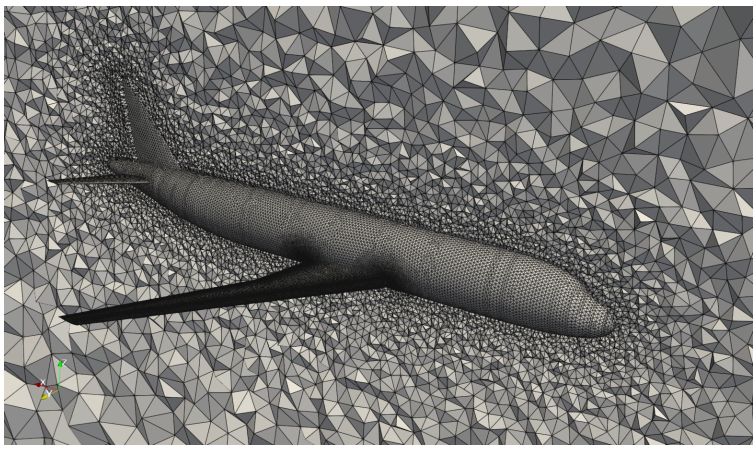
Technology
Arts Sciences
TH Köln

DLR

Automatic and robust aircraft mesh generator for CPACS using open-source tools

Automatischer und robuster Flugzeug-Gittergenerator für CPACS unter Verwendung von Open-Source-Werkzeugen



Masterarbeit zur Erlangung des akademischen Grades
Master of Science
im Studiengang Maschinenbau
an der Fakultät für Anlagen, Energie und Maschinensysteme
der Technischen Hochschule Köln

vorgelegt von: Ole Henrik Albers

eingereicht bei: Prof. Dr. Claudia Ziller - Technische Hochschule Köln
Zweitgutachter: Dr. Jan Kleinert - Deutsches Zentrum für Luft- und Raumfahrt

Köln, 31.07.2024

Abstract

In this thesis, an open-source automated mesh generator for the predesign phase of aircraft is developed to expedite aerodynamic analysis and reduce the time required for optimization processes. This enables early-stage predictions of characteristics for both conventional and unconventional aircraft configurations.

The mesh generator is built upon the generalized data format Common Parametric Aircraft Configuration Schema (CPACS) from German Aerospace Center (DLR), utilizing DLR's geometry modeling tool TiGL Geometry Library (TiGL) and the third-party meshing tool Gmsh to create aircraft-specific meshes. The generator is developed as a standalone Python tool with dependencies on TiGL and Gmsh. It produces unstructured and hybrid meshes with a prism boundary layer suitable for both Euler and RANS simulations. However, due to a flaw in Gmsh, the boundary layer generated is not usable for the entire aircraft. A temporary workaround has been developed to create the boundary layer only on the wings.

The generated meshes are validated through Computantional Fluid Dynamics (CFD) simulations of well-documented aircraft configurations and additional unconventional designs. Mesh convergence studies are performed, and the results are compared against existing experimental and simulation data to ensure accuracy.

The convergence studies indicate that the faulty boundary layer affects convergence. Simulations without a boundary layer are considered converged. However, detailed analysis of the convergence is beyond the scope of this work. The comparison of simulation results to existing data confirms that the generated meshes are sufficient for accurately replicating aircraft characteristics.

In conclusion, the developed mesh generator proves capable of producing adequate meshes for the predesign phase of aircraft models, enhancing the efficiency and effectiveness of early-stage aerodynamic analysis.

Contents

List of Tables

List of Figures

Acronyms / Abbreviations

AFLR Advancing-Front/Local-Reconnection.

AGARD Advisory Group for Aerospace Research and Development.

AOA Angle of Attack.

AR Aspect Ratio.

BREP Boundary Representation.

BWB Blended Wing Body.

CAD Computer Aided Design.

CFD Computantional Fluid Dynamics.

CPACS Common Parametric Aircraft Configuration Schema.

DLR German Aerospace Center.

DPW-II Drag Prediction Workshop.

FVM Finite Volume Method.

GUI Graphical User Interface.

HTP Horizontal Tail Plane.

MDAO Multidisciplinary Design Analysis and Optimization.

NURBS Non-Uniform Rational B-Splines.

OAD Overall Aircraft Design.

RANS Reynolds-averaged Navier-Stokes.

SA Spalart-Allmaras.

SST Menter shear-stress transport.

TiGL TiGL Geometry Library.

UID Unique Identifier.

VSP Vehicle Sketch Pad.

VTP Vertical Tail Plane.

XML Extensible Markup Language.

1 Introduction

A key requirement in the predesign phase of an airplane is to test the aerodynamic performance of new aircraft configurations quickly. The Open Source C++ Library TiGL generates three-dimensional geometric representations of an aircraft. These can be used to optimize the aircraft's geometry e.g. to minimize fuel consumption. [1] For an aerodynamic optimization or simulation the geometry needs to be discretized into a mesh. [2] TiGL however has shortcomings that require the time consuming process of manually exporting and meshing an aircraft model. This is not practical in an optimization with many loops, as it would lead to an immense amount of work. In the future, it would be more efficient to automate the process of generating geometry, mesh and simulation results after a parameter variation.

The aim of this thesis is to automate the process of generating meshes by implementing a robust mesh generator for TiGL. This would lead to a reduction in time required, as the export and external meshing of the geometry becomes obsolete. The mesh generator would simplify the process starting from the parametric geometry to generating the mesh. In NASA's CFD Vision 2030 study, automating CFD analysis is also one of the main goals. This requires automated mesh generation, which is the goal of this thesis. [10]

1.1 Related work

There are various approaches for the automated mesh generation for aircraft models that aim to solve the presented problem. Most of these approaches are part of an Overall Aircraft Design (OAD) environment, which aims to also include the flow simulations and the optimization into one workflow. A selection of papers regarding this topic is presented in the following paragraphs.

An automated CFD analysis workflow in overall aircraft design applications In this approach Gu et al. [4] present an automated CFD analysis workflow, using the CPACS. This workflow consist of a Computer Aided Design (CAD) aircraft model, TiGL, Pointwise as mesh generation tool as well as SU2 and ANYSY FLUENT for the CFD simulation. The presented tool Ggrid is used to create Python functions for Pointwise. These include information on "pre-implemented knowledge" regarding leading edges, trailing edges and other "critical positions". The mesh size is controlled by a global factor, which controls the number of nodes. With that a design environment is created, which consist of the created Python functions and therefore gives the user a set of selected Pointwise options to control the mesh parameters. This work is verified by the DLR-F6 wing body configuration, which is a well-known test case for CFD-simulations. The aircraft is meshed and a Reynolds-averaged Navier-Stokes (RANS) simulation is performed. The results are compared to wind tunnel and other CFD results with different meshing strategies. The approach presents a suitable, non-open source workflow for the automated mesh generation combining TiGL with the commercial meshing tool Pointwise.

CEASIOM The open source tool CEASIOM [5] is a conceptual aircraft design environment which is developed by CFS Engineering and Airinnova. It uses CPACS

configurations of aircraft to create meshes for CFD simulations and consist of the tool Tornado and the meshing tool Sumo. Tornado is used to handle aircraft for the analysis of linear effects. Since the scope of this work goes beyond that, Tornado will not be discussed further. The second tool Sumo can be used for non-linear flow analysis. It handles aircraft similar to CPACS which makes it easy to import the aircraft model. In general, Sumo is an unstructured mesh generator. With its extension Pentagrow, however, structured boundary layers can be created. Pentagrow takes a 2D surface mesh and extrudes prismatic elements onto the surfaces automatically. This creates a three-dimensional boundary layer. The remaining volume is then meshed by the meshing tool TetGen. The latest development of the Python version, CEASIOMpy, also works with Gmsh next to Sumo. The surface mesh for the boundary layer is now created by Gmsh. As it is in current development, it is not ready to use yet. [6]

Automated Tetrahedral Mesh Generation for CFD Analysis of Aircraft in Conceptual Design The main purpose of this concept by NASA [7] is the sonic boom analysis of supersonic aircraft configurations. The mesh is created in several steps. First, an inner boundary mesh is created with the CAD software Vehicle Sketch Pad (VSP). Then a half-cylindrical flow field is created. The volume mesh within the half-cylinder is then meshed with Advancing-Front/Local-Reconnection (AFLR)3. The resulting unstructured mesh is fine near the aircraft and the element size increases towards the boundary. The main achievement of this work is the automatic creation of an extending flow field in the back of the aircraft. The mesh is extruded backwards to three body lengths and the diameter of the half-cylinder also increases. This makes it possible to simulate the off-body pressure distribution behind the aircraft. One disadvantage of this approach is that it is not possible to refine the mesh at user-specified regions.

A framework for the generation of high-order curvilinear hybrid meshes for CFD simulations Turner et al. [8] present different meshing pipelines from a CAD model to a mesh. They either use OpenCASCADE or the CAD module of the commercial tool CADfix, CFI. The surface mesh is then generated by NekMesh or CADfix itself. In the presented work the used pipeline NekMesh uses the CFI for the geometry preprocessing and CADfix for the creation of a linear mesh which is then transferred to NekMesh to create a High-order mesh. The mesh generation involves building a "shell" around the airplane to create the volume required to create a linear boundary layer mesh. An O-type mesh is created around the wing to be able to mesh it with prismatic elements. To get a better mesh alignment at e.g. wing root junctions, the medial halo is used. This leads to a better result in these areas with potentially large velocity gradients. The creation of the medial object is one of the unique creations of the present work. To ensure a conformal mesh, a bottom up approach is used to create the mesh. This means, the created shell, which is divided into partitions, is meshed from the curve to the surfaces to the volume. To create the boundary layer mesh, which is required for a RANS simulation, the prismatic mesh of the shell is further split into layers to create an ascending layer thickness. Then the volume around the shell is meshed with an unstructured mesh, created with the Delaunay method. The results are verified with the NACA0012 airfoil, a well-known test object.

1.2 Research Question and Scope

In this thesis, an automated mesh generation tool, using the open source mesh generator Gmsh, especially for the aircraft predesign with TiGL, is developed and evaluated. The aim is to create CFD meshes for different aircraft configurations automatically and evaluate these meshes by performing CFD simulations. To do this, the general requirements for such a mesh have to be determined. This is done

by a research about CFD simulations and aircraft drag analysis. Furthermore the selected tools for the mesh generation and verification are analyzed. The simulations are used to proof the capabilities of the mesh generation tool in general and the automatic aircraft specific adaptions developed in this thesis. The evaluation is done through a convergence study in which certain values should converge with an increasing refinement of the mesh. For some models, experimental results exist which are used to evaluate the simulation results and therefore the quality of the mesh.

High fidelity aircraft drag analysis are not the subject of the thesis. The resulting values of the simulations cannot be taken as an exact reflection of reality. The meshes are considered as adequate when the simulation runs converge and the results are reasonable.

2 Fundamentals

The methods used for this thesis are explained in the sections 2.1 and 2.2. Sections 2.3 through 2.6 cover the tools and software that were used to achieve the aim of the work. Those tools are TiGL for the geometrical handling of aircraft configurations, Gmsh for the automated meshing and SU2 as the CFD tool to test the created meshes and verify the presented method. TiGL is in scope since the goal is to create a mesh generation tool for TiGL. Gmsh has also been already used in previous works and has several advantage, e.g. it is possible to export the mesh and the geometric representation and thus obtain the geometrical information. This is used by the adaptive mesh refinement research group at the Institute of Software Technology and the usage of Gmsh could help to bring the respected software of the two groups closer together. The presentation and explanation of the software used will be the content of this chapter.

2.1 Computational Fluid Dynamics

The partial differential conservation equations which occur in fluid mechanics cannot be solved analytically. They must therefore be approximated numerically. CFD offers the possibility to approximate the equations by simulating the flow around a geometry without performing an experiment such as a wind tunnel test. Due to the many

factors that influence a complex flow simulation, it is very challenging and need to be thought through. [9]

The development of CFD in recent decades has fundamentally changed the aircraft design process. Which can partly be attributed to the progress in computer science and the increase in computing power. CFD evolved from low-level calculations of linear flows via inviscid flow methods (Euler) towards higher fidelity simulations with RANS. [10] With the increasing complexity of modeling in the early stages of aircraft design, physics-based analyses such as CFD are becoming increasingly important. The following disciplinary fidelity levels of modeling complexity are described by La Rocca [11] as follows:

- "level 0: consisting of typical conceptual OAD approaches, based on empirical relations and existing databases;
- level 1: refers to disciplinary analysis based on simplification on the modeling, and on the representation of the physics phenomena, mainly accounting for linear effects;
- level 2: refers to an accurate modeling of the aircraft components, accounting for a higher level of details, and physics representation accounting for non-linear phenomena;
- level 3: refers to the state of the art of physics simulations, mainly dedicated to non-linear local effects, and whose disciplinary models cannot be fully automated, as required for extensive Multidisciplinary Design Analysis and Optimization (MDAO) applications."

The higher the fidelity level, the greater the demands on the aerodynamic analysis. Level 2 can be associated with an inviscid, nonlinear CFD simulation (Euler) and

Level 3 with a compressible, viscid RANS simulation. These are used in the predesign of an aircraft for simple wing-body configurations.

In CFD the continuous problem domain is replaced by a discrete problem domain. This is done by using a mesh (in the following also referred to as grid). The equations can then be solved at the grid points and the remaining locations are interpolated from these points. This leads to a large number of algebraic equations which can be solved iteratively by a computer. [9]

The most commonly used method for the discretization is the Finite Volume Method (FVM). In two-dimensional meshes, a quadrilateral or triangle is called a cell and a grid point a node. In three-dimensional meshes, the cells are mostly hexahedrals, tetrahedrals or prisms. Such a cell defines a control volume in the FVM. In the FVM, the integral from the conservation equations is applied to the respected control volume to get a discrete equation for the cell. One advantage of the FVM is that it maintains the flow across the cell boundaries and across the entire mesh. Another method, the Finite Element Method (FEM), is not able to do this. In addition to the mesh, boundary conditions must also be defined so that the discrete equations can be solved. These conditions are defined for the cells near the boundary of the domain. Boundary conditions are, for example, the pressure inlet, the pressure outlet, the velocity inlet or a symmetry. [2, 9]

2.1.1 Meshes

As already mentioned, in CFD simulations the geometry is represented by a mesh, i.e. the geometry is approximated by basic geometrical shapes. The resolution of the mesh is the key part between computing time and results. In CFD applications, a flow field (in the following also referred to as farfield) is created and then meshed. The flow field is often a box of the surrounding fluid which is analyzed in the simulation.

In most cases, the symmetrical properties of aircraft models are used, meaning the aircraft and the farfield are cut in half at the symmetry plane of the aircraft. The surface of the farfield which is then intersected by the aircraft's shell is also referred to as symmetry plane. From this flow field the geometry is subtracted so the flow around the geometry can be analyzed. [9] In meshing there a two general approaches, the structured and the unstructured meshing.

Structured meshes have a uniform topology and are commonly quadrangles (two-dimensional space) or hexahedras (three-dimensional space), as pictured in Figure 2.1. [12]

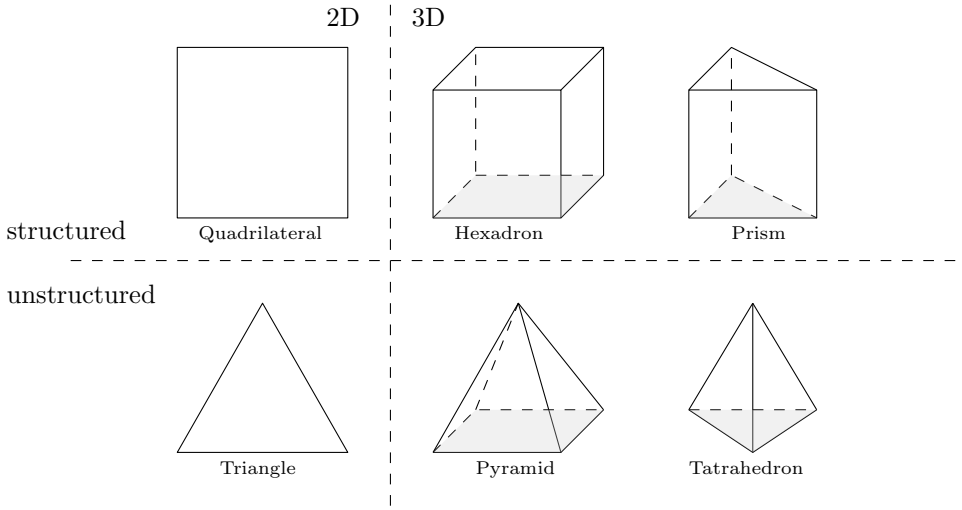


Figure 2.1: Structured and unstructured mesh elements in 2D and 3D [12, 13]

The simplest case is a structured mesh in which all edges have the same length, which is known as a cartesian mesh. However, more complex geometries can also be represented by a structured mesh. Three common topologies are the O-, the C- and the H-mesh (Figure 2.2). The three types differ in particular in the course of the edges, which are created by the position of the individual elements. [2]

Unstructured meshes are very flexible and consist of triangles and quadrangles. They

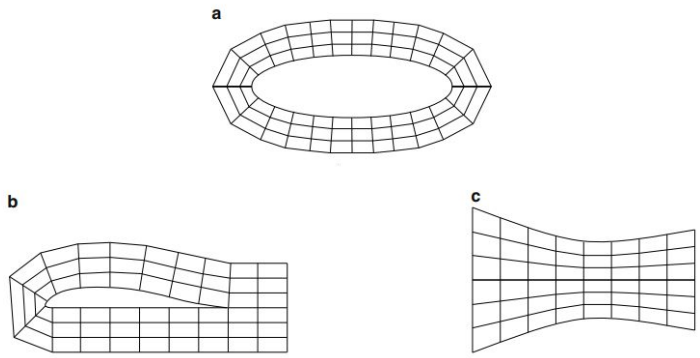


Figure 2.2: The three common mesh topologies O-(a), C-(b), H-(c) mesh (image from [2])

can therefore be easily used for complex geometries. Unstructured meshes can also be created automatically and therefore are useful for many applications. [14] There are different approaches to create unstructured meshes. Some of these are the Delaunay, advancing-front and quad-octree methods. In the Delaunay method, adjacent points from a given set of nodes within the area are connected and tetrahedral cells are created to ensure that no other points fall within the sphere defined by the vertices of each tetrahedron. In the advancing-front technique, the grid is constructed incrementally, moving from the boundary towards the interior by systematically connecting new points to those on the advancing front until all remaining unmeshed regions are filled with grid cells. The quad-octree method covers the physical domain in cubic cells. The cells which lie on the boundary are then recursively subdivided until the desired resolution is reached. The cells are then transferred to tetrahedral cells, creating an unstructured mesh. [12]

A third approach, opposed to structured and unstructured meshing, are hybrid meshes. Those are most useful for viscid simulations such as RANS, because they combine the advantages of the previous two approaches. An example is the creation of a structured boundary layer around an airfoil to describe the viscous flow near the surface of a

body. [15] This is necessary because the fluid directly at the surface is at rest. The velocity increases with distance from the surface. The boundary layer height (y) is defined from the surface to the point in which the speed coincides the surrounding speed. As shown in Figure 2.3, there are different kinds of velocity (U) profiles which have the same start (y = 0) and end (y = δ) but different accelerations. In order to simulate these effects, it is crucial to create a fine, structured mesh in this area. [14]

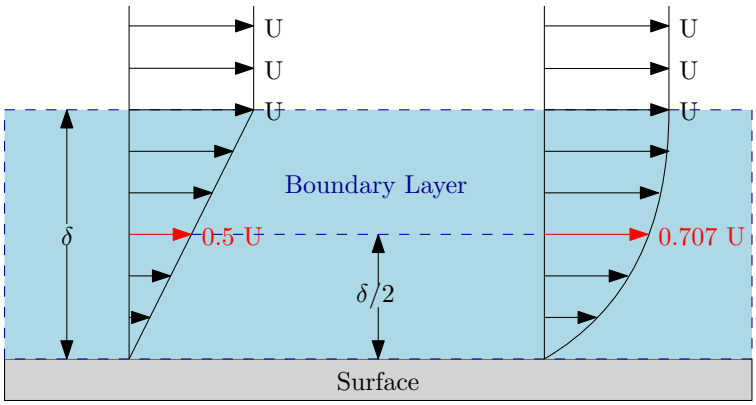


Figure 2.3: Possible ways the speed can increase in the Boundary Layer [14]

Since the mesh quality is crucial for the quality of the results, a mesh convergence study is mandatory when performing a CFD analysis. This ensures the discrete results are as close as possible to the exact results. Therefore, it is good practice to carry out several simulations to investigate the effects. [9] There are two ways of reducing the cell size or increasing the number of elements: uniform and non-uniform refinement. Assuming there are three meshes containing N_1, N_2 and N_3 elements, such that $N_1 < N_3 < N_3$, then a uniform mesh refinement would mean $N_1 * r = N_2$ and $N_2 * r = N_3$ where $r = \frac{N_2}{N_1} = \frac{N_3}{N_2}$. In the case of local mesh refinements, such as the refinement of a leading edge, the refinement is non-uniform. [15] To evaluate the convergence, the order of grid convergence, which describes the behavior of the

solution error, can be determined. There are two types of order of convergence, the theoretical order of convergence, provided by the numerical algorithm of the CFD tool and the observed order of convergence, influenced by the numerical models, the boundary conditions and the mesh. The observed order of convergence is mostly lower than the theoretical order of convergence. The solution error E is the difference between the discrete solution $f(h)$ and the exact solution f_{exact}:

$$E = f(h) - f_{exact} \tag{2.1}$$

To approximate the function E around 0, a Taylor expansion is performed. The Taylor series of E around 0 is as follows:

$$\begin{aligned} E &= 0 + E'(h) * h + E''(h) * h^2 + H.O.T. \\ E &= Ch^p + H.O.T \\ log(E) &\approx log(C) + plog(h) \\ E &\in O(h) \end{aligned} \tag{2.2}$$

C = constant

h = measure of mesh size

p = order of convergence

The simulation quantity f can be determined with the assumption that the simulation result $F(h) \to f$ as $h \to 0$:

$$\mathrm{f} = F(h) + Ch^p + H.O.T. \tag{2.3}$$

When $F(h)$ is known for three different mesh sizes $h_1 > h_2 > h_3 > 0$ and the Higher Order Terms (H.O.T.) from the solution error are neglected, f can be determined as

follows:

$$\mathrm{f} \approx F(h_1) + Ch_1^p, \mathrm{f} \approx F(h_2) + Ch_2^p, \mathrm{f} \approx F(h_3) + Ch_3^p \tag{2.4}$$

By the eliminating f and C, the equation can be written as:

$$(F_3 - F_2)h_1^p + (F_31 - F_3)h_2^p + (F_2 - F_1)h_3^p = 0 \tag{2.5}$$

where $F_n = F(h_n)$

For a uniform mesh refinement, it then follows that:

$$p = log(\frac{F_3 - F_2}{F_2 - F_1})/logr \tag{2.6}$$

Since there is no single r in a non-uniform mesh refinement, Equation (2.6) must be solved numerically. As one solution of p is $p = 0$, the question is whether there is another solution for $p > 0$. Dividing both sides of the equation by h_3^p leads to:

$$g(p) = (F_3 - F_2)(\frac{h_1}{h_3})^p + (F_1 - F_3)(\frac{h_2}{h_3})^p + F_2 - F_1 \tag{2.7}$$

and:

$$\frac{\mathrm{d}g}{\mathrm{d}p} = (F_3 - F_2)(\frac{h_1}{h_3})^p log(\frac{h_1}{h_3}) + (F_1 - F_3)(\frac{h_2}{h_3})^p log(\frac{h_2}{h_3}) \tag{2.8}$$

If $g(p)$ has a stationary value, it is given by:

$$p_s = log(\frac{F_3 - F_1}{F_3 - F_2})\frac{log(\frac{h_2}{h_3})}{log(\frac{h_1}{h_3})}/log(\frac{h_1}{h_2}) \tag{2.9}$$

However, these equations are only suitable for a monotone convergence, meaning that either $F_1 < F_2 < F_3$ or $F_1 > F_2 > F_3$ must be given. [15] An additional way to evaluate the grid convergence is the Grid Convergence Index (GCI) suggested

by Roache [16]. The GCI is determined by two meshes with different sizes and is a measure of percentage that indicates how far the value is from the asymptotic value. It is based on the Richardson Extrapolation [17], which is used to improve the accuracy of a simulation. The GCI is defined as follows:

$$GCI = F_s \frac{|\epsilon|}{r^p - 1} \tag{2.10}$$

where the refinement ratio r is:

$$r = (\frac{N_2}{N_1})^{\frac{1}{3}} \tag{2.11}$$

and the relative deviation of $F(h)$ for two different meshes ϵ is:

$$\epsilon = \frac{F_2 - F_1}{F_1} \tag{2.12}$$

The factor of safety F_s for the comparison of two grids is usually $F_s = 3$. As the GCI indicates how much a further refinement would change the solution, a small GCI indicates a sufficient refinement. [2]

2.1.2 Euler Simulations

The Euler equation describes the flow of an inviscid fluid. The equation is a simplification of the Navier-Stokes equations and can be used to approximate flows. A good example is the lift of a small airfoil, where the Euler equation provides a good model of reality. As described previously, Euler simulations are used in OAD to simulate lower fidelity geometries of Level 2. Since friction is not taken into account, the boundary layer effect will not occur in Euler simulations. Therefore, creating an unstructured mesh is sufficient. [2]

2.1.3 RANS Simulations

One of the most common CFD methods is to use the RANS equation. It is easy and quick to solve, which leads to short calculation time. The RANS method uses the Reynolds-average to model turbulent flows. In comparison to other methods like the Large-Eddy-Simulation (LES), RANS does not solve individual turbulence elements. Rather, the Reynolds-averaged flow field is created. The influence of the turbulence to the flow field must then be approximated by turbulence models. [9, 2]

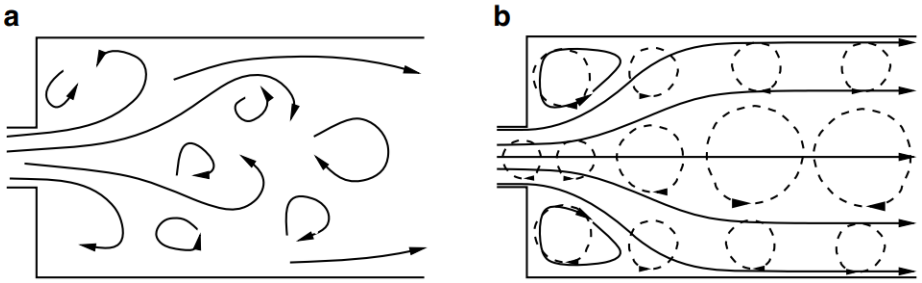


Figure 2.4: Turbulent flow (a) and Reynolds-averaged flow with turbulence model (b) (image from [2])

In Figure 2.4 a turbulent flow downstream of a widened cross-section is displayed. On the left side the pure turbulent flow can be seen. On the right side the turbulent flow is averaged. The dashed lines show the approximated turbulent flow which would represent the real turbulent flow in a CFD simulation.

2.2 Aircraft Drag Analysis

There are several effects to consider when analyzing an aircraft. Since the focus of this work is predesign, only basic aircraft configurations are analyzed. This means the aircraft only consist of a fuselage with wings. Components like flaps, engines, landing

gear, etc. are not taken into account. The components of an aircraft that are most important for this thesis are visualized in Figure 2.5.

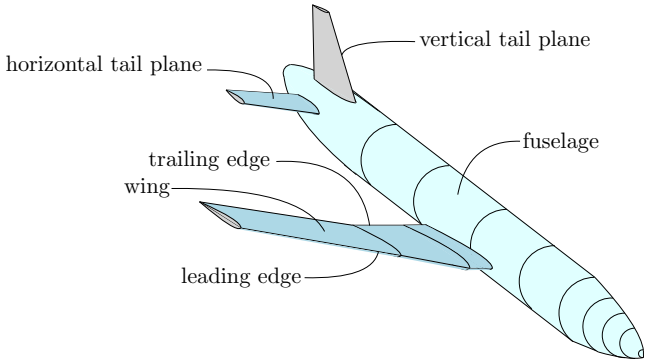


Figure 2.5: Components of an aircraft

2.2.1 Analysis of an Airfoil

When an object moves through a fluid, it experiences two types of forces: compressive force, caused by pressure fluctuations due to its shape, and friction force, resulting from viscous shear stress. The combination of the tangential friction and compressive force, which is orthogonal to the surface, is the resultant force r. As seen in Figure 2.6 the lift (l) is the component of r which is perpendicular and the drag (d) tangential to the flight path. The point, one-forth of the chord (quarter chord) starting at the leading edge is chosen to consider the resulting moment. [14]

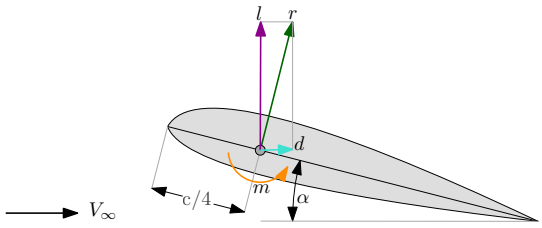


Figure 2.6: An airfoil with it's forces and moments [14]

The resultant force r can be written as:

$$r = \frac{1}{2}\rho V_\infty^2 S C_r \tag{2.13}$$

where

ρ = Density

V = Airspeed

S = Reference wing area in m^2

C_r = Coefficient that relates Angle of Attack α to force

In the same way lift, drag and pitching moment can be determined in the quarter chord, as seen in Figure 2.6. When analyzing an airfoil, S is assumed to be the reference area of a wing with a unit span (S = chord x 1 = chord). With the chord c, lift, drag and the pitching moment can be written as:

$$\begin{aligned} l &= \frac{1}{2}\rho V_\infty^2 c C_l = \frac{1}{2}\rho V_\infty^2 c C_r \cos\alpha \\ d &= \frac{1}{2}\rho V_\infty^2 c C_d = \frac{1}{2}\rho V_\infty^2 c C_r \sin\alpha \\ m &= \frac{1}{2}\rho V_\infty^2 c^2 C_m \end{aligned} \tag{2.14}$$

where

C_l = lift coefficient

C_d = drag coefficient

C_m = pitching moment

The lift-to-drag ratio (ld) can be determined dividing the lift and the drag coefficient. It is crucial for the airfoil efficiency and is often plotted against C_l. When comparing two of these graphs one point to look at is the ld_{max} and the respected C_l value. A lower C_l here means a higher efficiency. When comparing airfoils, the airfoil efficiency

ratio can also be used. It is defined as $C_{l_{max}}/C_{d_{min}}$. [14]

Another important value is the pressure coefficient C_p. For compressible flows it is defined as:

$$C_p = \frac{2}{\gamma M_\infty^2}\left(\frac{p}{p_\infty} - 1\right) \tag{2.15}$$

where

p = Pressure

p_∞ = Farfield pressure

M_∞ = Farfield Mach number

γ = Ratio of specific heats = 1.4 at typical altitudes

The most common way of looking at C_p is the chordwise pressure distribution. This reflects the distribution of surface pressure on an airfoil. It is typically shown as a plot from C_p over the upper and lower surface for a given α (Angle of Attack (AOA)). Usually for better visibly the vertical axis is inverted with the negative scale at the top. [14]

2.2.2 Analysis of a Wing

This section provides an overview of the wing. Since many parts of a wing are similar to the airfoil, this subsection is presented in a compressed form. In general the wing is the three-dimensional shape of an airfoil. One of the most important information of a wing is the reference area. Many important values require calculation of the reference area. The equations in section 2.14 can be rewritten for the wing by swapping the chord c with the reference area S. In general, the reference area is set in the beginning of an aircraft design process and not adjusted by minor changes to the aircraft's geometry. Another geometrical parameter of a wing is the chord, which was also used already in subsection 2.2.1. There are different types of chords, the root chord c_r, the

tip chord c_t or the main chord, the mean geometric chord c_{MGC} or just MGC. This is not the average chord, as it is the chord length at the geometric centroid of an area. Considering half a wing cut in the symmetry plane, the c_{MGC} would be located in the middle for a rectangular wing and at a third of the length from the symmetry plane for a triangular wing. [14]

Another element of the wing is the aspect ratio. The general Aspect Ratio (AR), with b being the length of the wing planform from tip to tip, is defined as:

$$AR = \frac{b^2}{S} \tag{2.16}$$

It influences the lift coefficient and the stall AOA. The stall AOA is the Angle of Attack with the maximum lift coefficient. A low AR leads to a low lift coefficient but a higher stall AOA. Therefore, for example, a sailplane which requires a very high lift coefficient has a high AR, meaning that the length of the wing planform is large and the chord length comparatively small. [14]

2.3 CPACS - Common Parametric Aircraft Configuration Schema

CPACS is an aircraft and aviation data model. Its goal is to enhance the process efficiency within a collaborative environment by being an open and central data model. CPACS is developed by the DLR since 2005 and is based on the Extensible Markup Language (XML). The generic character of the open standard XML makes it possible to use it as a computer-processable meta-language. Other markup languages such as HTML do not provide this possibility, since it has a fixed list of tags. CPACS uses the hierarchical structure of XML and follows a top-down approach to describe aircraft models, as seen in Figure 2.7. As the level of detail in aircraft design processes

increases over time, CPACS can reflect this in its structure. Some components, however, follow a bottom-up approach where they are first described in detail and then linked to the rest of the aircraft. An example are engines, which only need to be created once and then can be linked multiple times. [18]

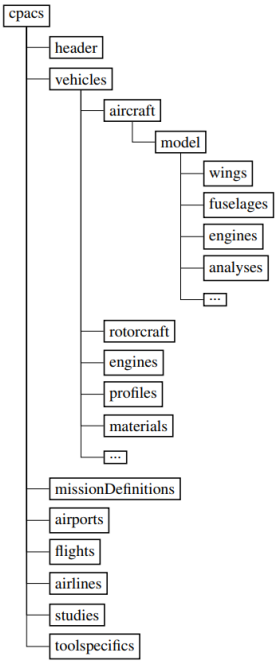


Figure 2.7: The CPACS hierarchical structure (image from [18])

2.4 TiGL Geometry Library

TiGL is an open source library for geometry modeling. It creates aircraft models from its standardized parametric CPACS configuration. As well as CPACS, TiGL is also developed by the DLR. [1]

TiGL converts the parametric description of aircraft and helicopters to three-dimensional shapes. The geometry modeling in TiGL is based on the OpenCASCADE CAD kernel.

It is able to represent the geometries by using B-splines, Non-Uniform Rational B-Splines (NURBS) and B-spline surfaces. The OpenCASCADE Boundary Representation (BREP) adds metadata to the shapes, which can be used for example to refine certain areas in the meshing process. TiGL and CPACS are designed to be used in optimizations. The parametric design of CPACS leads to easy controllable configurations with few parameters. TiGL enables the user to integrate those easily changeable configurations in an automated workflow. For example by changing just a few parameters, the shape of the fuselage or the wing can be varied. By the aircraft compound module, both external aircraft components, such as wings, fuselages, flaps, and internal structures, like ribs, spars, cutouts can be created. In this thesis, the focus is on the external components for CFD simulations, but the meshing of the internals should be considered in the future. Next to BREP, TiGL can export the geometries to standard CAD formats such as IGES, STEP and VTK. Even though TiGL is originally a C++ library it can be used with Python, C, MATLAB and Java. In this work TiGL is mainly used with Python. [1]

2.5 Gmsh - Three-dimensional Finite Element Mesh Generator

Gmsh is the an open source software to create fast and light meshes. It consists of four modules, the geometry, mesh, solver and post-processing. Since the solver is an finite element solver for structural analysis, it is not suitable for the thesis and only the geometry and mesh modules are describes and used. [3]

2.5.1 Geometry Modeling

Gmsh has two CAD kernels for geometry modeling. The native built-in Gmsh CAD kernel and the OpenCASCADE CAD kernel. The built-in CAD Kernel can be used to create simple geometries, but does not provide Boolean operations. Therefore the OpenCASCADE kernel can be used to create or import more complex geometries. These can then be further modified. Another advantage of using the OpenCASCADE kernel is that it can process BREP files. It is also possible to import a geometry in a selected format and modify it using one of the CAD kernels. The kernel can even be changed for different functions, but there must be a synchronization between them. Both CAD kernels define a 3D model using four topological entities. [3]

1. Vertices dimension 0
2. Edges dimension 1
3. Faces dimension 2
4. Regions dimension 3

Each entity contains the information about the up- and downward adjacencies. Following the path of entities, as described above, is the only way to create a model. [3]

1. Define points
2. Define edges, which connect the points
3. Define faces, which are enclosed by chosen edges
4. Define volumes, which are enclosed by chosen faces

2.5.2 Mesh Generation

As mentioned before, the TiGL geometries are defined by NURBS. Gmsh has the ability to mesh theses parametric surfaces in the parametric space rather than the real space. This leads to extremely robust mesh generations, because it is guaranteed, that elements do not overlap in a plane. But the mesh in the parametric space is non-uniform. [3] The Delauny approach solves this problem in Gmsh by using a metric based on the form of the surface. This metric then defines angles and distances in the parametric space. [19]

Another advantage of Gmsh is the concept of local mesh modification. This means after the standard process of discretization of the edges, the mesh is modified. This is done by splitting edges that are too long, removing edges that are too short, swapping edges and relocating vertices. For all of these modifications the real and the parametric coordinates are saved. For the three-dimensional mesh generation, Gmsh uses standard algorithms which rely on the previously explained surface meshes. In addition Gmsh has a built-in mesh optimization. This is crucial for a robust mesh generation. Gmsh uses edge- and face-swapping as well as vertex relocation. These are similar to the local mesh modification for the surfaces. In addiation to that, Gmsh can also use third party tools such as Netgen [20]. An example for the built-in optimization is the improvement of the aspect ratio. The desired aspect ratio is 1. Assuming there are about 600 000 tetraherads in a mesh and more than 5000 have an aspect ratio worse than 0.2 and the worst is 1e−3. One iteration of the optimization can improve this and the worst aspect ratio now is 0.32. [3]

Gmsh can create meshes in different dimensions, the 1D-mesh (curve mesh), the 2D-mesh (surface mesh) and the 3D-mesh (volume mesh).

2.6 SU2 - Computational Analysis and Design Package

The selected CFD-Solver is the open source software SU2. It is developed at Stanford University. The software uses the explained FVM and is specifically designed to solve problems in aerospace engineering using the RANS method for compressible, turbulent flows. These are the kind of simulations which will be performed in this work. SU2 can use two turbulence models for the RANS Simulation, the Spalart-Allmaras (SA) and Menter shear-stress transport (SST). These won't be discussed more deeply here, since this is not the scope of the thesis.

In addition, SU2 can use the adjoint method. A method which is widely used in optimization processes. The decision for SU2 was therefore not only based on the purpose of this work, but also to propose a software environment that can be used for further projects such as the development of a complete aircraft optimization or a OAD environment. Using SU2 also benefits from SU2's software architecture as it is also written in C++ and can be used with Python. [21]

When SU2 is used for basic CFD simulations, two input files are required. The first input file is the mesh in the *.su2* format. The second file is the so-called configuration file. It contains all the information about the simulation conditions, the file name of the mesh, the names of the output files, etc.. SU2 can be run serially or in parallel. In this thesis SU2 was operated in parallel on the DLR HPC-cluster CARA.

3 Mesher Architecture

The mesh generator (also referred to as mesher), which is developed in this thesis, uses selected Gmsh functions and combines them with TiGL information. This brings the challenge to map the geometrical information from TiGL to Gmsh. In this work, methods are developed to solve this challenge. The developed mesher can be divided into three modules. The first module extracts the information about the aircraft from TiGL, the second module maps the information to the aircraft in Gmsh and the third module is the actual mesher with the meshing functions. These here created modules are described and explained in this chapter, followed by a short guide for the mesh generation.

When using the mesher a CPACS (.xml) file is imported in TiGL manually. Then the TiGL configuration is created and the mesher class can be initialized.

In the first mesher function, the aircraft is merged into one shape and exported to the OpenCASCADE BREP format in order to use the aircraft in Gmsh. This is a common step when using TiGL, but it is modified to get further information. Typically, all wings and the fuselage are fused together by a simple command. In this case, the faces of the wings that do not intersect the fuselage are important for the boundary layer and therefore the process is extended. Instead of taking the wings as a whole, all wing surfaces are fused with the aircraft and the list of all intersecting entities is exported. Subsequently, these intersecting entities are compared with all surfaces and the not intersecting surfaces can be saved in a new list. As this fusing method can

lead to errors in the resulting shape, it cannot be used to proceed with the meshing. For this reason, the fused shape is deleted and the fuse is also performed in the usual way. With this more complicated process, both the list and a functional shape are created. Figure 3.1 shows the different types of fusing. In a) the wing is divided into three segments with the corresponding upper and lower shapes. These are fused with the fuselage in the first step. In b) the wing and the fuselage are used as whole and in c) the fusing in complete. The dashed line shows what is cut from the wing by the fuselage.

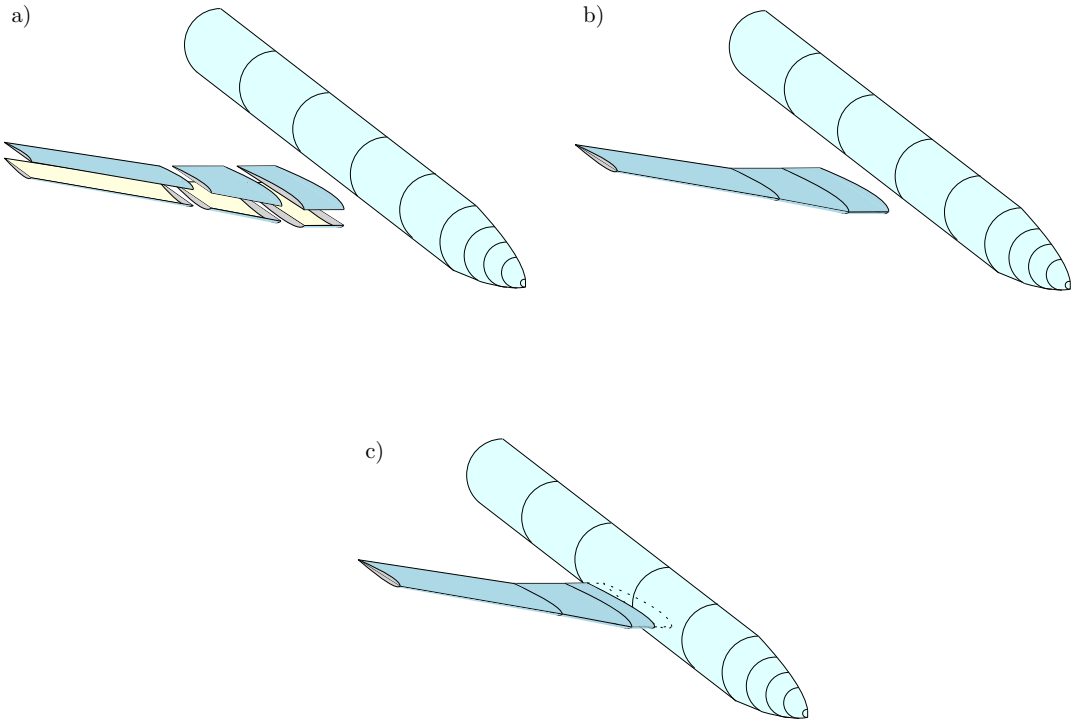


Figure 3.1: The steps of fusing an aircraft

Since the goal of the mesher is to create robust meshes, there must be fusing procedures for configurations such as a Blended Wing Body (BWB) containing only wings or a rocket with only a fuselage. To address these cases, all fuselages are fused together in the beginning. This also works, when there is just one fuselage. The wings are then

fused together one after the other and then fused to the fuselage if there is one. In fusing processes there is a parent and a child. The parent will trim the child and will not be trimmed. The here implemented fusing procedure always suggests that the fuselage or the fused fuselages are the parents of the process.

3.1 TiGL module

To gather all the needed information of the aircraft a method is developed. In this method all information is saved in a so-called map. The map is a built-in Python function, which can be structured in categories and these can then be accessed by using keys. The here created map is called the TiGL map, as it stores TiGL information. In this thesis the most important key is the so-called Unique Identifier (UID). It is used from CPACS and therefore from TiGL to identify the components of the aircraft. Every wing for example has its own UID. The categories with the information which is considered to be the most important to use for meshing or mesh refinements from a wing are:

- leading edge
- trailing edge
- lower trailing edge
- blunt trailing edge

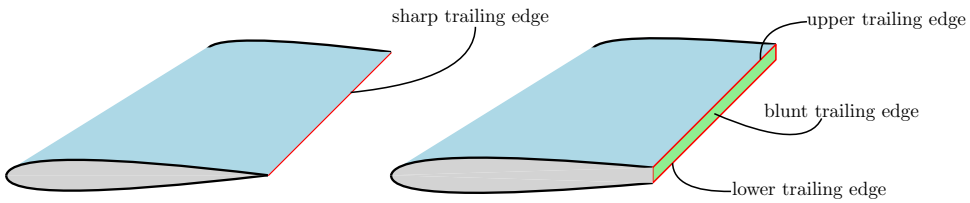


Figure 3.2: Two wings with a sharp trailing edge and a blunt trailing edge

As displayed in Figure 3.2, a blunt trailing edge exist when the trailing edge is not just an edge, but rather a flat surface. In this case, there is an upper and a lower trailing edge, as well as the surface itself. The normal trailing edge only ever takes the upper trailing edge into account, since there is only one edge if the trailing edge of this wing is sharp. The lower trailing edge is used if there is a blunt trailing edge and both trailing edges are refined. The second option is the refinemet of the whole trailing edge surface. As the TiGL map is created at the beginning, all information are stored there. So if the wing has a blunt trailing edge, both trailing edge categories as well as the blunt trailing edge category are filled with information. Since theses edges usually consist of multiple edges, they are stored in these categories. So every category consist of a list of all edges, which e.g. are the trailing edges of the wing with the UID "wing".

For every edge which is stored in the map, four pieces of information are considered. These pieces are:

- The edge itself as TopoDS Edge which is a pointer to the edge in the OpenCASCADE CAD kernel.
- The start point, saved as the OpenCASCADE entity *gp_pnt* which contains the coordinates of the point.
- The end point, also saved as *gp_pnt*.
- An index which shall remain empty in the TiGL module and will be filled later by the Gmsh module.

As describes above, each wing consist of wing segments and theses segments consist of shapes. There is an upper shape, a lower shape and also a trailing edge shape, if the trailing edge is blunt. Looking at the upper shape, it has four boundary curves, the leading edge, the (upper) trailing edge and two which form the upper half of the

airfoil. Figure 3.3 displays a wing with two segments split into the respected upper and lower shape. This wing has a sharp trailing edge and therefore there is no trailing edge surface.

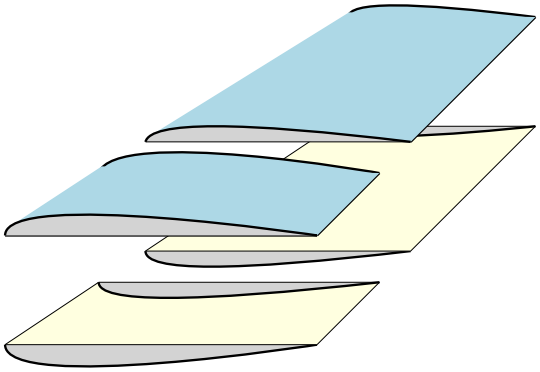


Figure 3.3: Wing with two segments split into upper and lower shapes

CPACS has the so-called eta and xsi coordinates for these shapes. These are shown in Figure 3.4 with eta and xsi as coordinates in the format: (eta, xsi). They each go from 0 to 1 and describe the entire shape parametrically. Eta extends along the leading edge and xsi along the chord of the wing. The two points with xsi being 0 are therefore the start (eta = 0) and the end (eta = 1) of the leading edge. The start and end of the trailing edge can be determined the same way. With this information, in the developed method, all edges of the shape are checked and the matching edges are saved to the TiGL map alongside with the start and end points. If a wing has a blunt trailing edge the trailing edge of the lower shape is saved the same way. In this case, the category "blunt trailing edge" is added to the TiGL map to save the index of the surfaces in the Gmsh module as well. This process is repeated for every wing and wing segment.

Other properties of the aircraft can be added to the wings, such as the fuselage. However, this is not done yet, as there is no need for it in this work. But the developed mesher already has the placeholder category, which will be expanded in

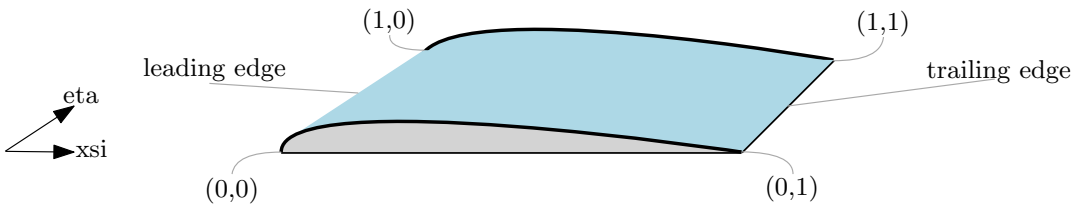


Figure 3.4: Upper shape with eta and xsi coordinates

the future

3.2 Gmsh module

The Gmsh module is used to insert in the fourth piece of information in the TiGL map. Gmsh uses OpenCASCADE as well as TiGL, but the native entities cannot be accessed by Gmsh. It is therefore not possible to take a TopoDS Edge and access it with Gmsh to refine it. Gmsh rather uses so-called dimtags. These are used to control the CAD and the mesh functionalities of Gmsh. A dimtag consists of two parts, the dim being the dimension of the entity as explained in section 2.5.1 and the tag being an explicit index. Theses tags are the desired indices for the fourth piece of information. Since in this work only edges are considered, the dim does not have to be preserved.

Before accessing one of the models, the given CPACS configuration is imported to TiGL and then exported as a BREP CAD file by the mesher. The mesher then creates a new directory and saves the BREP file there. This is used to import the aircraft model into Gmsh, as Gmsh cannot handle CPACS configurations.

When the aircraft model is imported, each edge in the CAD file is written into a list, which the toll will the iterate through. At each iteration, the start and end of the current edge are extracted with Gmsh. With the start and end points also written as "*gp_pnts*", the next step is to iterate through the UIDs of the TiGL map. If the current UID describes a wing, it is used to access the next level of the map, the categories. Here, the leading edge as well as the upper and lower trailing edges are taken into account since the blunt trailing edge is a face which will be considered at a later stage of the code. Now the three categories are iterated through, starting with the leading edge. The first leading edge of the current UID is taken and the start and end points are extracted from the map. These are then compared with the start and end of the current Gmsh edge. As there may be small differences in the geometries between two software applications, the comparison is done with a tolerance of 1e−6.

If the start and end points align, the edges are considered to match and the tag of the current edge is saved as the index in the map.

One challenge when accessing a wing UID is that the geometric information is always based on the original wing, rather than the trimmed shape after the fuse. The original edges are stored in the TiGL map, so for any edge where the start and end points do not match, it must be considered that the edge may have been trimmed. This is done by projecting the Gmsh start and end points onto the edges of the map and check whether they are on the edge with the same tolerance as above. Since the function only considers the orthogonal distance it does not take into account points that are collinear to the edge. Therefore, both described ways are conducted.

In the case that the wing has a blunt trailing edge, two trailing edges are saved, the lower and the upper. In addition the faces of the trailing edge shape are also saved. This is done by iterating through every surface of the Gmsh model and comparing the boundary edges with the two trailing edges. If one surface has an upper and a lower trailing edge as its boundary, it is a blunt trailing edge surface and its index will be saved in the TiGL map.

3.3 Mesher functions

As described above the developed mesher functions mainly wrap Gmsh functions and combine them into some aircraft related meshing functions. The first function is the *import_model* function. It starts the fusing and is used to set the option to create a farfield and a boundary layer. Since Gmsh requires a CAD file to work with, it is automatically exported in the background. If the user does not want a farfield or a boundary layer, this leads to a simple mesh of the aircraft's geometry, which in most cases is not useful for CFD simulations. If both, a farfield and a boundary layer, are

selected, the aircraft is cut in half along the symmetry plane by creating a box and performing a Boolean operation with both shapes. In this process a new surface is automatically created to close the now open shape in the symmetry plane. The new surface will also be deleted, as it is not required for a farfield. The trimmed aircraft is then saved as the current CAD model to be used with Gmsh. The same process is used if the boundary layer is not selected. The function *__create_farfield* creates the farfield for both scenarios. Since the symmetry plane of the farfield is intersected by the aircraft's shell, a hole is created. In the case of a boundary layer, the hole must be created by the outer layer of the boundary layer and not the shell of the aircraft.

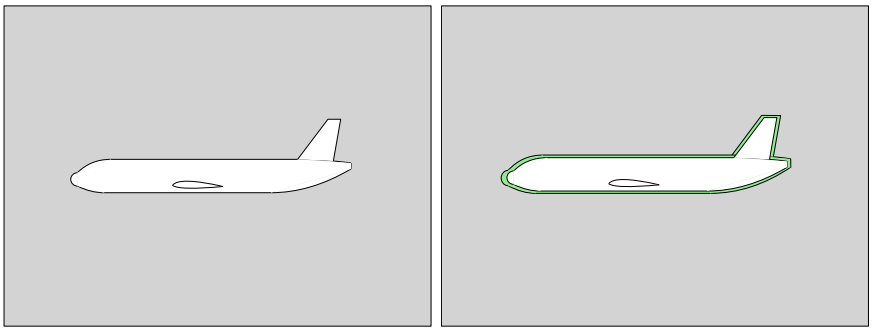


Figure 3.5: Symmetry plane of the farfield cut by the aircraft (left) and by the boundary layer (right)

General options To give Gmsh a range for creating an automatic mesh, the user can specify a minimal and a maximal mesh size. The scale of the value depends on the size of the aircraft. The minimal mesh size is only required if no refinement is selected. The refinement of certain areas of the aircraft is done with the so-called mesh fields in Gmsh. For these fields, Gmsh creates a volume around a given entity with a refined mesh. If one were to give Gmsh a line, Gmsh would create a cylinder around this line in which the mesh is refined to the given mesh size. From that cylinder, the mesh size increases until it reaches the maximal size. The two options for controlling

this behavior are the minimal and maximal distance. The minimal distance specifies the radius of the cylinder or the distance of the refinement. The maximal distance specifies the distance until the maximal mesh size is reached.

Refinements The developed commands for the refinements start with the expression refine, followed by the entity. To this point there are four refinement functions implemented in the developed mesher:

- *refine_leadingEdge*
- *refine_trailingEdge*
- *refine_blunt_trailingEgde*
- *refine_aircraft_surfaces*

The first three refinements are called with the UID of the wing and a mininmal mesh size to be used for this refinement. As described above, the mesher will use the UID to automatically identify the leading and trailing edges. The *refine_aircraft_surfaces* function only requires the minimal mesh size as input and is used in particular with farfields to ensure that the mesh size around the aircraft is fine enough. The mesh around the boundary of the farfield is very coarse and can therefore affect the mesh of the aircraft.

Boundary Layer The developed *boundary_layer* function is used by the user to specify options for the boundary layer and simultaneously creates the boundary layer and the farfield in the background. This is possible, because a boundary layer can only be used in combination with the farfield. The options that can be specified are the thickness of the first layer and the number of layers. The *boundary_layer* function has a third option which is to only create the boundary layer on wings. This option

is currently mandatory to get a working boundary layer. With the Gmsh function *extrudeBoundaryLayer* and given surfaces, Gmsh simply extrudes the boundary layers in the normal direction of the surfaces. This happens without any quality control which can lead to a self-intersecting mesh and intersections between the mesh and the geometry. During this thesis it became evident that these problems occur at the intersection between wing and fuselage. The problems are sketched in Figure 3.6 on the left-hand side. In b) and c) the boundary layers intersect each other. In a) the boundary layer intersects the fuselage's geometry and in d) the boundary layer leaves a gap to the fuselage. One idea to solve these problems, are faces which split the boundary layer of the fuselage and the wing in an angle. However, since the extrusion does respect geometry boundaries, this approach cannot be used. The solution which avoids this problem is to only create a boundary layer on wing surfaces which do not intersect with the fuselage. The process of extracting these surfaces is described in the introduction of chapter 3. This enables the user to perform RANS simulations for the entire aircraft, however the analysis is only possible for the part of the wing with the boundary layer. Further side effects and deficits are described in chapter 4. A second flaw of the boundary layer generation with Gmsh is that it is always extruded orthogonally to the selected surfaces not respecting any boundaries. This becomes a problem if only a wing is meshed and the boundary layer is orthogonal to the sweep angle and therefore not to the symmetry plane, resulting in a deformed symmetry plane. The sketch on the right-hand side of Figure 3.6 shows this problem, with the dashed line representing the supposed symmetry plane.

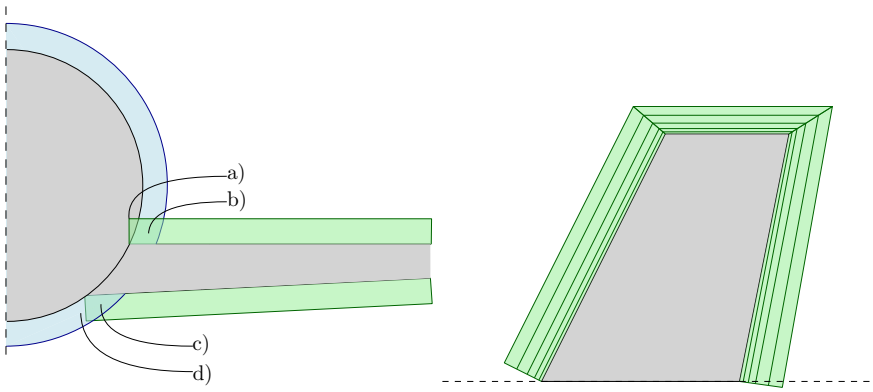


Figure 3.6: Boundary layer problems

3.4 Creating a Mesh

Before the CFD simulations can be conducted, a mesh must be created with the presented mesh generator. A Phyton file is created for this purpose, which then executes the mesher functions. The libraries used are imported at the beginning. These are the *Mesher* itself, the *tigl3wrapper*, the *tixi3wrapper* and the *CPACSConfigurationManager*. TiGL is then set up, the CPACS file (here "D150_v30_new.xml") is imported and a so-called configuration (*config*) is created. This configuration describes the aircraft and is now used to access its information via TiGL.

```
1  from Mesher import Mesher
2  from tixi3 import tixi3wrapper
3  from tigl3 import tigl3wrapper
4  from tigl3.configuration import
     CCPACSConfigurationManager_get_instance
5
6  tixi_h = tixi3wrapper.Tixi3()
7  tigl_h = tigl3wrapper.Tigl3()
8
9  tixi_h.open("TestData/D150_v30_new.xml")
```

```
10    tigl_h.open(tixi_h, "")
11    mgr = CCPACSConfigurationManager_get_instance()
12    config = mgr.get_configuration(tigl_h._handle.value)
```

Once this process is completed, the mesher class can be initialized with the *config*. The *import_model* function is then called to determine whether a farfield and a boundary layer are required. The function also starts the described fusing process.

```
13    mesher = Mesher(config)
14    mesher.import_model(farfield=True, boundary_layer=True)
```

The Mesher now has all the basic information and the mesh parameters can be set. As described above, the maximal mesh size and the minimal and maximal distance are mandatory. If the boundary layer is selected, the thickness of the first layer (here 1e−5) as well as the number of layers (here 5) have to be set. In addition, the user must select the *wing* option. This enables the described workaround to create only the boundary layer on the wing. In this case, the aircraft's boundary curves cut the hole in the farfield's symmetry plane, as if there was no boundary layer. Therefore, this option excludes an intersection of the boundary layer with the farfield's symmetry plane. For configurations such as the BWB or only one wing such as the ONERA M6, the option must be set to *false* to create a boundary layer over the entire model which creates the hole in the farfield. The refinements for the leading and trailing edges as well as the surfaces can then be set. In this case, the wing with the UID "D150_wing_1ID" is selected to be refined with a mesh size of 0.02 at the leading edge. The refinement for all aircraft surfaces is set to 0.2.

```
15    mesher.set_maxMeshSize(4)
16    mesher.set_minDistance(0.5)
17    mesher.set_maxDistance(8)
18
```

```
19  mesher.boundary_Layer([1e-5], 5, True)
20
21  mesher.refine_leadingEdge("D150_wing_1ID", 0.02)
22
23  mesher.refine_aircraft_surfaces(0.2)
```

Finally, the mesh is created in the selected dimension using the *mesh* function and then saved in the selected format using the *write* function. To display the mesh in Gmsh's Graphical User Interface (GUI), the *run_gmsh_fltk* function is used.

```
24  mesher.mesh(3)
25  mesher.write("D150_mesh.su2")
26  mesher.run_gmsh_fltk()
```

4 Results

To test the capabilities of the mesher, two well known test cases are meshed, simulated and analyzed. These are the ONERA M6 wing and the DLR-F6 aircraft configuration. To further test the capabilities of the mesher, additional aircraft are tested, but analyzed in less detail. These include fundamentally different designs as well as variations in the configuration, such as a different sweep angle of the wings or newly added wings. This chapter is organized into sections, each dedicated to a specific configuration. In each section, the respective configuration is introduced and the mesh is described. Following this, a CFD simulation is performed and analyzed. If experimental or simulation results already exist for the aircraft models, they are compared with the simulations conducted in this thesis. In addition convergence studies are performed to test whether a chosen value converges with an increasing number of mesh elements. The tested configurations are listed in Table 4.1. The table indicates whether the creation of both 2D and 3D meshes, as well as the boundary layer, was successful using the mesher. The note "wing" in the boundary layer column signifies that the boundary layer could only be created on the wing, as described in Section 3.3. The final two columns indicate whether the performed CFD simulations converged. With the exception of the boundary layer and the unsuccessful RANS simulation for the D250, all categories in the table are marked as completed. The detailed reasons for the boundary layer problems of the D250 are explained in Section 4.4.

Table 4.1: Overview of tested aircraft

Aircraft	2D mesh	3D mesh	Boundary layer	Euler	RANS
ONERA M6	√	√	√	√	√
F6	√	√	wings √	√	√
D150	√	√	wings √	√	√
D250	√	√	X	√	X
BWB	√	√	√	√	√
Concorde	√	√	wings √	√	√

To further evaluate the mesher, more configurations were tested. However these models are confidential and can therefore not be presented in this thesis. The conclusions that are drawn from these tests are nevertheless included in chapter 5. As mentioned above, all simulations were run on the DLR HPC-cluster CARA. Each simulation used 20 compute nodes with two AMD EPYC 7601 (32 cores; 2,2 GHz) processors and 128 GB RAM per node.

4.1 ONERA M6 - Wing

The ONERA M6 was created in 1972 by Bernard Monnerie to get experimental results of three-dimensional flows at transonic speeds and high Reynolds numbers. It was created at Onera in cooperation with the former NATO agency Advisory Group for Aerospace Research and Development (AGARD). [22] The wing's geometrical definition can be found in the Schmitt and Charpin report [23]. This geometrical layout is displayed in Figure 4.1. Since there is no CAD model available, a CPACS configuration was created for this thesis. Its wide use in literature brings many results for a comparison. For this thesis, the results from the SU2 tutorials are used. This has the advantage that the same SU2 configuration file can be used and the differences in the results are exclusively caused by the mesh.

As mentioned above, the results of the ONERA M6 wing are compared to the results

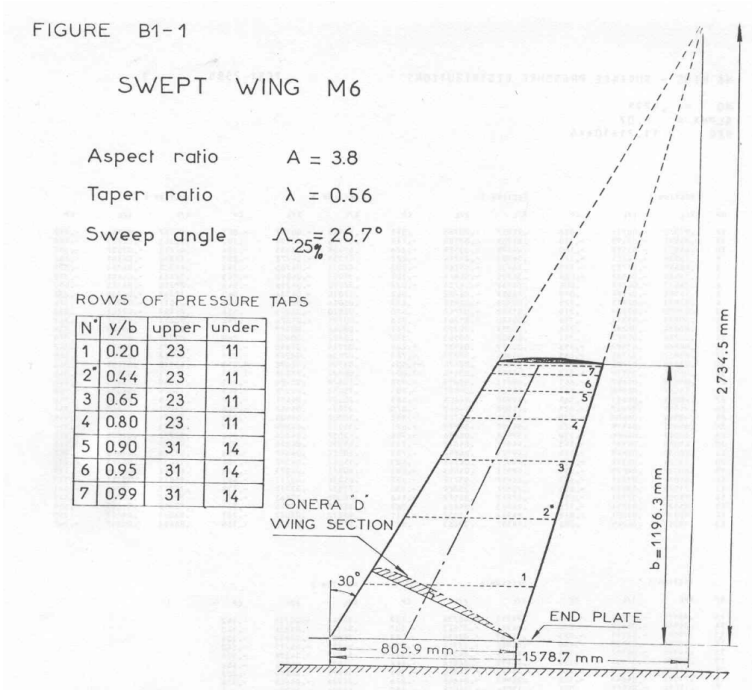


Figure 4.1: Geometric layout of the ONERA M6 wing (image from [23])

in the SU2 documentation [24, 25]. There are Euler as well as RANS simulations. Therefore, both were recreated using the SU2 configuration files. The Euler simulations were performed with the conditions in Table 4.2. The temperature T and the pressure P originate from the ISO 13443 standard natural gas reference conditions with the pressure of 101 325 Pa being equal to one standard atmosphere. The Mach number Ma and the AOA are also taken from SU2. Four simulation runs were carried out with successively finer meshes.

Table 4.2: ONERA M6 simulation conditions

Ma	0.8395
AOA	3.06°
T	288.15 K
P	101 325 Pa

Table 4.3 shows the four simulations and the target simulation from the SU2 documentation. The four simulations differ in the mesh and are therefore called Mesh 1, Mesh 2, Mesh 3 and Mesh 4. In order to get the complete set of target results, the SU2 simulation was recreated with the given mesh and configuration file. The first simulation with a rather coarse mesh shows that the values for the lift coefficient C_l and drag coefficient C_d are close to the target. But with an increasing number of elements they get closer. When comparing the SU2 mesh with the Mesh 1 qualitatively in Figure 4.2, the SU2 mesh seems to be more structured than the created mesh. The figure also shows that the coarser Mesh 1 has a larger minimal element size and the increase is faster than in the SU2 mesh. The SU2 mesh has more than 200 000 elements more. This changes with the rising number of elements between the simulations, which was reached by increasing the refinement at the edges. The refinement of the surfaces was increased additionally for Mesh 3 and Mesh 4. For a more detailed comparison between the SU2 results and the results created here, the simulation of Mesh 4 is used since it is considered to be the most accurate.

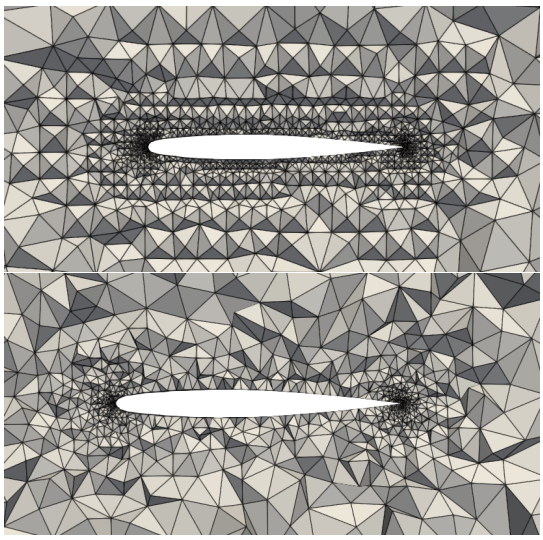


Figure 4.2: Comparison of SU2 (top) and mesh 1 (bottom)

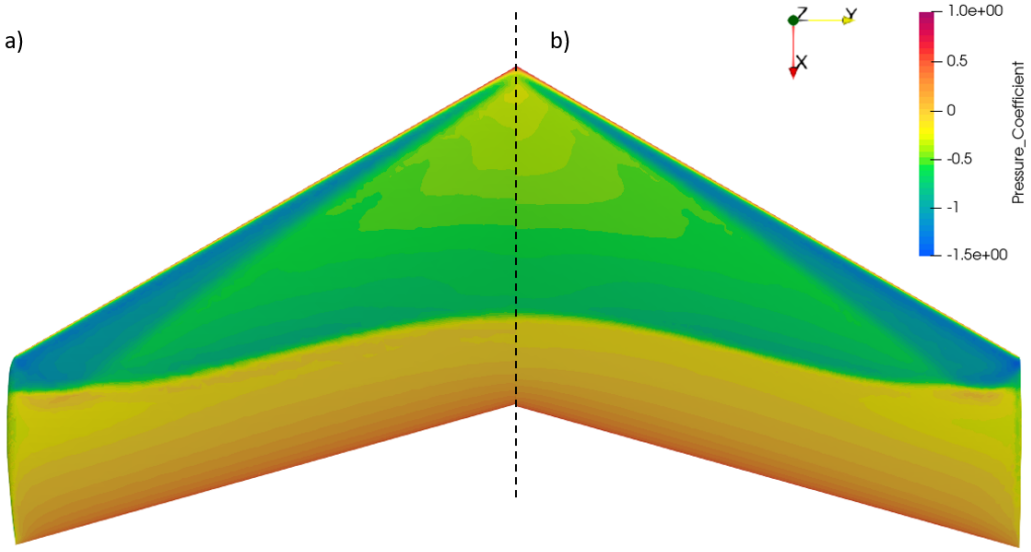


Figure 4.3: Comparison of the pressure coefficient C_p from SU2 (a) and simulation 4 (b)

Table 4.3: ONERA M6 Euler simulation

	Elements	Iterations	C_l	C_d
SU2	582 752	617	0.286 167	0.012 13
Mesh 1	355 395	617	0.279 271	0.012 733
Mesh 2	1 681 411	600	0.281 022	0.012 328
Mesh 3	2 692 838	648	0.282 218	0.012 348
Mesh 4	6 428 422	134	0.281 431	0.012 221

Figure 4.3 shows that the distribution of the pressure coefficient over the upper surface of the wing is very similar. Both sides use the same legend and the values match on both sides. Considering Table 4.3 and Figure 4.3 the targeted SU2 simulation is recreated with the developed mesh generator. To evaluate the mesh convergence the C_d value in Table 4.3 is analyzed. By comparing the values it becomes evident that they differ in the forth decimal digit. This however, does not reflect the mesh convergence, as the mesh size is neglected. Since there are no experimental or exact results a solution error cannot be determined either. Another way to try to evaluate the

mesh convergence is described in Section 2.1.1. The order of convergence, as defined in Equation (2.9), cannot be calculated for these simulations, since the convergence is not monotone. The result for Mesh 3 is greater than the results for the Meshes 2 and 4. Therefore only the GCI can be calculated, as it only requires two meshes. With Equation (2.10), the GCI is calculated for two consecutive meshes and displayed in Table 4.4. As the order of convergence cannot be calculated, but is required for the GCI, it is assumed that the theoretical order of convergence or order of accuracy from the numerical method is equivalent to the observed order of convergence. The standard numerical method in SU2 is the Jameson-Schmidt-Turkel (JST) Scheme, which has a second order accuracy, meaning $p = 2$ is used for all GCI calculations in this thesis. The two compared meshes are specified in the first column of the table.

Table 4.4: GCI for ONERA M6 Euler simulations

	GCI
Mesh 1 to Mesh 2	5.2 %
Mesh 2 to Mesh 3	1.3 %
Mesh 3 to Mesh 4	3.9 %

The calculated GCI suggests that the difference in the is relatively low. The interpretation of the GCI, as indicator for a good mesh convergence, however, is not easily done. In this work the GCI is calculated and taken as another way of comparing meshes but not as concluding evaluation.

To further verify the mesher's abilities a RANS simulation is performed with the ONERA M6 wing, using the conditions from Table 4.2 extended by the Reynolds number = 11.72e6, the Reynolds length = 0.646 07 m and the SA turbulence model. These values are also specified in the SU2 documentation [25]. The main differences between the mesh from SU2 and the created mesh are that the SU2 mesh is a 100 % structured mesh. Such a mesh cannot be created by Gmsh and therefore a hybrid mesh with a prism boundary layer is used. The difference can be seen in Figure 4.4,

with the structured SU2 mesh at the left and the hybrid mesh at the right. The boundary layer cannot really be seen here since it is too small with a first layer thickness of 1.1818e−4 m and six layers.

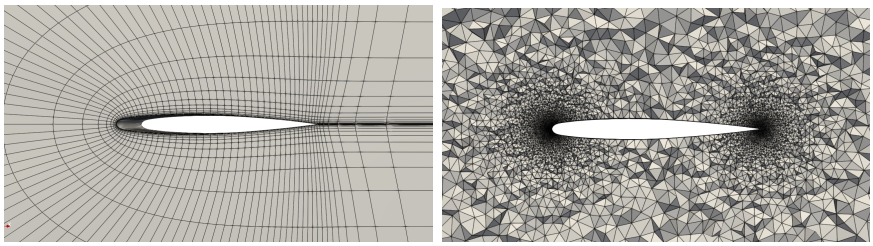


Figure 4.4: Comparison of SU2 (left) and mesh 1 (right)

Three RANS simulation runs were performed, which are listed in Table 4.5. The large difference in the number of elements can be explained by the different meshing approaches. In the structured mesh from SU2, the mesh size rises uniformly and a smaller amount of elements is created. A more complex mesh can also lead to an increased simulation time which is reflected by the large number of iterations. But since the transitions between the elements can be smoother, the iterations do not always rise by the number of elements, as seen in the simulation runs for Mesh 2 and Mesh 3.

Table 4.5: ONERA M6 RANS simulation

	Elements	Iterations	C_l	C_d
SU2	43 008	837	0.251 491	0.015 835
Mesh 1	1 045 853	7619	0.236 391	0.014 034
Mesh 2	2 036 762	11 128	0.238 39	0.013 338
Mesh 3	15 598 875	11 063	0.259 898	0.013 205

Analyzing the drag coefficient C_d throughout the three simulations, the values decreases monotonously. This enables the calculation of the order of convergence p with Equation (2.9). Since only three simulations where conducted, p can only be calculated once and therefore is $p = 6.96$. This order of convergence is unusually high, as the order of

accuracy should be greater than the observed order of convergence. As for the Euler simulations, the GCI is calculated for the RANS simulations with Equation (2.10).

Table 4.6: GCI for ONERA M6 RANS simulations

	GCI
Mesh 1 to Mesh 2	26.6 %
Mesh 2 to Mesh 3	1.0 %

As the GCI values is Table 4.6 decrease while the number of elements increases, this can be interpreted as a good convergence result. The change between Mesh 1 and Mesh 2 is high in perspective to the number of elements. With a increasing number of elements, the difference becomes smaller and a potential further increase of elements would not make a big difference in the value of C_d. This interpretation is an assumption based on the conducted simulations. Further simulations must be performed to better evaluate the convergence.

To better compare the results of the here conducted simulations to the SU2 simulation, the pressure coefficient is plotted over the chord of the wing in different sections. Figure 4.5 shows a comparison between the SU2 mesh, a coarse mesh (Mesh 2), a fine mesh (Mesh 3) and the wind tunnel results from Schmitt and Charpin [23, 26]. The sections are defined by y/b, which is the relative position on the model, where 0 is the symmetry plane and 1 the wing tip. The same applies for the chord length x/c, where 0 is the leading edge and 1 is the trailing edge. The comparison shows that the results of the coarse and fine mesh are close to the results of the targeted SU2 mesh for most of the section. At the trailing edge however the pressure coefficient behaves differently. The sudden jump in the C_l can be explained by the poor creation of the boundary layer by Gmsh. Figure 4.6 shows that the boundary layer is almost non-existent at the trailing edge, which leads to an abrupt transformation to an unstructured mesh and a poor simulation result. This can be observed for all four wing sections. In section $y/b = 0.8$, the C_p deviates from the SU2 meshes at around $x/c = 0.3$, but

the wind tunnel results suggest that the created meshes might even work better since they are closer to the experimental results. In section $y/b = 0.95$, the SU2 results are closer to the experimental results. This can be explained by the influence of the wing tip. The geometry used in the SU2 simulation and the wind tunnel experiments have a rounded wing tip. The geometry used in this thesis has a flat plane as wing tip. This geometrical difference can also be observed in Figure 4.7 with the SU2 simulation on the left and the created mesh simulation on the right. As seen in Figure 4.7, the wingtip has an influence on the pressure coefficient. In b) a yellow area can be observed near the wing tip, which is not present in a). Except this small difference, this comparison of the pressure coefficient shows an accurate simulation result.

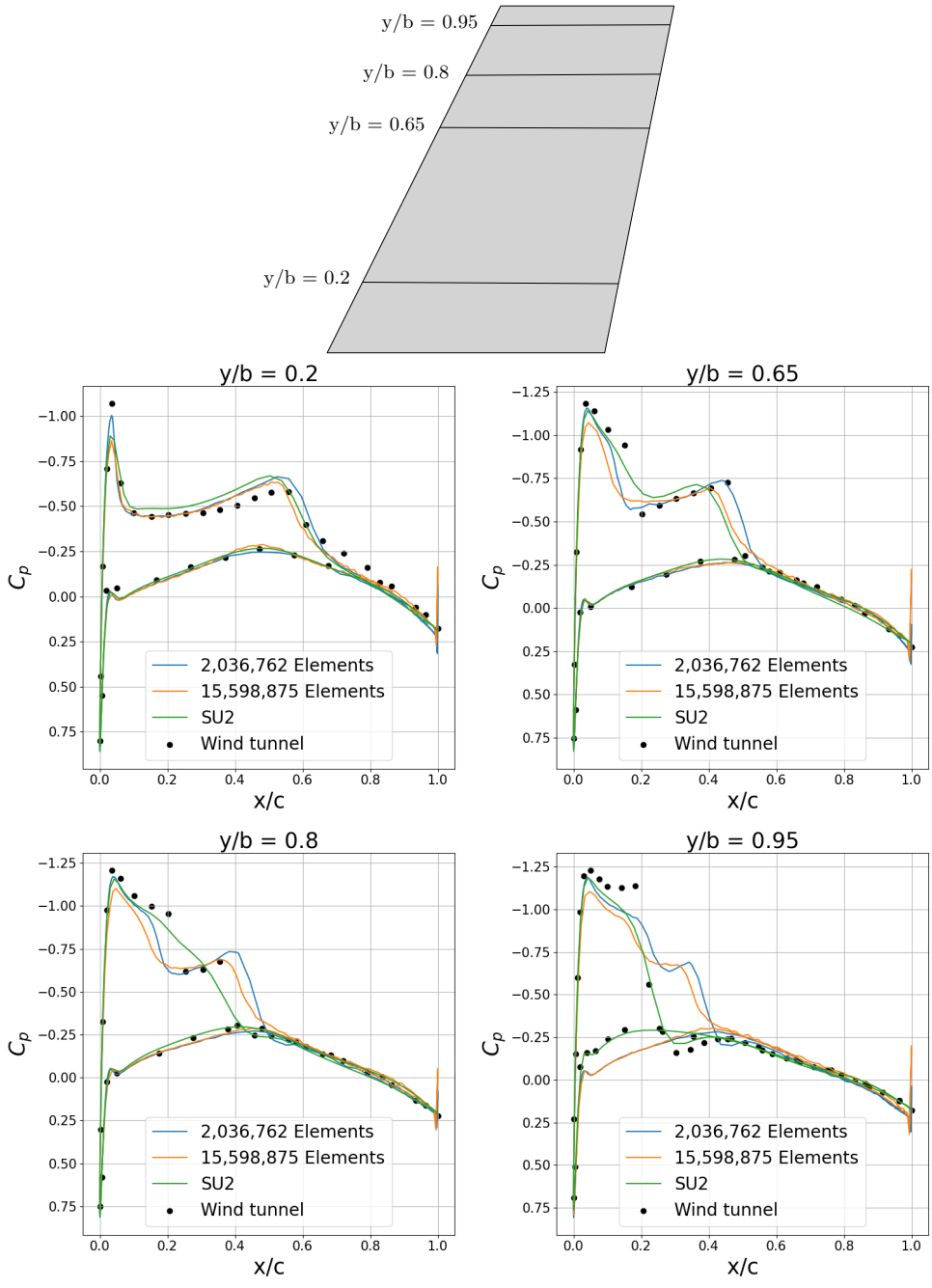


Figure 4.5: Comparison C_p of targeted SU2 results to TiGL mesher results at different sections

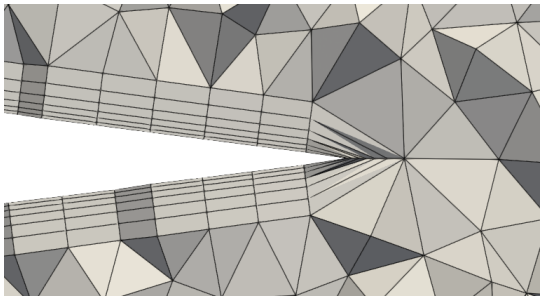


Figure 4.6: Boundary layer at trailing edge

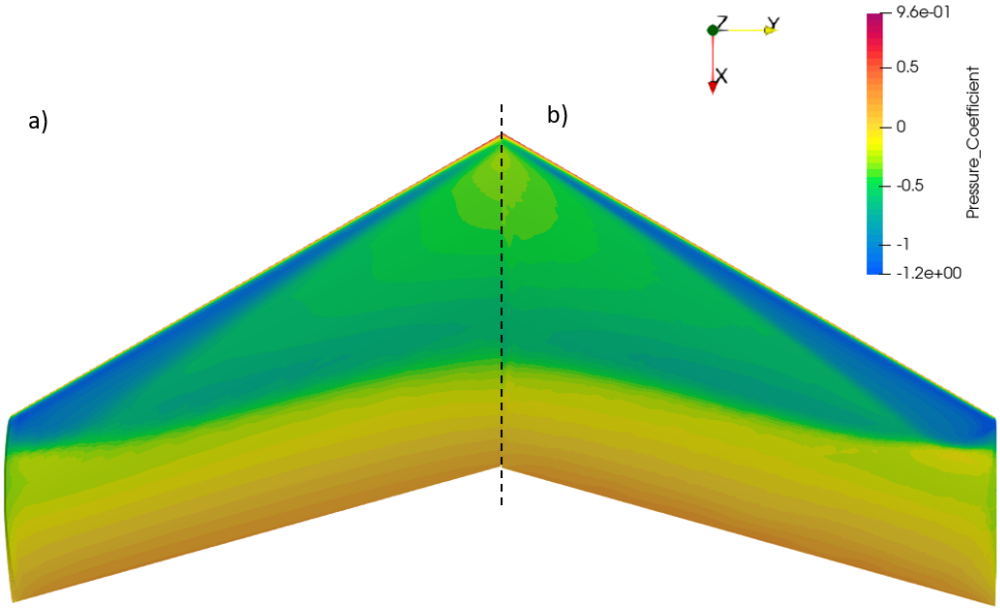


Figure 4.7: Comparison of the pressure coefficient C_p from SU2 (a) and simulation 3 (b)

4.2 DLR-F6 - Generic Transport Configuration

The second test case is the DLR-F6 generic transport configuration. It was created for the second Drag Prediction Workshop (DPW-II) in 2003 [27]. It is an aircraft configuration with only one wing and without a Vertical Tail Plane (VTP) or a Horizontal Tail Plane (HTP) which was analyzed in a wind tunnel and then used to match a simulation to the real results. Ever since it has been a frequently used CFD test object [4, 21, 28, 29, 30, 31]. Figure 4.8 shows a screenshot of the DLR-F6 in the TiGL viewer. The shown configuration is similar to the F6 used in other studies, but it cannot be assured that the geometry is exactly the same. The results from this work are compared to the results from the simulations of Gu et al. [4], Lee-Rausch et al. [28] and the experimental results published by NASA [32]. The used CPACS configuration of the F6 is scaled to the size of a A320 (or D150). In order to use the given simulation parameters and match the wind tunnel model, it must be scaled down by the factor 0.0327. This can also lead to inaccuracies when comparing the models.

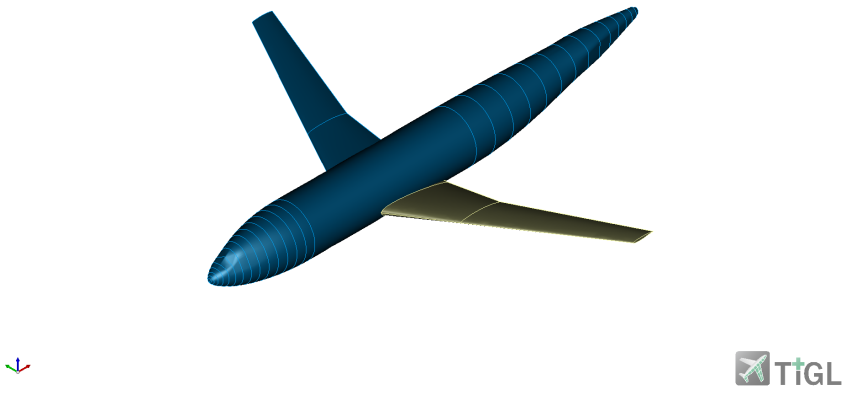


Figure 4.8: The DLR-F6 in TiGL

For the described test case, there are also Euler and RANS simulations conducted, however only the RANS simulations are matched to experimental wind tunnel results. Unlike the ONERA M6 wing, there are no Euler simulation results available for a comparison of the F6. Since the boundary layer is not able to cover an entire aircraft, the mesher is primarily suitable for Euler simulations of a whole aircraft. The RANS simulations are performed to test the capabilities for wing analysis.

The F6 simulations were conducted with different parameters than the ONERA M6 simulations. The parameters for the F6 simulations were taken from the wind tunnel experiments which formed the basis for the DPW-II [27] and are displayed in Table 4.7.

Table 4.7: F6 simulation conditions

Ma	0.75
AOA	0.49°
T	288.15 K
P	101 325 Pa

Eight Euler simulations were conducted to determine the change of C_l and C_d with a rising number of elements. Table 4.8 shows that there is a significant change in C_l and C_d between the simulations for Mesh 1 and Mesh 2 when the number of elements is tripled. Subsequently, the change in C_l and C_d gradually decreases. The last three steps are so close to each other that they are considered as almost equivalent. This concludes the mesh convergence study with more than 200 million elements. To further analyze the convergence, the GCI is calculated. As the convergence is non-monotone, the order of convergence cannot be determined. As described above, $p = 2$ is assumed for the GCI in Equation 2.10. The GCI is displayed in Table 4.9.

In Table 4.9 there are considerable variations in the GCI values. The GCI decreases to 2.1 % for Mesh 2 to Mesh 3 but then increases again to 31.0 % for Mesh 5 to Mesh 6. This can be interpreted as an instability in the simulation results. The small GCI

Table 4.8: F6 Euler simulation

	Elements	Iterations	C_l	C_d
Mesh 1	121 847	474	0.347 28	0.013 606
Mesh 2	361 662	506	0.393 918	0.010 474
Mesh 3	1 336 157	538	0.404 871	0.010 37
Mesh 4	6 609 337	577	0.410 598	0.011 032
Mesh 5	12 122 654	139	0.415 83	0.011 363
Mesh 6	19 369 438	142	0.421 011	0.011 794
Mesh 7	53 026 553	1672	0.430 158	0.012 172
Mesh 8	204 231 783	104	0.429 96	0.012 148

Table 4.9: GCI for F6 Euler simulations

	GCI
Mesh 1 to Mesh 2	64.8 %
Mesh 2 to Mesh 3	2.1 %
Mesh 3 to Mesh 4	10.1 %
Mesh 4 to Mesh 5	18.1 %
Mesh 5 to Mesh 6	31.0 %
Mesh 6 to Mesh 7	10.0 %
Mesh 7 to Mesh 8	0.4 %

from Mesh 7 to Mesh 8 however suggests a good convergence at the end of the study. As explained above, the interpretations are bases on the conducted simulations and cannot be evaluated further.

Figure 4.9 shows the pressure coefficient results of the Euler simulations. Comparing the C_p of the wing to the ONERA M6 wing in Figure 4.3, the distribution can be explained physically. On the upper surface, the C_p has a high negative value near the leading edge and a second smaller peak in the center, both creating the lift of the wing [14]. Towards the trailing edge, the C_p rises positively and ends at zero, which can be observed in the side view. In this view it can also be seen how the C_p behaves around the airfoil, with high values around the leading and trailing edge. The high C_p at the leading edge and nose of the plane can be explained by the air

being compressed at the impact. The high C_p value on the lower side of the trailing edge is related to the trailing edge stall [14].

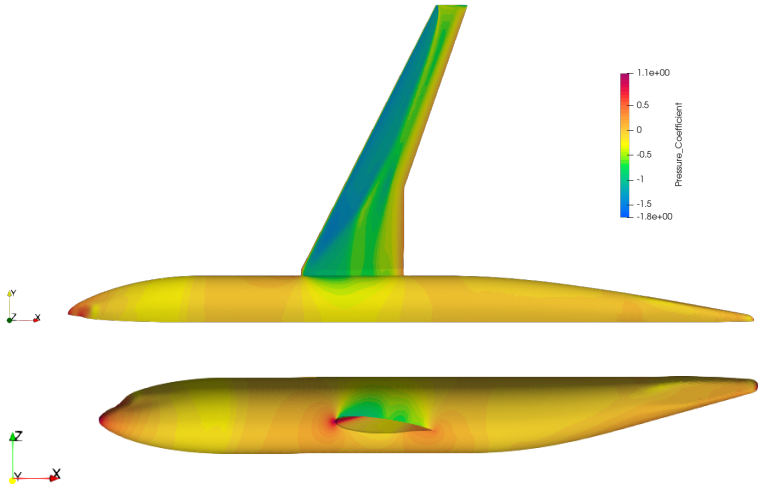


Figure 4.9: Pressure coefficient C_p of F6 in top view and side view

As for the ONERA M6, the RANS simulation of the F6 uses the same simulation conditions as the Euler simulation. This time the Reynolds number is 3e6 and the Reynolds length is 1 m. The used turbulence model again is the SA. [27, 4] Due to the in section with the boundary layer explained in section 3.3, a convergence study of C_l and C_d is not possible. This also applies to matching the target values from the example simulations. The detailed reasons for that are explained later in this section. The pressure coefficient is compared with the pressure coefficient plotted over the chord of different sections.

The wall distance vector y^+ describes the boundary layer thickness. With a given y^+ the thickness of the first layer of the boundary layer for the mesh can be calculated. As described in section 3.4, this is needed for the *boundary_layer* function. In the work of Gu et al. [4, 13], used for comparison, the y^+ for the F6 mesh is 1. With that

information the thickness of the first layer is calculated using the equations presented by Rodriguez in [33]. y^+ is defined as:

$$y^+ = \frac{yu_*}{\nu} \tag{4.1}$$

Since y^+ is known, it can be rewritten to:

$$y = \frac{y^+\nu}{u_*} \tag{4.2}$$

y is the desired distance to the wall and ν the kinematic viscosity of the fluid, which is $14,61 * 10^{-6} \frac{m^2}{s}$ at the selected temperature of 288.15 K. The friction velocity u_* (also called wall friction velocity, wall shear stress velocity and shear velocity) can be calculated with:

$$u_* = \sqrt{\frac{\tau_w}{\rho}} \tag{4.3}$$

The density ρ is $1,225 \frac{kg}{m^3}$ at 288.15K and the wall shear stress τ_w is defined as:

$$\tau_w = \frac{1}{2} C_f \rho U_\infty^2 \tag{4.4}$$

where

$$C_{f_{turb}} = \frac{0.0315}{R_e^{1/7}} = \frac{0.0315}{(3 * 10^6)^{1/7}} = 3.7412 * 10^{-3} \tag{4.5}$$

and $U_\infty^2 = 0,75Ma = 257.25 \frac{m}{s}$. For this case $C_{f_{turb}}$ can be used for C_f [14]. Inserting these values into Equation (4.4), τ_w and u^* can be calculated. These can then be used to calculate y which results in $1.3131 * 10^{-6} m$ for the given conditions.

The nine performed RANS simulations are presented in Table 4.10. The number of elements rises from around 120 000 to over 70 million in course of the simulations. As already mentioned there are always changes in the values of C_l and C_d. Looking at the wing in Figure 4.10, this is probably caused by the vortices that can be observed

at the root, which affects the area between the boundary layer and the fuselage, which together cause problems in the simulation. Since these vortices cannot be observed for the Euler simulation, this problem is probably caused by the cut off boundary layer. But the different turbulence models used for the Euler and RANS simulations can also have an influence. Even if the vortices do not affect the simulation performance of the wing, the overall results such as the displayed C_l and C_d cannot be compared to the example simulations.

Table 4.10: F6 RANS simulation

	Elements	Iterations	C_l	C_d
Mesh 1	122 149	9845	0.150 579	0.293 78
Mesh 2	423 243	10 642	0.178 764	0.022 926
Mesh 3	1 578 022	9353	0.202 475	0.020 086
Mesh 4	3 235 147	14 848	0.199 544	0.017 015
Mesh 5	6 530 112	12 226	0.213 472	0.016 967
Mesh 6	8 761 495	8320	0.206 292	0.018 377
Mesh 7	12 045 742	16 170	0.179 069	0.015 906
Mesh 8	58 795 361	8091	0.169 024	0.016 244
Mesh 9	72 830 128	13 749	0.205 035	0.014 012

As for the previous simulations, the mesh convergence is analyzed for these simulations. The convergence is non-monotone, therefore only the GCI is calculated with Equation (2.10). The values are displayed in Table 4.11. Analyzing the progression of the values, there is a considerable variation. The GCI from Mesh 5 to 6 even rise over 100 % and from Mesh 1 to Mesh 2 as well as Mesh 8 to Mesh 9 over 200 % indicating a significant difference in the C_d values. This leads to the assumptions, that the mesh might have errors and does not distribute the error well. This might be connected to the incomplete boundary layer, as mentioned above. The simulation cannot be assumed as converged, but it might not be possible to complete a convergence study with theses meshes.

For the the comparison of the C_p over the chord of the wing in multiple sections, a

Table 4.11: GCI for F6 Euler simulations

	GCI
Mesh 1 to Mesh 2	214.4 %
Mesh 2 to Mesh 3	26.5 %
Mesh 3 to Mesh 4	74.7 %
Mesh 4 to Mesh 5	1.4 %
Mesh 5 to Mesh 6	115.1 %
Mesh 6 to Mesh 7	170.6 %
Mesh 7 to Mesh 8	3.4 %
Mesh 8 to Mesh 9	268.7 %

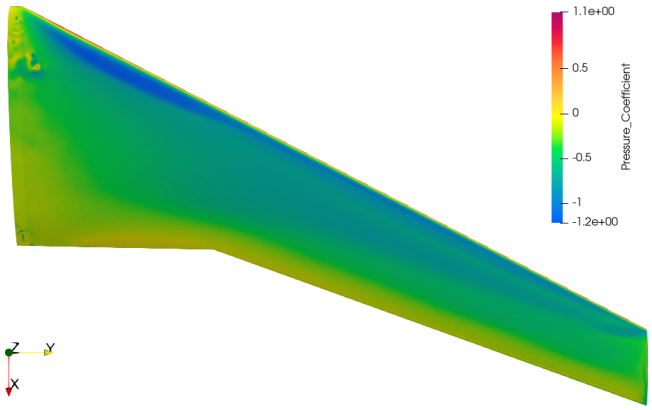


Figure 4.10: C_p of F6 RANS wing in top view

few simulations are selected and compared to the experimental wind tunnel results published by NASA [32]. A quantitative comparison to the the simulation results of Gu et al. [4] cannot be conducted since the exact values are not available. However, these results are considered for the assessment of the results created here.

Figure 4.11 displays the pressure coefficient C_p for seven sections of the F6 wing. Eight sections are marked in the overview at the top left because wind tunnel results are available for these eight sections. However, in the F6 model, the boundary layer does not cover the section at y/b = 0.15, which is why the results at y/b = 0.15 are unsuitable for a comparison and are not displayed. An analysis of such a part of a

wing is given in the next chapter in Figure 4.14. Looking at the remaining seven plots, it becomes evident that the simulations do not reach the target C_p between x/c = 0.2 and 0.4 (or 0.8 till y/b = 0.411) on the upper surface. The exact reason for this cannot be determined in this thesis, but possible reasons are identified. One possibility is the inaccuracy of the geometry and therefore a different result. Another possibility is the simulation environment, as only a few conditions were specified and the overall environment was estimated using the SU2 ONERA M6 simulation as an example. Since both, the fine and the coarse mesh have a very similar progression, the mesh quality is not considered as the main issue. However, it cannot be ruled out, that the general mesh quality prevents a better fitting result in theses simulations. Regardless the ONERA M6 simulation and the tests in the following sections suggest that the meshes can be considered adequate. Considering all other areas of the sections the results match well with the experimental results and the mesh enables a good simulation. The comparison with the lower surface at y/b = 0.847 could not be conducted, because the results are incomplete in [32]. When comparing the created results to the results from Gu et al. [4], their results fit better to the experimental results. However, the authors could not reach an exact fit either. The conditions and the F6 model used by Gu et al. are not specified, therefore a recreation of their simulations cannot be performed to test the created meshes.

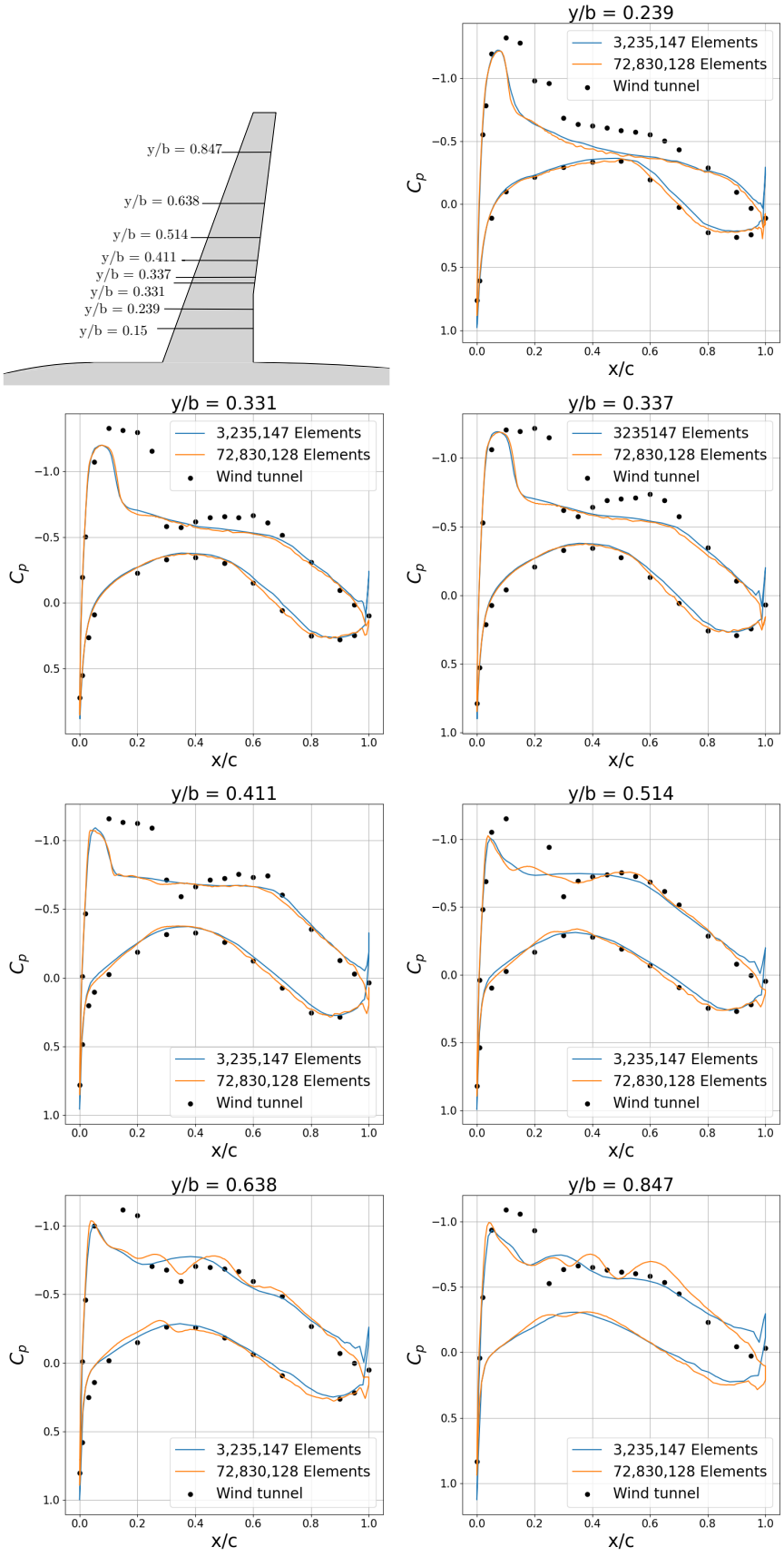


Figure 4.11: Comparison of the pressure coefficient C_p of targeted wind tunnel results to TiGL mesher results in different sections

4.3 DLR D150 - A320 like Configuration

The DLR D150 is the generic test aircraft in TiGL, which is shown in Figure 4.12. It is similar to the A320 in size [34]. Besides a slightly different shape, the only significant geometrical differences to the F6 are the HTP and VTP.

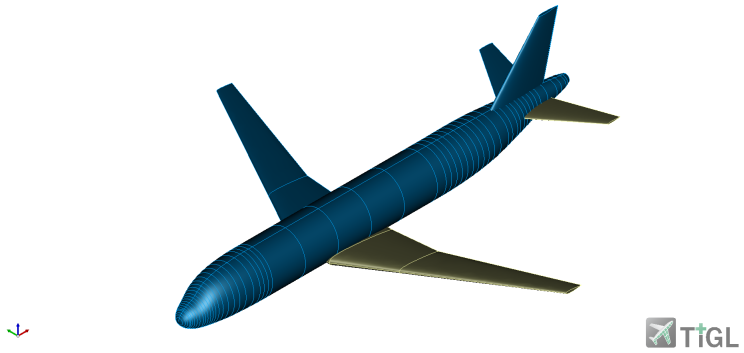


Figure 4.12: The D150 in TiGL

The conditions of the F6 simulations from the previous section are used for the simulations of D150. The simulation of the D150 converged in 284 iterations for the Euler and in 13 758 iterations for the RANS simulation, suggesting that the created mesh is suitable for CFD simulations. The mesh is refined at the leading and trailing edge of the main wing. The surface mesh is shown in Figure 4.13, but due to the small element sizes the refinement on the wing is difficult to recognize.

The Euler and RANS results generally look similar to the F6 results with slight differences due to the different geometry. However, this aircraft is used to show further boundaries of the created RANS meshes in detail. Looking at Figure 4.14, two sections were selected for analyzing the C_p to show the difference between the boundary layer and the area of the wing without the boundary layer. The left plot at y/b = 0.13 shows that without a boundary layer the C_p values are not as smooth as

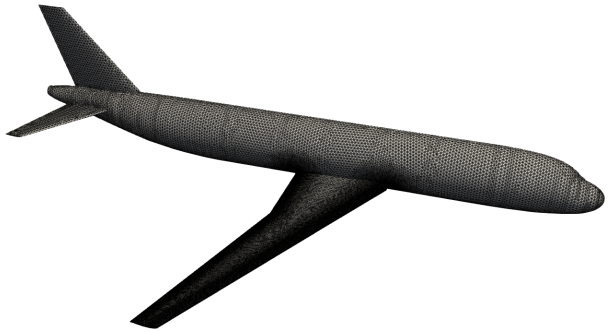


Figure 4.13: D150 surface mesh with refined leading and trailing edge

in the right plot. On the leading edge both plots perform well, but along the chord the left plot (y/b = 0.13) begins to show considerable fluctuations. This also shows the importance of a boundary layer for viscid simulations like RANS in general, as explained in section 2.1.1. Comparing the simulation results of the section with a boundary layer (y/b = 0.2) to the previous simulations of the ONERA M6 and F6, the results are reasonable. Since the progression of the C_p depends on the shape of the airfoil, the results will not match the previous results but there are similarities. The high negative C_p on the upper surface around the leading edge can be observed for the ONERA M6 and the F6 as well. The C_p of the lower surface surpassing the upper surface at y/b = 0.3 is displayed in Figure 4.5 for the ONERA M6. It can be stated that it is possible to performe a wing analysis with a created mesh for different aircraft models.

The D150 is further used to test the robustness of the mesher. Therefore, a variation of its geometry is created by adding another wing above the existing wing an tilting it to merge into the main wing. In addition, the sweep angle of both wings and the HTP is changed to see if this makes a difference in the meshing. Figure 4.15 shows the geometry on the left and the created mesh on the right. The mesher was capable

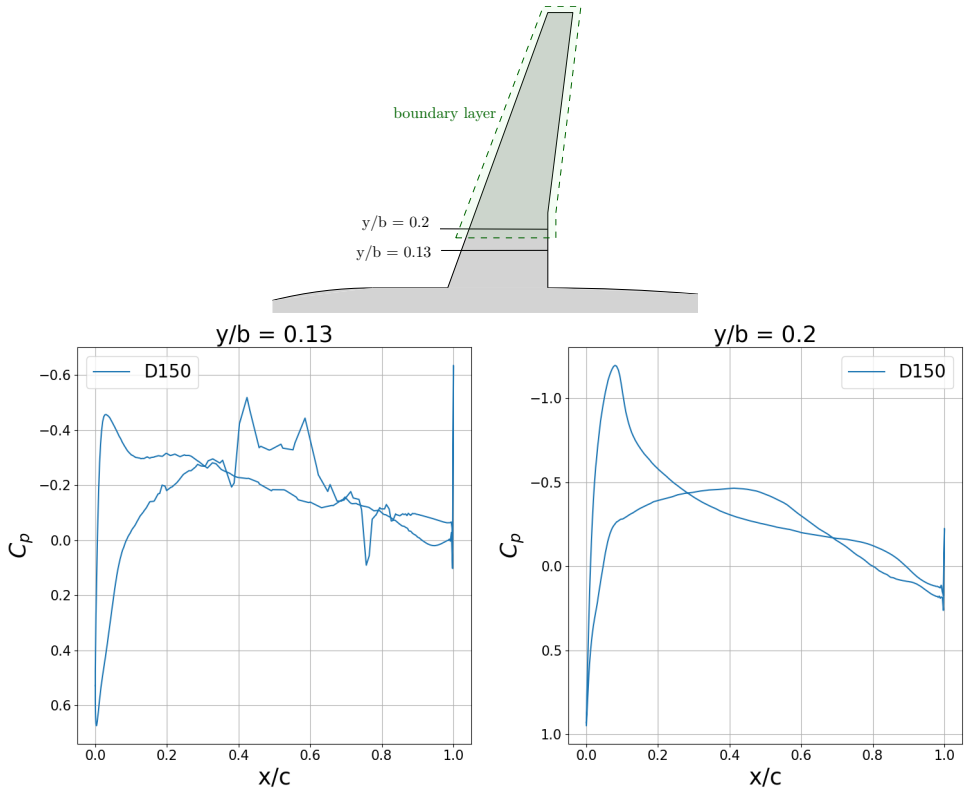


Figure 4.14: Comparison of the pressure coefficient C_p on two sections of D150 with (right) and without a boundary layer (left)

of meshing this configuration, but the refinement performance of the leading and trailing edge was influenced by the merging of the wings. The segment of the upper wing which merges with the original wing loses the information about these edges and therefore they are only refined in the other segments. The changed sweep angle of the wings had no influence on the meshing results. The Euler simulation was also performed, which converged, proving that the mesh quality is sufficient. Since the geometry is made up, an aerodynamic analysis would be unnecessary.

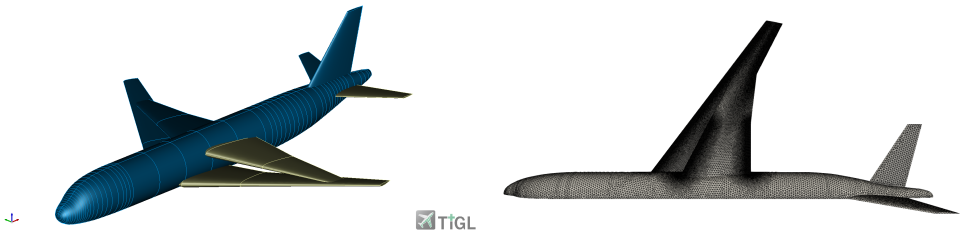


Figure 4.15: D150 variation in TiGL (left) and as mesh (right)

4.4 D250 - A330-200 like Configuration

The D250 is a larger aircraft similar to the long range aircraft A330-200 and features winglets [35]. A winglet is a small wing-like surface which is positioned at the wing tip to reduce the drag coefficient. These winglets are often nearly vertical. The main winglets ar located above the wingtip, as in the case of the D250. [36] The D250 is used to test whether larger aircraft and winglets would work with the mesher. As with the other test cases, the surface mesh did not cause any problems. However, due to the geometry of a wing with a winglet, the boundary layer can intersect itself, which leads to problems. This happens when trying to create a mesh for a RANS simulation. Therefore only Euler simulations could be performed. Figure 4.16 shows the D250 in TiGL next to the surface C_p of the Euler simulation. The winglet can be observed at the tip of the wings. The simulation results show that the mesher is capable of creating unstructured meshes for larger aircraft with special appliances such as winglets.

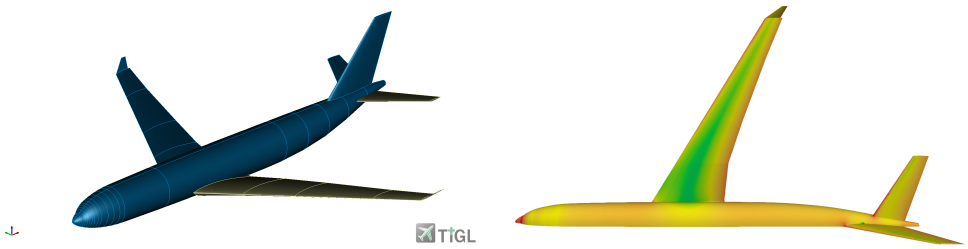


Figure 4.16: The D250 in TiGL (left) and the pressure coefficient C_p of the D250's Euler simulation (right)

4.5 Blended Wing Body

The BWB is an aircraft without a fuselage. In this case it consist of only one wing which is shaped to portrait a BWB. The example file originates from a DLR project from around 2011 [37]. A CFD simulation is performed to test the mesh. Figure 4.17 shows the BWB in TiGL.

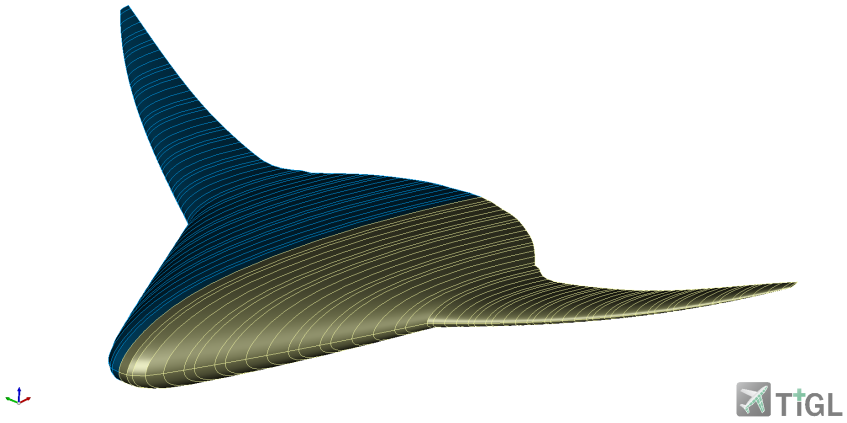


Figure 4.17: The BWB from the TiGL examples

An Euler and a RANS simulation are conducted for the BWB. However, the aerodynamic capabilities are uncertain, a convergence study is unnecessary. Since the BWB does not have a fuselage, the boundary layer can cover the entire aircraft as with the ONERA M6 wing. In Figure 4.18 the Euler (left) and RANS (right) results are

displayed next to each other. The comparison shows that the results are close to each other, suggesting that the created mesh enables accurate simulations. This can also be observed for the ONERA M6. Although the comparison was not made directly, the Euler and RANS results from the Figures 4.3 and 4.7 are similar, which leads to the conclusion that the mesher is capable to create an accurate mesh for one wing configurations. However, the problem described in section 3.3 between the boundary layer and the symmetry plane persist for the BWB as well.

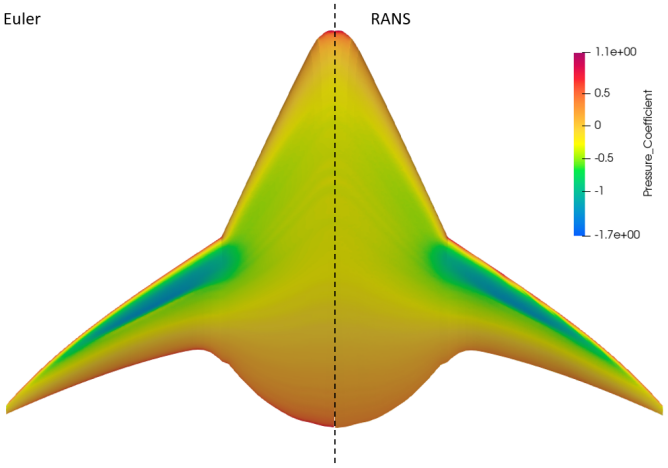


Figure 4.18: Comparison of the Euler (left) and RANS (right) simulations of the BWB

4.6 Concorde

The Concorde was a well-known supersonic commercial airliner which was operated between 1976 and 2003 [38]. As with the BWB, the TiGL examples also contain a Concorde CPACS configuration which is probably aerodynamically incorrect (Figure 4.19). Therefore, the same tests as for the BWB are performed. But since the results of the simulation are not meaningful for this work, they are not analyzed here.

However, the C_p analysis of the RANS simulation again shows the vortices created by

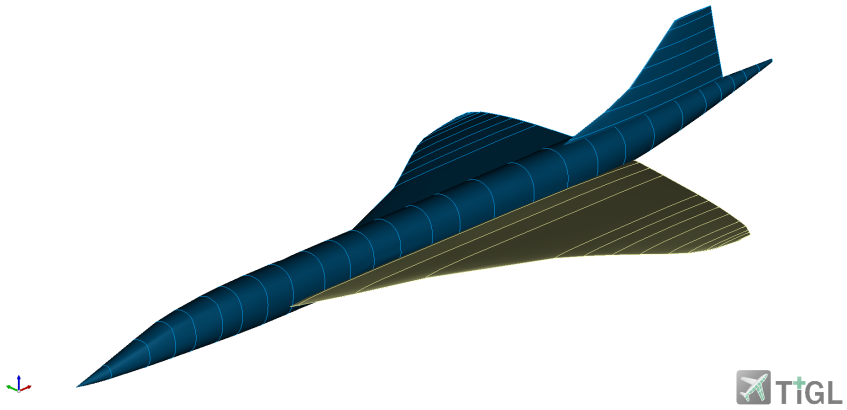


Figure 4.19: The Concorde from the TiGL examples

the boundary layer workaround which was implemented in this thesis. The long chord of the delta wing design of the Concorde offers the opportunity for many vortices to form. Figure 4.20 shows the C_p of the Concorde with many vortices and without a legend, since the values are irrelevant.

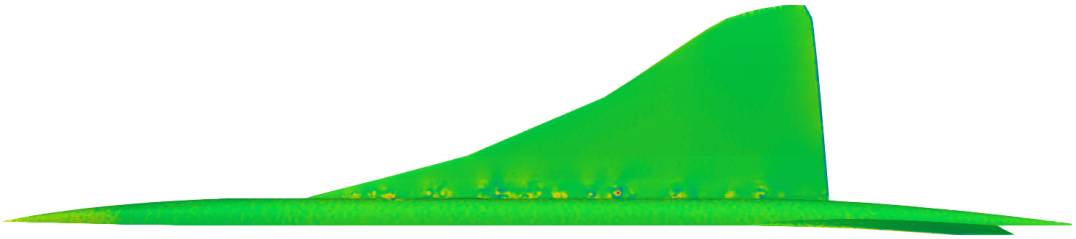


Figure 4.20: Vortices in the pressure coefficient C_p representation of a Concorde RANS simulation

5 Conclusion

In this thesis, a meshing tool for the aircraft predesign with TiGL was created and tested. In order to do this, the current state of the art was analyzed. It was shown that various approaches already exist to reach this goal. Most of the approaches were part of a OAD workflow. This thesis can be used to build a TiGL OAD in the future. The different approaches showed different advantages and disadvantages. The aim of this thesis was to find or develop a TiGL and CPACS based approach that can be integrated with internal workflows within the DLR. Additionally, due to the research-oriented nature of the DLR, the proposed software solution needed to be open source. While two of the analysed existing approaches would fit the requirements for TiGL and CPACS they were deemed unsuitable due to the lack of Gmsh integration and the fact that neither is fully open source. As a consequence a new mesher was created for this thesis.

The fundamental research showed that Euler and RANS simulations proved to be the most suitable CFD simulations for testing the mesher, since they are carried out as part of a predesign optimization. It has become evident that Euler simulations require an unstructured meshed farfield and a RANS simulation requires a structured boundary layer.

The created mesher consists of three modules, the TiGL module, the Gmsh module and the mesher module. The TiGL collects all important information about the aircraft in the so-called TiGL map. This map is structured by the UID of the components and

saves for example the start and end points of the edges which resemble the leading edge of the wing. Subsequently, this information is matched with the Gmsh indices in the Gmsh module. The indices or tags are the only way to identify entities in Gmsh and are therefore crucial for local mesh refinements. In the meshing module, important Gmsh functions are wrapped and used to create an aircraft specific mesh generator. The required structured boundary layer could only be created on most parts of the wing due to Gmsh limitations.

The mesh generator is tested to confirm that the meshes are suitable for CFD simulations. Six different aircraft configurations are analyzed for this purpose. The most important ones are the ONERA M6 wing and the DLR-F6 generic transport configuration. These are well-known test objects for CFD simulations. Therefore many results are available for comparison. Since the CFD tool SU2 used the ONERA M6 in its documentation, even the configuration files for the simulations were available, which enabled a comparison based solely on the mesh quality. The results for the DLR-F6 came from the DPW-II in 2003. With the created meshes it was possible to recreate the Euler and RANS simulations of the ONERA M6. For the F6, only RANS data were available from experimental wind tunnel experiments and simulations in other papers [4, 32]. These results could not be reached completely with the conducted simulations. On the upper surface behind the leading edge, the pressure coefficient had a different course. This however could also be affected by a slightly different geometry and differences in the simulations conditions. The general lift and drag coefficient could also not be compared due to the boundary layer issues. Besides these issues, an analysis of the wings can be conducted after the RANS simulations, considering the effect the absence of the boundary layer has at the joint of the wing to the fuselage. in addition, the mesh convergence was evaluated by calculating the GCI. These evaluations showed, that the two ONERA M6 mesh studies and the F6 study without the boundary layer can be interpreted as converged. The interpretation of

the GCI can be taken as trend rather than proof. The further simulated test aircraft confirmed the results and proved a limited robustness of the meshes considering the boundary layer. The D250 configuration featured winglets, which also led to a self-intersecting boundary layer. A similar problem was observed in other aircraft models tested, where one wing intersected another wing.

In view of the results, it can be stated that the goal of this thesis was achieved. A robust mesh generator for unstructured, aircraft specific CFD meshes was created and tested. Gmsh proved to be a generally reliable tool for robust mesh generation and the aircraft specific adjustments or refinements were suitable for the presented test cases. The evaluation was done with Euler simulations which proved to be the most suitable for the created meshes. The creation of a structured boundary layer with Gmsh, however, caused problems, which led to a limitation of robustness. The boundary layer proved to be suitable for configurations consisting of only one wing such as the ONERA M6 or the BWB, but entire aircraft models can only be analyzed partially. The effort to integrate CEASIOMpy's module Pentagrow to solve this problem was not successful during this thesis, since its installation process is currently flawed. If Pentagrow becomes available as a standalone module in the future, it should be integrated to provide a fully functional boundary layer. Pentagrow uses a Gmsh surface mesh as input and extrudes the boundary layer. Therefore, the entire mesher which proved to be robust, could be preserved. This would transform the mesher to a fully functional tool for the aerodynamic preseign with TiGL. It is recommended to investigate whether a 100 % structured mesh can be created by only using OpenCASCADE. But the conducted tests showed that the workflow that was created and presented in this thesis has big advantages and enables users to quickly conduct a simulation after modifying a CPACS file. This makes it possible to optimize an aircraft in the predesign phase and saves a lot of time. As also described in NASA's CFD-Vision 2030 study [10] automated meshing is a key component of

future OAD workflows. Looking ahead there should be a more detailed analyzes of the mesh convergence. The conducted analyses by calculating the order of convergence and the GCI is one approach to evaluate the mesh convergence. This approach can be used to conduct further simulations and refining the meshes more uniformly to create results, which are more comparable. The attempt was made to also make a low-speed stall analysis of the ONERA M6 wing by adjusting the AOA over 20 simulation runs. This could not be completed in this thesis and should be considered in future work. The here created meshes, however, where able to recreate experimental and simulation data to an adequate level for the predesign. This work enables users to quickly create meshes with sufficient quality from CPACS configurations and perform CFD simulations that show a reasonable result. This leads to a significant time reduction in an optimization process.

Bibliography

[1] Martin Siggel et al. TiGL: An Open Source Computational Geometry Library for Parametric Aircraft Design. In: *Mathematics in Computer Science* 13 (Sept. 2019).

[2] R. Schwarze. CFD-Modellierung: Grundlagen und Anwendungen bei Strömungsprozessen. Springer Berlin Heidelberg, 2012.

[3] Christophe Geuzaine and Jean-François Remacle. Gmsh: A three-dimensional finite element mesh generator with built-in pre-and post-processing facilities. In: 79 (Nov. 2008).

[4] Xiangyu Gu et al. An automated CFD analysis workflow in overall aircraft design applications. In: *CEAS Aeronautical Journal* 9 (2018), pp. 3–13.

[5] Mengmeng Zhang et al. Aircraft Geometry and Meshing with Common Language Schema CPACS for Variable-Fidelity MDO Applications. In: 5 (Apr. 2018).

[6] Airinova CFS Engineering. CEASIOMpy. `https://github.com/cfsengineering/CEASIOMpy`. 2024.

[7] Irian Ordaz et al. Automated Tetrahedral Mesh Generation for CFD Analysis of Aircraft in Conceptual Design. In: *52nd Aerospace Sciences Meeting*.

[8] Michael Turner et al. A framework for the generation of high-order curvilinear hybrid meshes for CFD simulations. In: *Procedia Engineering* 203 (2017). 26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain, pp. 206–218.

[9] Rajesh Bhaskaran and Lance Collins. Introduction to CFD basics. In: *Cornell University-Sibley School of Mechanical and Aerospace Engineering* (2002), pp. 1–21.

[10] Jeffrey P Slotnick et al. CFD vision 2030 study: a path to revolutionary computational aerosciences. In: (2014).

[11] Gianfranco La Rocca. Preliminary Design for Flexible Aircraft in a Collaborative Environment. Jan. 2013.

[12] Vladimir D. Liseikin. Grid Generation Methods. Springer Cham, 2017.

[13] Gu Xiangyu. Application of Computational Aerodynamic Analysis and Optimization in a Multi-Fidelity Distributed Overall Aircraft Design System. PhD thesis. RWTH Aachen University, 2017.

[14] Snorri Gudmundsson. General aviation aircraft design: Applied Methods and Procedures. Butterworth-Heinemann, 2013.

[15] Timothy J. Baker. Mesh generation: Art or science? In: *Progress in Aerospace Sciences* 41 (2005), pp. 29–63.

[16] P.J. Roache. Verification and Validation in Computational Science and Engineering. Hermosa Publishers, 1998.

[17] L. F. Richardson. The Approximate Arithmetical Solution hy Finite Differences o f Physical Problems involving Differential Eq, with an Ap Stresses in a Masonry Dam. In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character Volume 210* (1911), pp. 307–357.

[18] Marko Alder et al. Recent Advances in Establishing a Common Language for Aircraft Design with CPACS. In: (Jan. 2020).

[19] P.L. George and H. Borouchaki. Delaunay Triangulation and Meshing: Application to Finite Elements. Hermès, 1998.

[20] Joachim Schoeberl. NETGEN An advancing front 2D/3D-mesh generator based on abstract rules. In: *Computing and Visualization in Science* (July 1997), pp. 41–52.

[21] Thomas D. Economon et al. SU2: An Open-Source Suite for Multiphysics Simulation and Design. In: *AIAA Journal* 54.3 (2016), pp. 828–846.

[22] Onera. ONERA-M6 Wing, Star of CFD. 2013. URL: `https://www.onera.fr/en/news/onera-m6-wing-star-of-cfd` (visited on 06/13/2024).

[23] Adavisory Group for Aerospace Research and Development. Experimental data base for Computer Program Assessement - Report of the Fluid Dynamics Panel. AGARD, 1979.

[24] SU2. Inviscid ONERA M6. 2020. URL: `https://su2code.github.io/tutorials/Inviscid_ONERAM6/` (visited on 07/17/2024).

[25] SU2. Turbulent ONERA M6. 2020. URL: `https://su2code.github.io/tutorials/Turbulent_ONERAM6/` (visited on 07/17/2024).

[26] NPARC Alliance Validation Archive. ONERA M6 Wing. 2021. URL: `https://www.grc.nasa.gov/www/wind/valid/m6wing/m6wing.html` (visited on 06/25/2024).

[27] Kelly Laflin et al. Summary of Data from the Second AIAA CFD Drag Prediction Workshop. Jan. 2004.

[28] EM Lee-Rausch et al. Transonic drag prediction on a DLR-F6 transport configuration using unstructured grid solvers. In: *Computers & fluids* 38.3 (2009), pp. 511–532.

[29] Numerical simulation of DLR-F6 wing-body flow field in ground effect. In: *Computers Fluids* 245 (2022), p. 105576.

[30] Qiuya Zheng et al. Transonic Drag Prediction on a DLR-F6 Transport Configuration. In: *Advanced Materials Research Vol. 566* (2012).

[31] Ralf Rudnik et al. Experimental Investigation of the Wing-Body Juncture Flow on the DLR-F6 Configuration in the ONERA S2MA Facility. In: (June 2009).

[32] Neal T. Frink. 2nd AIAA CFD Drag Prediction Workshop. 2003. URL: `https://aiaa-dpw.larc.nasa.gov/Workshop2/workshop2.html` (visited on 06/25/2024).

[33] Sal Rodriguez. Applied Computational Fluid Dynamics and Turbulence Modeling. Springer Cham, 2019.

[34] Wolf R. Krüger and Thomas Klimmek. Definition of a Comprehensive Loads Process in the DLR Project iLOADS. In: *Deutscher Luft- und Raumfahrtkongress 2016*. 2016.

[35] Matthias Schulze et al. Parametric modelling of a long range aircraft under consideration of engine wing integration. In: *DLRK 2019 - Deutscher Luft- und Raumfahrtkongress*. 2019.

[36] R. T. Whitcomb. A design approach and selected wind tunnel results at high subsonic speeds for wing-tip mounted winglets. In: *NASA Technical Note* (1976).

[37] Joerg Fuchte et al. Optimization of revenue space of a blended wing body. In: *29th Congress of the International Council of the Aeronautical Sciences, ICAS 2014* (Jan. 2014).

[38] I. I. Greshnikov et al. Advanced Supersonic Passenger Aircraft Cockpit Technologies. In: *Recent Developments in High-Speed Transport: Selected Contributions from International Conference on High-Speed Transport Development 2022*. Singapore: Springer Nature Singapore, 2023, pp. 13–21.

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer oder der Verfasserin/des Verfassers selbst entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Ort, Datum	Unterschrift