to review them as soon as all cases have finished. Note that access to all available parameters might require modifications to the template scripts and the template Ansys File:

`Run_[...]/AeroFlow/CFD/Solver/CFD_Basecase.cas.h5`

Simulation setups should be validated against experimental data. A simplified geometry variant containing only a finite NACA0012 wing is supplied as a helpful requisite in directory `Run_2`.

Studying geometry Variants

Once a basecase has been set up, variations of the geometry can be investigated. To do this, it is sufficient to copy the basecase and replace the included geometry definition. All geometry variants can be simulated automatically using the task scheduler `studyRunner.m`.

5.1.3 Validation done by the Developer

All parameters set should be investigated and set according to the case of interest. A simple validation has been made for a finite NACA0012 wing in `Run_2`, showing reasonable results. The results were compared against [6]. This however was just to show the viability of the implemented workflow, not the contents of the cases themselves. The developer does not recommend using this validation case without further investigation!

5.1.4 Usability

The implemented solution greatly simplifies studies of multiple similar aircraft designs and different simulation setups. When a validated setup is supplied, even users inexperienced in CFD can produce useful results. This might also help in learning about the process, as the user can productively work with the tool and gain more insight when necessary by exploring its contents.

5.1.5 Reliability

The tool does not offer any check on user supplied input and can therefore run into problems, causing simulations to crash. Especially the meshing parameters are critical. Reasonable, stable setups should be found before starting with geometry variation. The following component reviews rely on tests during the development phase of AeroFlow.

Geometry Creation

The geometry creation worked very reliabily and within seconds on a 4-core consumer PC from 2015.

Initial Mesh Creation

Mesh creation is commonly known to be one of the biggest challenges in CFD. The connection of Ansys Fluent Meshing into the automated workflow seems to be reliable. The default parameters for the published version of this work produced meshes of reasonably good quality, most of the time for default conditions. For production use however it is necessary to tune these parameters.

Solution

The integration of Ansys Fluent into AeroFlow seems robust, as long as no manual changes are made to the journal file. Ansys Fluent sometimes showed unexpected behavior and was not able to properly read journal files. Workarounds have been found to solve the problems. Unfortunately further inconsistencies can not be ruled out when functionallity is to be added to:
`Run_[...]/AeroFlow/CFD/Solver/fluentSolverTemplate.jou`.

Expandability

Unfortunately, the included Ansys software has been found to throw errors for some script functionallity. This mainly concerns correct interpretation of journal (`.jou`) files for meshing and solving. As no extensive testing of the tool could be done as of this writing, the reliability in productive use could not be established so far.

5.1.6 Ideas for Future Development

The tool can be improved in many ways. The most important as viewed by the developer will be mentioned here.

1. Transition from Ansys software to OpenFoam: The currently used Ansys software (especially the Ansys Fluent Solver) has been found to be unreliable when used with journal files. Documentation supplied by Ansys is also insufficient in many aspects, which will make any further development difficult. In its current form, the tool can only properly be used when the user has a valid license for Ansys software. The use of opensource software would make the tool more accessible. The developer recommends a transition to OpenFoam, as it has been found to be more reliable and better documented. Its approach of being controlled by simple text files also promises to be easier to modify and to accomodate more complex physics.

2. Transition from Matlab to GNU Octave: Switching to this alternative might be very simple, as GNU Octave is made to be a drop in replacement for Matlab. This would make the tool more accessible to everyone.

3. Implementation of checks for user input: Currently, improper user input can cause the simulations to crash or provide misleading results. To prevent wasting CPU time on setups that might have been ruled out in advance, a check could be implemented.

Bibliography

[1] *CPACS Documentation*. URL: https://cpacs.de/pages/documentation.html (visited on 02/19/2023).

[2] *Dakota | Explore and predict with confidence*. URL: https://dakota.sandia.gov/ (visited on 02/19/2023).

[3] Falk Götten et al. “A review of guidelines and best practices for subsonic aerodynamic simulations using RANS CFD”. In: Dec. 5, 2019. URL: https://www.researchgate.net/publication/337756101_A_review_of_guidelines_and_best_practices_for_subsonic_aerodynamic_simulations_using_RANS_CFD (visited on 02/19/2023).

[4] Xiangyu Gu, Pier Davide Ciampa, and Björn Nagel. “An automated CFD analysis workflow in overall aircraft design applications”. In: *CEAS Aeronautical Journal* 9.1 (Mar. 2018), pp. 3–13. ISSN: 1869-5582, 1869-5590. DOI: 10.1007/s13272-017-0264-1. URL: http://link.springer.com/10.1007/s13272-017-0264-1 (visited on 02/18/2023).

[5] Hashan Mendis. *Better meshing using ANSYS Fluent Meshing?* July 23, 2018. URL: https://www.linkedin.com/pulse/better-meshing-using-ansys-fluent-hashan-mendis (visited on 02/18/2023).

[6] A Merabet and B Necib. “Characterisation of Wings with NACA 0012 Airfoils”. In: (). URL: https://www.cder.dz/vlib/revue/nspeciauxpdf/icpwe_23.pdf (visited on 02/18/2023).

[7] Marvin Peschmann. “Entwicklung eines Computertools für den automatisierten Entwurf ziviler UAV”. Bachelorarbeit. Hochschule für Angewandte Wissenschaften Hamburg, June 23, 2021.

[8] Ideen Sadrehaghighi. *Adaptive Meshing*. Feb. 10, 2022. URL: https://www.researchgate.net/publication/339292239_Adaptive_Meshing (visited on 02/19/2023).

[9] Martin Siggel et al. “TiGL: An Open Source Computational Geometry Library for Parametric Aircraft Design”. In: *Mathematics in Computer Science* 13.3 (Sept. 1, 2019), pp. 367–389. ISSN: 1661-8289. DOI: 10.1007/s11786-019-00401-y. URL: https://doi.org/10.1007/s11786-019-00401-y (visited on 02/19/2023).

[10] Mengmeng Zhang et al. “Aircraft Geometry and Meshing with Common Language Schema CPACS for Variable-Fidelity MDO Applications”. In: *Aerospace* 5.2 (June 2018). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 47. ISSN: 2226-4310. DOI: 10.3390/aerospace5020047. URL: https://www.mdpi.com/2226-4310/5/2/47 (visited on 02/18/2023).