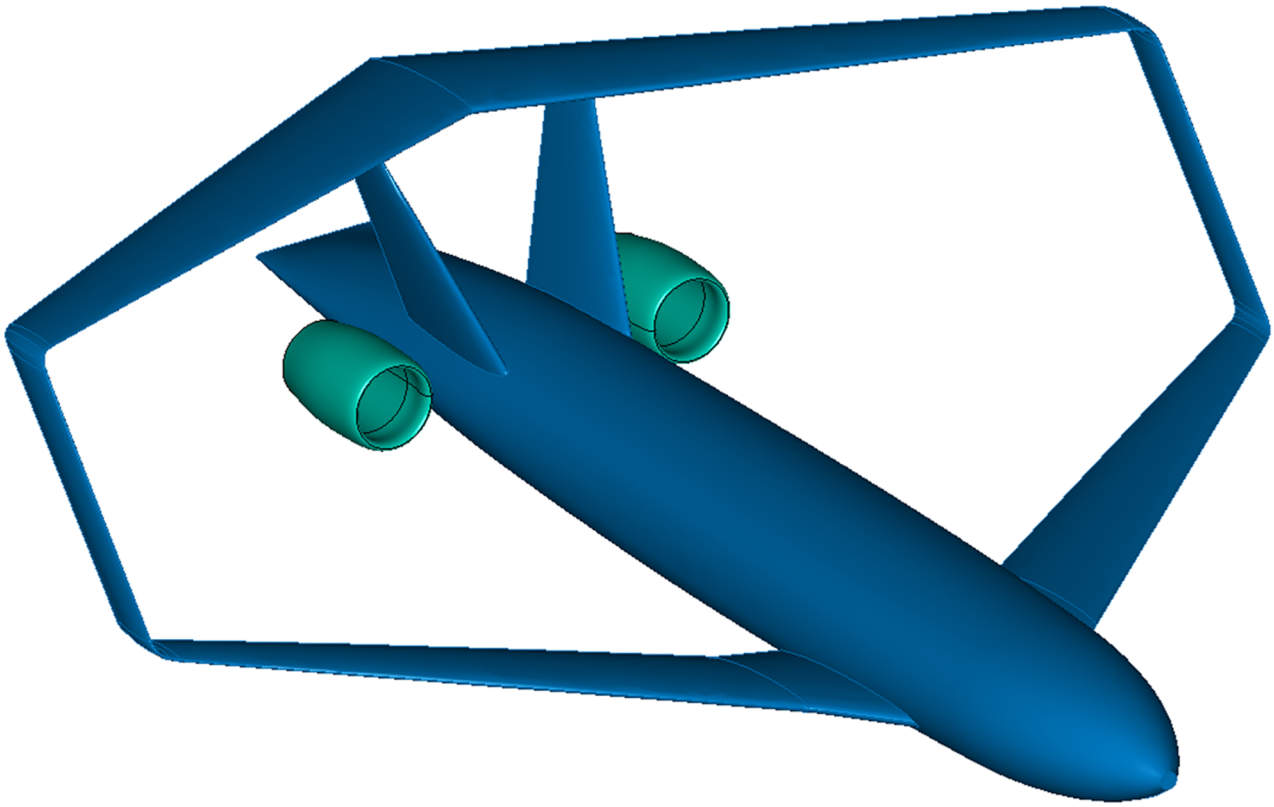
Aerodynamic Analysis of Engine Integration during the Preliminary Design Phase

C.A.E. Heimans

Technische Universiteit Delft



TU Delft Delft University of Technology

Challenge the future

Aerodynamic Analysis of Engine Integration during the Preliminary Design Phase

by

C.A.E. Heimans

in partial fulfillment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at the Delft University of Technology,
to be defended publicly on Friday, February 26, 2021 at 09:30.

Equivalent word count:	24990		
Student number:	4295919		
Project duration:	September, 2019 – February, 2021		
Thesis committee:	Prof. dr. ir. L.L.M. Veldhuis	TU Delft,	committee chair
	Dr. ir. G. La Rocca,	TU Delft,	supervisor
	Dr. ir. A.H. van Zuijlen,	TU Delft	
	Ir. R.J.M. Elmendorp,	TU Delft	

TU Delft Delft University of Technology

Preface

With this report and accompanying defence, I am completing my Aerospace Engineering master. First of all, I would like to thank my supervisor dr. Gianfranco La Rocca for this assignment and his guidance during my thesis. I would also like to thank prof. Veldhuis, dr. Van Zuijlen, and ir. Elmendorp for being part of my thesis committee.

To my friends, thank you for supporting me and making my study time a good one. Thanks to my colleagues of the ParaPy team too, who helped me with brainstorming on solutions. Finally, I want to thank my family a lot, for supporting me to go to Delft to fulfil a dream, becoming an aerospace engineer.

Colin Heimans
Delft, February 2021

Summary

To reduce the environmental impact of aviation, new aircraft configurations are investigated. One of these researches is the PARSIFAL project, that investigates an efficient box-wing aircraft for passenger transport. In previous work, an engine sizing tool is developed to design the engines of the PARSIFAL aircraft. The tool developed as part of the Multi-Model Generator (MMG), a knowledge based engineering application developed in the Flight Performance section. This work also included an engine integration study, which showed that a fuselage mounting is preferred over a wing mounting. However, the integration study lacked the desired design sensitivity for small design changes, e.g. engine diameter and length. Because of this lack of design sensitivity, the question whether the engine installation following from this work is ideal remains open.

Therefore, in this thesis a best practice for engine integration studies during the preliminary design phase is investigated. The found best practice is then used to re-asses the engine installation by aligning the engines with the local flow. The best practice is also employed to asses under wing engine installations, to establish the potential use of the practice for this type of installations.

Before the trade-off study for a best practice could be performed, changes to the MMG had to be made. Three primitives, the building blocks that create an aircraft geometry, had to be adapted. A new wing primitive is introduced, that uses common geometric features as inputs, removes the dihedral limitation of the current primitive, and introduces n^{th}-order rails, that define the planform. However, with this replacement the direct control of stagger in box-wings is lost, if they are modelled as a single shape. A through flow nacelle (TFN) geometry is introduced, since the previous practice of simulating this type of nacelle by setting boundary conditions on a full engine model resulted in simulations that do not stroke with physics. Moreover, a new pylon primitive is developed to enlarge the pylon design space. This new primitive is able to model realistic pylon planforms. However, the resulting 3d shape lacks smooth curvature for highly swept pylons.

In search of a best practice, textbook analyses, 3d panel solvers, and combinations of the two are compared. To fully assess an engine integration, information on lift, drag, and pitching moment is required. Since textbook methods only provide information on drag, these are not satisfactory on their own. Nonetheless, these methods can be used to predict the drag addition of a pylon, when only airframe and nacelle are simulated.

The used panel solvers are FlightStream and VSAERO. In both solvers, two analysis practices are tested. The first being a simulation including airframe, nacelle, and pylon and the second being a simulation including airframe and nacelle complemented with a textbook correction for the pylon drag. In case of VSAERO, this is done with TFNs and powered nacelles. Following a trade-off, using speed, accuracy of a clean aircraft simulation, ease-of-use, accuracy of engine installation effects prediction, and design sensitivity as criteria, a simulation with FlightStream of airframe and nacelle with a textbook addition of pylon drag proved to be the best analysis practice.

This best practice is employed to re-asses the engine installation of the PARSIFAL aircraft. While the location is fixed, since it aligns with the structure inside the airframe, the orientation is changed to align with the local flow. This alignment results in a 3% decrease of aerodynamic efficiency loss and a 17% reduction of pitching moment increase, with respect to the original engine orientation.

In a second study, the potential of the best practice to analyse under wing engine installations is tested by changing the location of an engine under the rear wing of the PARSIFAL box-wing. Also for this type of installation, the analysis returns trends for changing engine location. These results show that with proper placement, the same aerodynamic efficiency loss can be obtained as for fuselage installations.

Contents

List of Figures

List of Tables

Nomenclature

Symbols

A	Cross sectional area	[m^2]
A	Wing aspect ratio	[-]
$\bar{c}$	Mean aerodynamic chord	[m]
$C_D S$	Drag area	[m^2]
C_f	Flat plate skin friction coefficient	[-]
C_{D_0}	Skin friction drag coefficient	[-]
C_{D_p}	Pressure drag coefficient	[-]
$C_{L_{\min}}$	Lift coefficient for minimal drag	[-]
d	Diameter	[m]
e	Wing efficiency factor	[-]
f	Engine fineness factor	[-]
FF	Form factor	[-]
h/c	Vertical engine position, normalized with local chord	[-]
k	Skin roughness height	[m]
k_n	Factor for the effect of nacelles on aerodynamic centre	[-]
l	Characteristic length	[m]
l	Length	[m]
M	Mach number	[-]
P	Static pressure	[Pa]
Q	Interference factor	[-]
R	Specific gas constant	[$J\,kg^{-1}\,K^{-1}$]
Re	Reynolds number	[-]
S_{ref}	Reference area	[m^2]
S_{wet}	Wetted area	[m^2]
T	Static temperature	[K]
t/c	Airfoil relative thickness	[-]
V	Fluid speed	[$m\,s^{-1}$]
$(x/c)_m$	Relative location of airfoil maximum thickness	[-]
x/c	Horizontal engine position, normalized with local chord	[-]

x_{ac}	Aerodynamic centre location	[m]

Greek Symbols

α	Angle of attack	[°]
β	Cowl forebody length fraction	[-]
η	Coordinate along wing reference rail	[-]
γ	Heat capacity ratio	[-]
Λ	Sweep angle	[rad]
μ	Fluid dynamic viscosity	[Pa s]
ρ	Fluid density	[kg m^{-3}]

Subscripts

β	Boat tail
∞	Free stream flow
py	Pylon
c	Component
f	Engine fan
g	Engine core (gas generator)
h	Engine highlight
n	Nacelle
o	Flow capture area
p	Engine plug

1

Introduction

Currently, air traffic experiences two conflicting trends. On one hand, the demand of air transport is increasing ever since air transport became available to the broad public, while, on the other hand, air traffic has a large negative impact on climate change because of pollutants created by aircraft. To solve this conflict, the European government, aircraft industry partners, and academia joined together in the *Advisory Council for Aeronautics Research in Europe* (ACARE). This group set up goals to work towards solving the mentioned conflict. The goals, listed in FlightPath 2050, can be summarized as handling at least 25 million flights per year with a 75% reduction in CO_2 emissions and a 90% reduction in NO_x emissions per passenger kilometre. These goals also include a 65% reduction of perceived noise. The reductions are relative to the capabilities of typical new aircraft in 2000 [1]. Next to increasing the efficiency of conventional tube-and-wing aircraft, other configurations are investigated to reach the goals set in FlightPath 2050.

One of these investigations considers the PrandtlPlane, that builds upon the *best wing system* proposed by Ludwig Prandtl in 1924 [2]. More recently, Frediani [3] has created designs using this system. Frediani's designs led to an European research project, *Prandtlplane ARchitecture for the Sustainable Improvement of Future AirpLanes* (PARSIFAL), to design a PrandtlPlane for 250 passengers used for short range missions [4]. A render of this concept is shown in Figure 1.1.

The PARSIFAL project is divided into work packages that each consider a part of the design and analysis of the PARSIFAL aircraft. TU Delft was, amongst others, responsible for Work Package 7, that is about the design and integration of the propulsion and fuel system. The objectives of this work package are:

- Definition of the propulsion system layout (number of engines, location on the airframe, fuel system);
- Preliminary design and analysis of the propulsion system;
- Investigation into potential for using very large bypass ratio turbofan engines on the rear wing.

Figure 1.1: PARSIFAL PrandtlPlane concept render.[1]

[1]Obtained from https://parsifalproject.eu on October 26, 2020.

1.1. Previous work

Within the scope of the PARSIFAL Work Package 7, Proesmans made a tool that, sizes and analyses a turbofan engine [5]. This tool, called GTPy, sizes the engine's internal components based on a 0d-performance analysis and creates the geometry of the engine's nacelle.

Next to the engine sizing, Proesmans also investigated the aerodynamic installation effects of the engines on the PARSIFAL aircraft. The aerodynamic studies are performed using the 3d panel solver VSAERO, with a model containing airframe, pylons, and nacelles. The engines are modelled as powered engines, by setting its fan and exhaust velocities.

The geometry is modelled using a primitive in the Multi-Model Generator (MMG). This is a knowledge based engineering (KBE) application that uses primitives and capability modules to create and analyse aircraft models. This application is originally described by La Rocca [6]. The primitives are the components, i.e. wings, fuselages, engines, and pylons, that together form an aircraft. They allow to model many different aircraft configurations in different parametric variants. The capability models perform analyses on the generated aircraft geometry. These analyses can be built-in calculations or third-party tools like aerodynamic and finite element method solvers. In the latter case, the capability module generates input files to use in the external tool.

Proesmans' analysis was firstly used to find a more preferable engine location, either on the fuselage near the root of the vertical tail or under the rear wings. It was observed that the fuselage installation affected the aerodynamic performance the least. Secondly, a study to find the effect of changing the engine's overall pressure ratio (OPR) and bypass ratio (BPR) on the aerodynamic performance was conducted, this resulted in inconsistent trends. Possible sources for this inconsistency are the exhaust flows of the engines and the interference effects of the airframe, engine, and pylon. Analytical textbook methods showed to be a more consistent predictor of the drag added by the engines. However, no lift or moment coefficients are obtained with this method.

1.2. Objective

The results of Proesmans' integration study left the question open whether the used practice is the proper one to investigate the aerodynamic integration effects during preliminary design. Since the study showed a clear difference between fuselage and wing installation, but was not able to capture small design changes, the engine might not be positioned ideally in its current position.

Therefore, the goal of this research is to re-evaluate the engine position of the PARSIFAL aircraft by finding a best practice for aerodynamic engine integration studies during the preliminary design phase and using it for an engine integration study on the PARSIFAL aircraft. In this thesis, the following research questions are considered.

1. Which results are required from an aerodynamic engine integration study?
2. What practices can be considered for aerodynamic engine integration studies in the preliminary design phase?
   (a) What methods, i.e. the type of analysis, should be considered?
   (b) Which aircraft components should be included in the integration analysis?
   (c) What type of engines should be used for analysis during the preliminary design phase?
3. What changes should be made to the MMG to analyse the selected practices?
4. What criteria should be used in a trade-off study to compare different analysis practices?
5. What is the best practice for aerodynamic engine integration studies in the preliminary design phase, following from the trade-off study?
6. To what extent can the aerodynamic performance of the PARSIFAL PrandltlPlane be improved by changing the engine's orientation, using the found best practice?
7. To what extent can the best practice provide trends with changing engine position for under wing installations?

The first six questions are necessary to fulfil the research objective. The seventh is added to assess the applicability of the best practice for configurations other than the PARSIFAL fuselage mounted engines.

1.3. Outline

To be able to find a best practice, changes to the primitives of the MMG were required. These changes are described in Chapter 2. In Chapter 3 a best practice for engine integration studies is searched. This is done by firstly selecting methods and tools to use for a trade-off, which are further introduced before performing the trade-off itself. After a best practice is found, it is put to use in Chapter 4 by the investigation of fuselage and wing mounted engine orientations and positions. To finalise this report, conclusions and recommendations are presented in Chapter 5.

2

Geometry primitives

For the generation of geometry and pre-processing it before analyses, the MMG is used. This KBE application is developed at the Flight Performance section. Over time, different KBE platforms have been used for the development of the MMG. Currently, the underlying platform is ParaPy.

In this application, the geometry is created by primitives, which can be seen as parametric building blocks. To enable the research of this thesis, some of these primitives required an update. In Section 2.1 the wing primitive is discussed, followed by the nacelle primitive in Section 2.2, and the pylon primitive in Section 2.3.

2.1. Wing primitive

Currently, the wing primitive used in the MMG is DARwing. During this thesis an update of the MMG to the newest version of ParaPy was attempted. However, this update broke the DARwing primitive. An investigation towards the source of the issues was unsuccessful. This led to the choice to introduce a new wing primitive to the MMG: ParaWing. At the start of this thesis, the work on ParaWing had already started, but stopped before the primitive was finished. The contributions made to ParaWing during this thesis include the addition of wing twist and the integration of ParaWing in the MMG.

First an introduction to DARwing is given, which also shows drawbacks encountered during its use. These drawbacks are formed into requirements for the new primitive. After this, ParaWing is introduced, and finally a comparison between the resulting geometries of both primitives is presented.

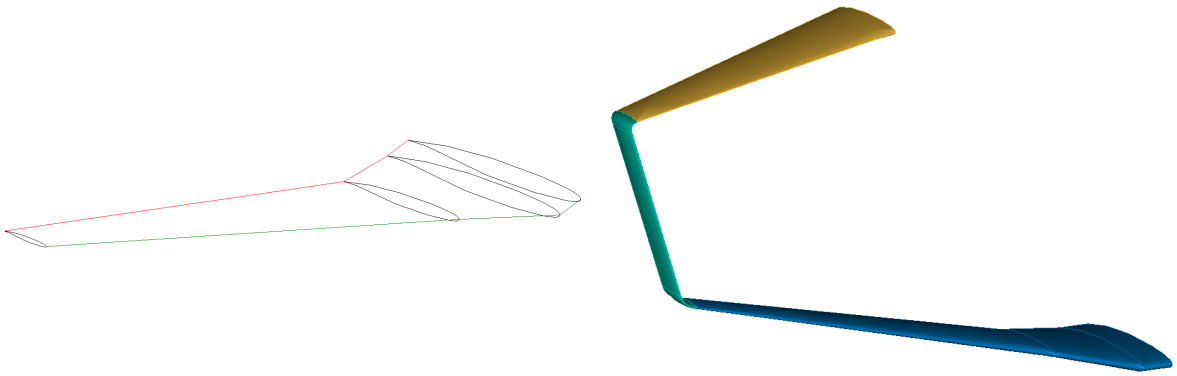
2.1.1. DARwing

DARwing is originally created for a previous version of the MMG, written in GDL, more information on this version can be found in the work of Koning [7]. With the transition from the GDL version to the ParaPy version of the MMG, done by Wei [8], some of the functionality has been lost. For example, while Koning showed continuous rails and different lofting methods, allowing for the creation of smooth lofts, DARwing currently only offers piecewise linear rails and the 'straight-lofts' method.

The DARwing primitive uses five rails to define the planform of the wing. These rails are the leading edge, trailing edge, twist axis, dihedral rail, and twist rail. The input data to create the rails is a set of points that define the rails directly, or act as control points for Bézier curves. The airfoils of the wing are provided by the user through files containing coordinates of the airfoil curve. These curves are placed on the leading and trailing edge rails to form a wire frame of the wing, shown in Figure 2.1a. This wire frame is used as input for a lofting operation to form a solid.

The inputs to create these rails are points that define a piecewise linear curve. A large drawback of this type of input is that often used wing properties, e.g. dihedral and sweep angles, are implicitly defined in these points. This makes it difficult for a user to change the wing geometry.

A second drawback of the DARwing is the procedure used to create the rails. This does not support dihedral angles of 90 degrees or larger. However, this is needed to create the box-wing of the PrandtlPlane or a winglet. To allow the box-wing and winglets to be modelled, Groot [9] added a 'connecting element' to DARwing. Using one or two existing wings, a winglet or connector between wings could be created. This requires to model the box-wing of the PrandtlPlane in three sections: the



(a) Wire frame of leading edge, trailing edge, and sections of PrandtlPlane front wing.

(b) Three components that make up a box-wing system, modelled in DARwing.

Figure 2.1: Geometry created using DARwing primitive.

front wing, the rear wing, and the connecting element. The triplet of wing components is shown in Figure 2.1b. At the connection of the connector with the wings, tangency constraints are applied to the leading and trailing edge, fixing the curvature of the corner elements. The implementation limits the design space of the connector corner and the separation in three parts increases the part count in the MMG. This increased part count increases the processing required to fuse all geometry into one model.

Based on these drawbacks, the following requirements for a new wing primitive are defined:

- The primitive should use a parametrisation that features common geometrical characteristics, e.g. dihedral angle, sweep angle, and chord length.
- The primitive should use a geometry creation procedure that supports continuous rails.
- The primitive should use a geometry creation procedure that does not limit the dihedral angle.
- The primitive should use a geometry creation procedure that allows a box-wing to be modelled as one wing component.

2.1.2. ParaWing

ParaWing, the newly introduced wing primitive, is based on the parametrisation described by Sobester and Forrester [10]. The inputs used in their parametrisation can be split into two sets: one defining the planform, the other defining the cross sections.

The first input set considered defines the planform. ParaWing's parametrisation uses chord lengths, wing span, and dihedral, sweep, and twist angles. The inputs and their type are gathered in Table 2.1. The usage of these inputs and the working principle of ParaWing are shown through an example.

Consider a swept, tapered, and twisted wing with an upward curved section at the tip, acting as winglet. The upward curve is made by increasing the dihedral angle. For a simple flat wing this angle should be kept low, but for this example wing, the dihedral is increased to 90° at the tip. The inputs for this wing are added to Table 2.1, as well. An input file containing a box-wing and vertical tail is added in Appendix A.

The first step is to create a reference rail, that acts as a spine of the planform, with unit length. This reference rail follows from the user specified dihedral, sweep, and *number of point used for rail generation*. During this step, first a list of coordinates η is created. This coordinate is the normalized

Table 2.1: Inputs for planform definition in ParaWing with example values of a swept, tapered, and twisted wing with an upward curved section.

Input	Type	Example value
Span	Number	10
Chord lengths	List of numbers	[4, 0.5]
Chord length definition fractions	List of numbers	[0, 1]
Chord distribution type	"linear" or "quadratic"	"linear"
Dihedral angles	List of numbers	[0, 6, 90]
Dihedral length definition fraction	List of numbers	[0, 0.8, 1]
Dihedral distribution type	"linear" or "quadratic"	"quadratic"
Sweep angles	List of numbers	[25, 25]
Sweep angle definition fractions	List of numbers	[0, 1]
Sweep distribution type	"linear" or "quadratic"	"linear"
Twist angles	List of numbers	[0, 5]
Twist angles definition fractions	List of numbers	[0, 1]
Twist distribution type	"linear" or "quadratic"	"linear"
Chord fraction of twist axis	Number or list of numbers	0
Number of points used for rail generation	Number or list of lists of numbers	200
Chord fraction of reference rail	Number	0.25

length along the reference rail. In Figure 2.2 [10], this is illustrated using the leading edge as reference rail and using ϵ as name.

If the *number of point used for rail generation* is an integer, these coordinates are equispaced. However, this input can also be a list of lists, each list containing a start η, end η, and a number of points to be placed in that wing section. This allows to provide more resolution in curved parts of the wing.

Using this coordinate list and the dihedral and sweep angles given at locations η, angle distributions for all values in the η-list can be found. Currently, piecewise linear and quadratic distributions are supported. Both find a distribution using two adjacent inputs from the user. To solve the third equation required for a quadratic distribution, the derivative is used. It is assumed that the derivative is zero at the root of the wing. Along the wing, this derivative is continuous to form a smooth distribution. To model a step change in the angle distribution, a user has to specify two different angles at two values of η that are close together.

With the angles following from these distributions and a distance following from the used point distribution, points are placed in space. By drawing a polygon through the points, a rail is formed. After this rail is created, it is uniformly scaled to place the outmost point at the given span. This results in the scaled reference rail, shown in Figure 2.3a.

In the second step, chord lines are added, with lengths following from the chord length inputs. Again, the user inputs are interpolated to find a distribution for all used values of η. The chord lines are placed around the reference rail, with the rail at a length fraction defined by the input *chord fraction of reference rail*. So that when this fraction is zero, the reference rail is the leading edge. After these lines are created, they are rotated around a line placed at *chord fraction of twist axis* along the chord lines. The angles used for the rotation follow from a distribution, found using the user specified twist angles. The chord lines are added to Figure 2.3b.

Thirdly, the start and end points of the chord lines are connected with two polygons, to form the leading and trailing edge rail, depicted in Figure 2.3c. Note that the use of polygons does result in a piecewise linear rail, similar to the leading and trailing edge rails resulting from DARwing. However,

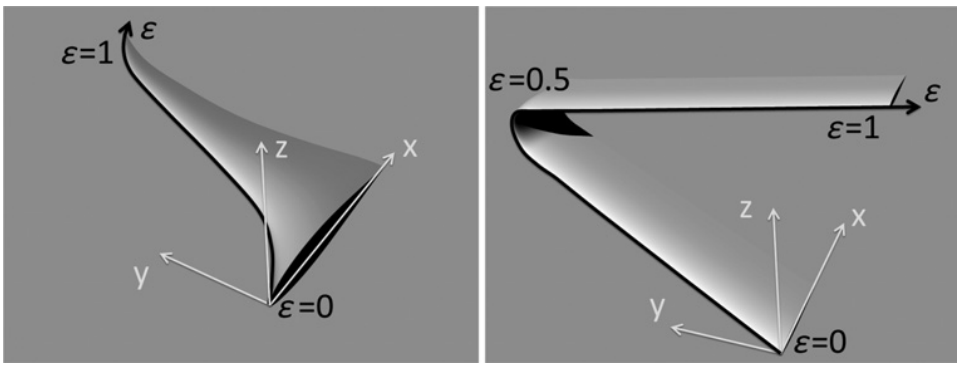


Figure 2.2: Two wings illustrating a Cartesian system with its origin at the root of the leading edge and an additional curvilinear dimension ϵ attached to the leading edge [10].

the use of curves that do result in a continuous rail, do not result in the desired rail. Since the given points are used as control points, the resulting curve does not pass through all points. With the use of a small spacing between the points that form the rails, i.e. a high number of points used for rail generation, a continuous rail can be approximated.

The fourth step connects the points on the chordlines at length *chord fraction of reference rail* to create a new, twisted reference rail. This is required in case the reference rail does not coincide with the twist axis. The new reference rail is added in Figure 2.3d. Finally, the three created rails are translated such that the leading edge point of the root is at the user specified position of the wing.

The second set of inputs consists of sectional data, describing the wing's cross sections. Again, the wing fraction η is used to place the sections. Per section, an airfoil should be specified. This can either be done by giving a data file with x- and y-coordinates of points that describe the airfoil, by giving CST-coefficients for both the upper and lower side of the section, or by giving a NACA 4-digit designation. These inputs are shown in Table 2.2. Depending on the type of input, a primitive that converts the inputs into an airfoil curve is called. This curve is then placed at the wing section, within the previously created rails, defined by the corresponding η. This results in a wireframe similar to that shown in Figure 2.1a.

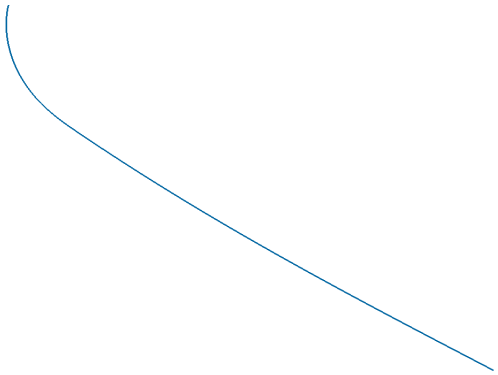
For each section the user defines, the airfoil can be flipped using the *flip airfoil* input. When this flag is set to true, the given airfoil is flipped 180° around its chord line. This can be used when creating box-wing aircraft. When the box-wing is modelled as a single wing, along its reference rail the top and bottom side are switched. Instead of turning the airfoils in the input files, this can now be done when the wing is created using this setting.

On a global wing level, the user can set a flag to close all airfoils. This is available to ensure a sharp trailing edge is created. To close the trailing edge of the airfoil, all points that define the airfoil curve are offset. This offset is done with respect to the camber line using the same offset on both the upper and lower side. This results in a slightly thinner airfoil, but the camber characteristics do not change.

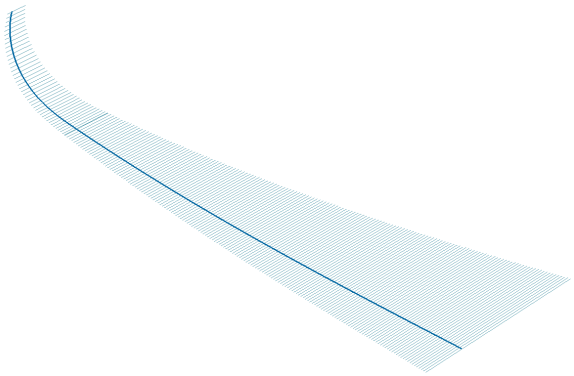
Furthermore, the user can ensure the rails of the wing end on the symmetry plane, by activating the global *is closed* flag. This is necessary for box-wing aircraft. The rails are trimmed with, or extended to the symmetry plane, if this flag is set to true. Also, to allow for isolated wings that start at the symmetry plane, the *flatten root* input can be set to true. If this is the case, the section of the wing root is not rotated with the dihedral angle.

The combination of sections and rails are used to create a loft to form the wing. This is done by firstly splitting the rails intro trunks, at the locations of the sections where an airfoil is defined. The trunks are lofted separately, and sewn together to form a single loft. This loft has open ends and is shown for the simple wing, together with its sections, in Figure 2.4a. The open ends are closed to be able to form a wing solid. From this solid, also the outer mold line can easily be obtained. Using this type of rail inputs and the process of rail creation, the box-wing of the PrandtlPlane can be modelled as one solid, depicted in Figure 2.4b.

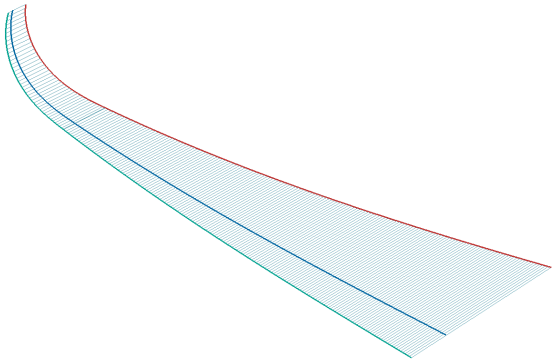
Instead of providing a single airfoil per section as shown before, the user can also provide multiple airfoils per wing cross section to form a multi-element wing, e.g. a wing with slats and flaps. For each element, their angle with respect to the total chord and chord ratio should be given. An example is



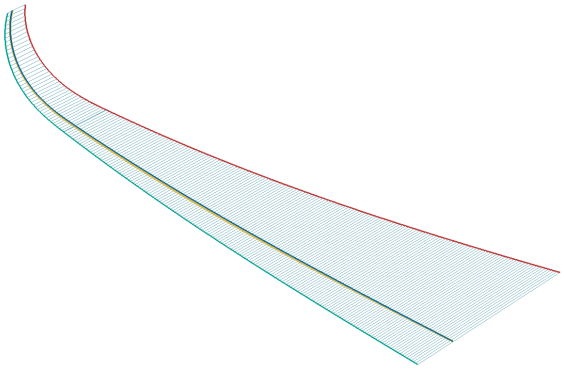
(a) Scaled reference rail.



(b) Scaled reference rail with chord lines.



(c) Scaled reference rail with chord lines and leading and trailing edge rails.

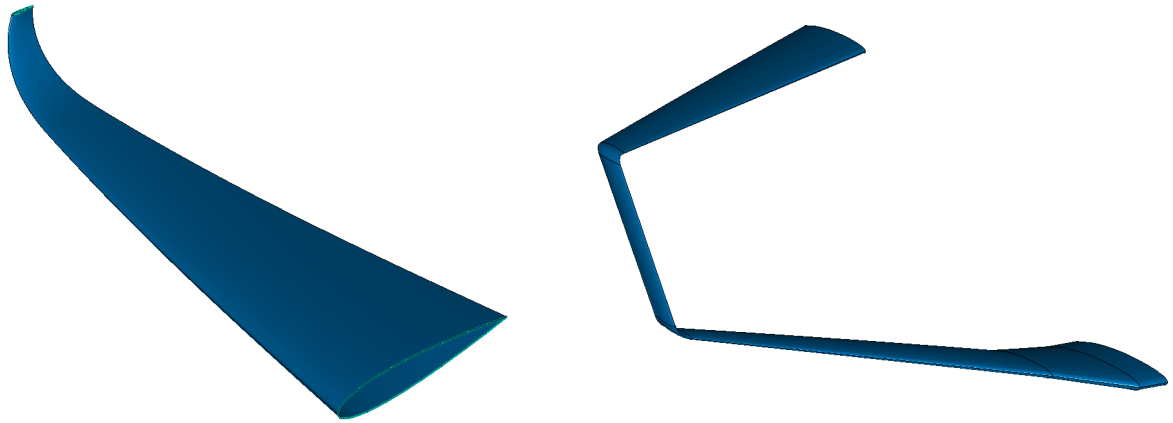


(d) Scaled reference rail with chord lines and leading and trailing edge rails. Added twisted reference rail.

Figure 2.3: Planform generation process in ParaWing.

Table 2.2: Inputs available to create wing sections in ParaWing.

Input	Type	Example value
η	Number	0
CST bottom	List of numbers	[0.23, 0.1, 0.05, 0.25]
CST top	List of numbers	[-0.23, -0.25, 0.05, 0.1]
Designation	String	”naca0012”
Filename	Pathlike string	”./airfoil.dat”
Flip airfoil	Boolean	False



(a) Simple wing loft with sections indicated in green.

(b) Single component that makes up box-wing system, modelled in ParaWing.

Figure 2.4: Geometry created using ParaWing primitive.

Figure 2.5: Multi-element wing generated using ParaWing.

shown in Figure 2.5 and Table 2.3 shows the inputs required to setup this type of cross-section. The use of this multi-element wings is not integrated with the MMG at this moment.

To be able to use the DARwing style of inputs with ParaWing, and thereby ease the step towards using this new primitive, a conversion tool is made. This tool converts the points and sections given to DARwing, to angles and chord lengths at wing fractions and sections in the format required by ParaWing. To do this, a leading edge is formed from the points given in the file. This is used to determine the sweep and dihedral angles of the leading edge. Together with chord lengths, obtained from the distance between leading and trailing edge points, and the given twist angles in the input file, this returns the required planform inputs. The tool works for single wings, wing system as the box-wing should be converted manually.

2.1.3. Primitive comparison

Comparing the functionality of ParaWing with the requirements from Section 2.1.1, it can be concluded that all these requirements are fulfilled. However, ParaWing is not ready to replace DARwing completely. For example, in DARwing, moveables such as ailerons can be modelled, but these are not added to ParaWing yet. Also, some wing customisation options, such as the orientation of the sections within the rails, are missing. Since these were not required during this thesis, no effort is put into it.

Next to the fulfilment of the requirements, also the resulting geometries of DARwing and ParaWing should be compared. For this purpose, the NASA Common Research Model (CRM) wing and the PARSIFAL Milestone 1 (MS1) box-wing are compared using both primitives.

Table 2.3: Inputs to create a multi-element wing in ParaWing.

Input	Type	Example value
η	Number	0.0
Airfoils	List of strings	["./ap3-slat.dat", "./ap3-main.dat", "./ap3-flap.dat"]
Element locations	List of lists of numbers	[(0.01028, 0.02), (0.08595, 0.53), (0.76, 0.0)]
Element chord fractions	List of numbers	[0.16935, 0.68928, 0.27939]
Element angles	List of numbers	[-55.38, 0.0, 14.80]

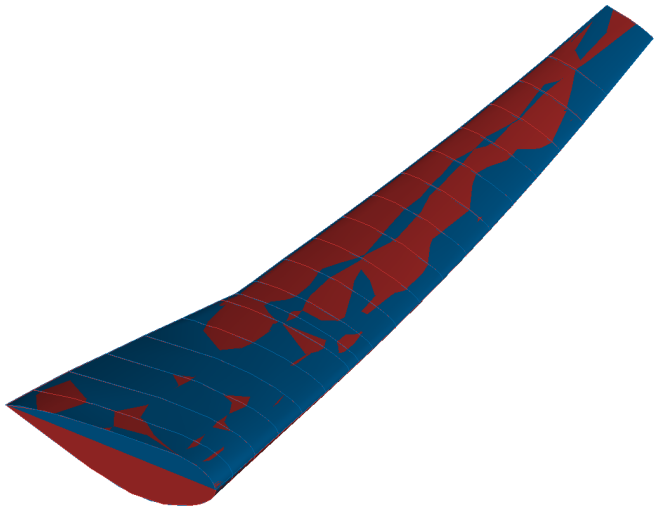


Figure 2.6: Overlay of CRM wings created using ParaWing (blue) and DARwing (red).

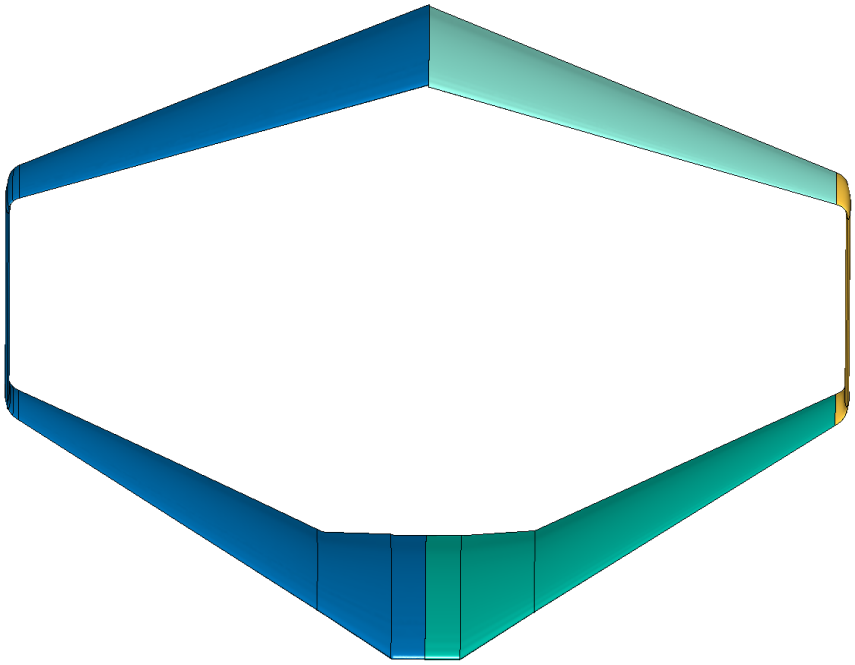
In Figure 2.6, two instances of the wing of the CRM model, created using ParaWing and DARwing, are compared. The inputs for ParaWing are obtained using the input conversion tool that is introduced in Section 2.1.2. The overlay shows that the two wings do not overlap perfectly. This is expected, since the inputs used for lofting differ between the primitives. An investigation of the location of points along the leading and trailing edges shows they differ less then 1%, calculated using Equation (2.1) for the x-, y-, and z-coordinates.

A comparison with a STEP model from the CRM[1] shows similar results for both the surface and leading and trailing edge points. Also the DARwing primitive shows a small difference with the STEP model.

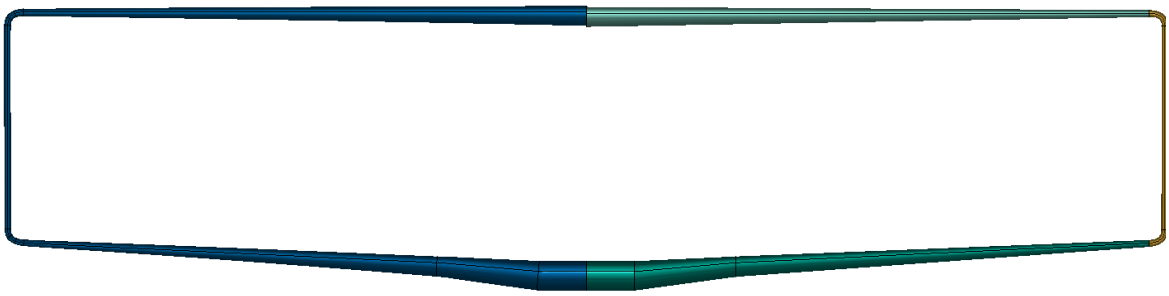
$$\delta = \frac{coordinate_{ParaWing} - coordinate_{DARwing}}{|coordinate_{DARwing}|} \cdot 100\% \tag{2.1}$$

The comparison of the PARSIFAL box-wing is shown in Figure 2.7, the resulting shapes of ParaWing and DARwing are placed next to each other. Two differences are easily noticed: firstly, in ParaWing the box-wing can be modelled as one shape and secondly, there is an offset in horizontal and vertical stagger. Because the wing system is modelled as one wing, direct control over the stagger is lost. With ParaWing's parametrisation, the stagger is a result of the dihedral and sweep distribution.

[1]Obtained from https://commonresearchmodel.larc.nasa.gov/geometry/stp-files/ on October 7, 2020



(a) Front top view



(b) Front view

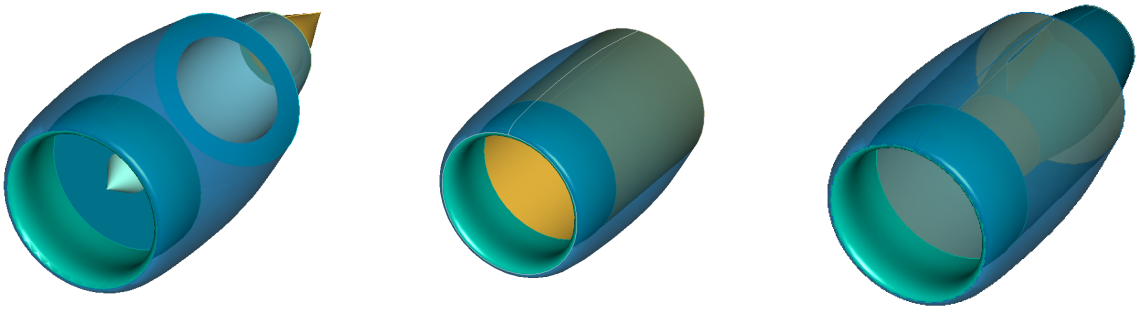
Figure 2.7: Comparison of PARSIFAL box-wing created using ParaWing (left, blue) and DARwing (right, green and yellow).

To obtain the dihedral and sweep distribution of this wing system, first the leading and trailing edges are extracted from the DARwing shapes. The edges of the three parts, i.e. front wing, connector, and rear wing, are fused to form one curve. This curve is sampled with points to form a linearisation. This sampling takes the curvature of the curve into account, so more points are used at highly curved parts. From this linearisation, the chord, dihedral, sweep, and twist distribution can be found to use as input for ParaWing. The result of this linearisation results in a 0.48% error in the horizontal stagger and a 0.41% error in the vertical stagger of the leading edge.

With its parametrisation, ParaWing offers more design freedom of the corner elements of the box-wing. Instead of just a tangency constraint as in the solution of Groot [9], both the dihedral and sweep angle can be modelled using n^{th}-order distributions. A drawback is the loss of direct control over the stagger if the box-wing is modelled as one shape. This control can be regained by using the three parts approach by Groot [9] in combination with ParaWing.

2.2. Nacelle primitive

The GTPy tool creates a geometry of a complete engine. This is a closed solid consisting of cowls, an inlet, exhaust faces, a fan face, and a spinner face. For information on the creation of this solid, the reader is referred to [5]. This complete engine solid is shown in Figure 2.8a. To model a powered



(a) Full engine geometry.

(b) Flow through nacelle geometry.

(c) Flow through nacelle geometry with core body.

Figure 2.8: Available engine geometries in nacelle primitive. Each component indicated by their own colour.

engine in an aerodynamic solver, the normal velocities on the fan face and exhaust faces are specified.

However, not all analyses require this full engine geometry, e.g. wind tunnel tests are often performed with through flow nacelles (TFN). These use the external geometry of the nacelle, but just a tube on the inside. Although a TFN can be approximated in an aerodynamic simulation by carefully setting the inlet and exhaust velocities on a full engine geometry, proper TFN analysis is impossible with this type of model. Because of the core cowl and plug, the exhaust area is often smaller than the fan area. To keep a constant mass flow through the engine, inlet and exhaust velocities should be set according to this area ratio. However, this requires the exhaust velocity to be larger than the inlet velocity, which will generate a thrust force in the simulation. Setting the inlet and exhaust velocities to the same value, as follows from the simulations of Malouin [11], will remove air from the simulation, which is in reality impossible.

To overcome this problem, a TFN is introduced to the MMG. This TFN model uses the cowls and inlet faces of GTPy, and connects the aft edges of the inlet and cowl with a face, to form a solid. By using the geometry already created using GTPy, the engine cowl and inlet still form a realistic nacelle geometry. A TFN is shown in Figure 2.8b.

This TFN limits the pylon shaping. Since there is no core body, pylons can only connect to the fan cowl. To extend the pylon design space, a TFN with core body is introduced, see Figure 2.8c. This uses, next to the outer nacelle and inlet, the geometry of the engine's core cowl. This forms a second channel, that models the inner flow tube of the engine. The part external of the fan cowl can be used to attach the pylon to.

2.3. Pylon primitive

To connect the engines to the airframe, a pylon is used. This can be seen as an aerodynamic, wing-like fairing that surrounds a structure that carries the loads of the engine into the airframe. For this research, the fairing is of interest. Unfortunately, no design rules are found for the design of the fairing. The design is highly custom, influenced by the flow around the engine and airframe. The fairing is shaped in a way that minimises the drag forces on it. In this research, no optimisation to reduce the drag is performed, a relatively simple pylon is considered.

The current pylon implementation takes attachment points on the engine, an airframe part to connect to, and leading and trailing edge sweep angles as inputs. Also, the angle between a vertical line and the pylon can be given, to set where on the engine the pylon should connect. This is the angle of attachment. With these inputs, a planform with straight leading and trailing edges is formed. The full procedure is described by Proesmans [5].

To increase the design space of the pylon in the MMG, a new primitive is introduced. In this new pylon primitive, the user starts with supplying connection points on both the engine and airframe. The engine's attachment points are obtained from a Turbofan instance, the engine primitive by Proesmans. The airframe side points can be given directly, but the user can also give location parameters for either

Table 2.4: Inputs for pylon creation.

Input	Type
Parent object uid	ParaWing or Fuselage instance
Airframe attachment points	List of points
Root airfoil	String
Tip airfoil	String
Leading edge longitudinal offset	List of numbers
Trailing edge longitudinal offset	List of numbers
Additional sections	List of dictionaries
Close airfoils	True or False
Fuselage front section	Number
Fuselage aft section	Number
Fuselage vertical offset	Number
Wing section location	Number
Wing front spar fraction	Number
Wing aft spar fraction	Number
Wing vertical offset ratio	Number
Angle of attachment	Number
Split at intersection	True or False

a wing or fuselage connection. With these parameters, the attachment points are created by the primitive. Similar as in the procedure of Proesmans, an angle of attachment can be given. If this is not specified by the user, the angle resulting in the shortest pylon will be used.

The attachment points are initially connected using straight lines, to form leading and trailing edges. These edges can be moved along the length of the aircraft using the leading and trailing edge longitudinal offset inputs. This allows to use the given attachment points as the connection points of the pylon's internal structure to the airframe and engine. With the offsets, the fairing can be shaped around the connections. To form the fairing, root and tip airfoils have to be given, as well. These can be either a NACA-4 designation or a path to a coordinate file. These inputs are tabulated in Table 2.4.

To further control the shape of the pylon, addition sections can be given. These sections are given using a span fraction, an airfoil for the section, and the leading and trailing edge offset it should apply to the original straight leading and trailing edges. The created leading and trailing edge are converted to ParaWing inputs using the conversion tool from Section 2.1.2. With these inputs, ParaWing creates a pylon fairing shape. Table 2.5 shows the inputs for the additional sections.

In case of an TFN, the user can chose to split the pylon on the intersection with the engine, so no pylon is in the engine's internal stream tube. This is depicted in Figure 2.9. Further downstream in the MMG, the red part of the pylon is not used, e.g. when fusing all parts to create a full aircraft solid or for drag determination using textbook methods.

As first example, a fuselage mounted pylon is modelled, shown with reference in Figure 2.10. The values of the inputs to create this pylon can be found in Appendix B. This type of pylon requires no additional sections and is therefore easy to create. Note that for this example, no airframe attachment points are given, but parameters to make the primitive find these.

[2]Adapted from https://customer.janes.com/JAWADevelopmentProduction/Display/JAWA0091-JAWA, obtained on September 16, 2020.

Table 2.5: Inputs for pylon additional sections with their type.

Input	Type
Span fraction	Float
Airfoil	String
Position	"absolute" or "relative"
Leading edge position	List of floats
Trailing edge position	List of floats

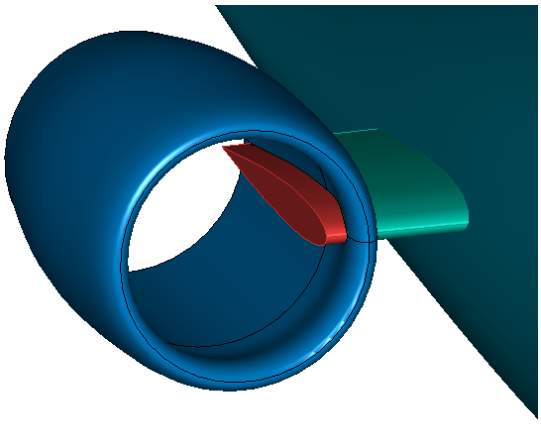
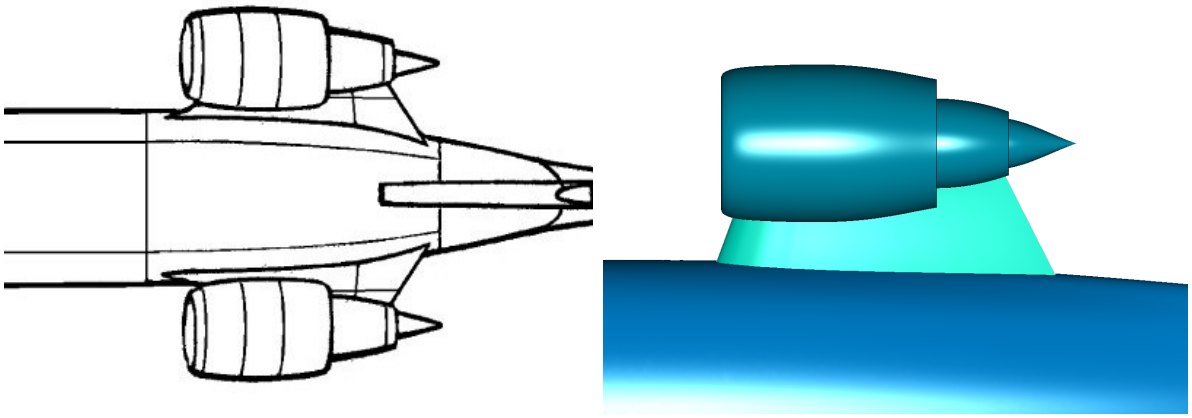


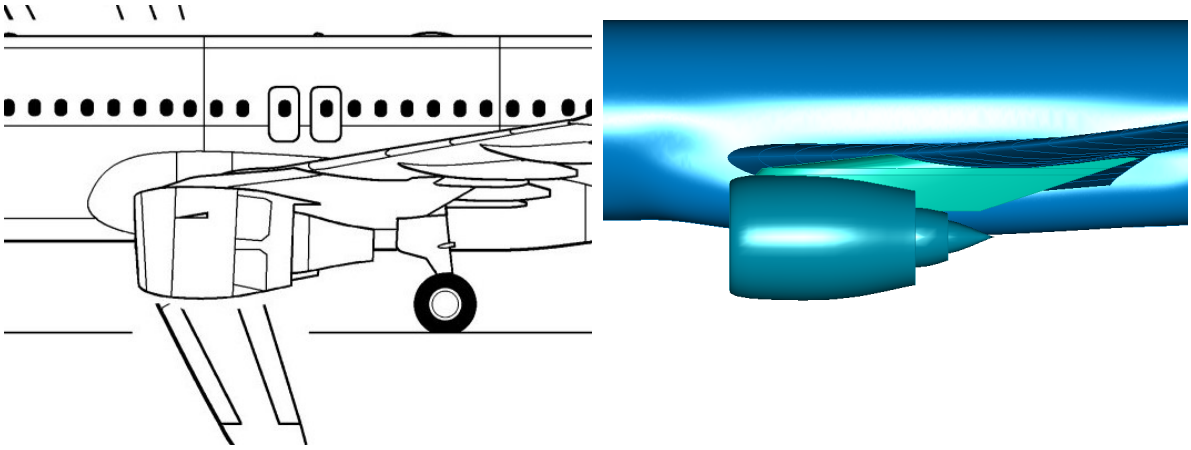
Figure 2.9: Pylon for a through flow nacelle. The red part inside the engine is not used downstream in the MMG.



(a) Reference pylon of the Bombardier CRJ700.[2]

(b) Geometry created using the pylon primitive.

Figure 2.10: Fuselage mounted engine pylon.



(a) Reference pylon of the Airbus A320.[3]

(b) Geometry created using the pylon primitive.

Figure 2.11: Wing mounted engine pylon.

Figure 2.11 shows a wing mounted pylon. To create this planform, additional sections are added to the inputs. One to fix the leading edge on the front of the nacelle, the second to form the steep leading edge part that is at the nacelle connection. The inputs for this pylon are given in Appendix B, as well. Although the planform can be modelled to match a realistic pylon, the final loft is not matching. The placement of cross-sections returns a flattened leading edge on the top part of the pylon. At the point where the leading edge sweep changes, a sharp angle is formed in the loft, while in reality a smooth curvature is used. To create this shape in the pylon geometry, another lofting method should be used. It might also be necessary to define the cross sections normal to the flight direction, instead of tangent as is done now.

[3]Adapted from https://customer.janes.com/JAWADevelopmentProduction/Display/JAWA0416-JAWA, obtained on September 17, 2020.

3

Aerodynamic analysis

Before a trade-off study for the engine integration analysis best practice can be started, methods and practices to include in this trade-off have to be selected. The analyses under consideration take place during the preliminary design phase, which is characterised by many design options to choose from. To rapidly gain an insight in the performance of the design options, low cost methods have to be employed. Also, the required results from an analysis have to be established. To assess an engine installation completely, information on the installation's effect on lift, drag, and moment are required. During this research, focus is on symmetric forces and moments, thus lift, drag, and pitching moment, since these are predominant in the largest part of a flight.

The first type of method considered are textbook methods. These methods use geometric data and analytical relations to predict the impact of aircraft components on the aircraft's performance. While textbook methods prove to be valuable to find the drag impact of engine integration, they don't provide means to estimate the contribution in lift and moment changes due to the engines to the overall aircraft. As such, textbook methods are not satisfactory on their own.

Therefore, simulations that do provide this information are required. For this, two 3d panel solvers are considered, despite the discouraging results obtained with VSAERO by Proesmans [5]. FlightStream is considered because previous research [12, 13] showed promising results when using FlightStream in engine integration optimisations. Also, VSAERO is not ruled out as tool, since other practices than the one used by Proesmans might give satisfactory results.

First, the used textbook methods are introduced and compared with each other in Section 3.1. This is followed by a description of FlightStream and VSAERO in Section 3.2. This includes the views the solvers use of the geometry created in the MMG. Also, a validation of FlightStream is added in this section, since it is not used with the MMG before. Finally, a trade-off to find an engine integration analysis best practice is performed in Section 3.3.

3.1. Textbook methods

In preliminary design, often analytical methods are used to estimate aircraft performance. Based on geometrical data combined with physics and statistical relations, aerodynamic performance coefficients can be calculated. These methods are easy to use, but also limited in their applicability, e.g. the flow conditions and aircraft configurations they can be used for. Since statistical data is used, the accuracy of these methods for unconventional aircraft might be limited. Furthermore, not all aircraft performance is captured by these methods, leaving gaps in the performance estimation. Despite these drawbacks, they can provide an early indication of aerodynamic performance.

In this section, two methods are introduced, one by Torenbeek and one by Raymer. Both methods divide the aircraft in components as wings, fuselages, pylons, and nacelles and provide equations to calculate the performance of all these components. The following sections focus on nacelle and pylon performance. After the equations are introduced, they are used to find their accuracy in comparison with high order CFD data.

3.1.1. Torenbeek

An often used prediction method is the textbook method by Torenbeek [14]. Based on design parameters, lift, drag, and moment coefficients can be calculated per component of an aircraft. However, not for all components a prediction can be made. For example, the effects of pylons and nacelles on lift are hard to predict and therefore, ignored in this method.

However, the shift in aerodynamic centre x_{ac} due to the nacelles is given in Equation (3.1). This uses the lift curve slope per angle of attack in radians of the wing and fuselage $\left(C_{L_\alpha}\right)_{wf}$, as well as the nacelle length and diameter l_n and d_n. The shift is also related to the reference area S_{ref} and mean aerodynamic chord $\bar{c}$. The factor k_n is -4.0 for under wing installations and -2.5 for installations on the rear of the fuselage.

$$\Delta_n \frac{x_{ac}}{\bar{c}} = \sum k_n \frac{d_n^2 l_n}{S_{ref}\bar{c}\left(C_{L_\alpha}\right)_{wf}} \tag{3.1}$$

An extensive method is written to calculate the drag of an aircraft, based on the drag created by individual components. For engine nacelles, a calculation of the profile drag is provided together with calculations for additional drag sources. Furthermore, some indications for the interference drag are given. The used equations are applicable for subcritical speeds, that is when no supersonic flow is encountered on the parts that are investigated.

The Torenbeek method calculates the drag-area C_dS of a component. To find the drag coefficient from this drag area, it should be divided by the reference area of the aircraft. In this method, the breakdown of the aircraft into components is extended to a breakdown of the nacelle as well. Separate drag contributions for the fan cowl, core (or gas generator) cowl, and core plug are calculated and summed.

The drag area of the fan cowl is calculated using Equation (3.2), a function of friction drag and afterbody pressure drag. The friction drag can be found using Equations (3.3) to (3.5) and is based on the flow capture area's diameter d_o, highlight diameter d_h, and nacelle diameter d_n. The engine highlight and nacelle diameter follow from the geometry of the engine. The capture area is determined using the engines inlet mass flow and free stream conditions, assuming no spillage occurs during the investigated flight phase. The fraction β is the fraction of the nacelle length where the largest diameter is found, the forebody length fraction. The factor φ is a correction for excess flow velocities due to the curvature of the cowl. The coefficient $\left(C_{D_p}\right)_\beta$ is the cowl's boat tail pressure drag coefficient. It is found using the correlations in [14].

$$(C_DS)_f = C_f\left(\beta\left(1+\varphi_n\right)^{5/3} + (1-\beta)\right)S_{wet} + \left(C_{D_p}\right)_\beta \frac{\pi}{4}d_n^2 \tag{3.2}$$

$$C_f = \frac{0.455}{\log_{10} Re^{2.58}} \tag{3.3}$$

$$Re = \frac{\rho V l}{\mu} \tag{3.4}$$

$$\varphi_n = 0.33\frac{d_n - d_h}{\beta l_n}\left(1 + 1.75\frac{d_h^2 - d_o^2}{d_n^2 - d_h^2}\right) \tag{3.5}$$

Both the engine core cowl drag and core plug drag are often seen as a loss of thrust and their values are provided by the engine manufacturer. Since this is not the case for the engines under consideration in this thesis, Equations (3.6) and (3.7) are used to find to core cowl and plug drag, respectively. Again, areas are directly obtained from the modelled geometry. All engine flow conditions, fan flow Mach number M_f and core flow Mach number M_g follow from the GTPy analysis.

$$(C_DS)_g = C_f\left(\frac{M_f}{M_\infty}\right)^{11/6}\left(\frac{1+0.116M_\infty^2}{1+0.116M_f^2}\right)^{2/3} S_{wet_g} + \left(C_{D_p}\right)_\beta \frac{\pi}{4}d_g^2 \tag{3.6}$$

$$(C_D S)_p = C_f \left(\frac{M_g}{M_\infty}\right)^{11/6} \left(\frac{1 + 0.116 M_\infty^2}{1 + 0.116 M_g^2}\right)^{2/3} S_{wet_p} \tag{3.7}$$

The profile drag of the pylon can be calculated as well, for this, Equation (3.8) is provided. This uses the pylon's relative thickness in streamwise direction t/c and quarter chord sweep angle Λ. To find the profile drag of the total engine installation, Equation (3.9) is used. The contribution of separate components can be found by using this formula with only that component.

$$(C_D S)_{py} = C_f \left(1 + 2.75\,(t/c)_{py} \cos^2 \Lambda_{py}\right) S_{wet_{py}} \tag{3.8}$$

$$C_{D_0} = \frac{(C_D S)_f + (C_D S)_g + (C_D S)_p + (C_D S)_{py}}{S_{ref}} \tag{3.9}$$

To account for interference with the airframe, for wing mounted low bypass engines 20% of the nacelle and pylon drag should be added. High bypass engines mostly remove five to ten drag counts, due to favourable interference. When mounted on the fuselage, low bypass ratio engine installations can give a drag penalty of 50% of the nacelle and pylon drag. No number is available for high bypass ratio engines.

3.1.2. Raymer

The drag estimation method by Raymer [15] is an equivalent skin friction method. The skin friction drag coefficient C_{D_0} of an aircraft component is calculated using a flat plate skin friction coefficient C_{f_c}, a component form factor FF_c, an interference factor Q_c, the wetted area of a component S_{wet_c}, and a reference area S_{ref}. These properties find a drag coefficient through Equation (3.10). To find the skin friction drag of an entire aircraft, the contributions of all components should be summed.

$$C_{D_{0_c}} = \frac{C_{f_c} FF_c Q_c S_{wet_c}}{S_{ref}} \tag{3.10}$$

The formulation of the flat plate skin friction coefficient depends on the type of flow, laminar or turbulent. Since Raymer states laminar flow is, on most aircraft, only found on wings and tails, the turbulent version is used, see Equation (3.11). This coefficient is a function of Mach number M and the Reynolds number Re. The used Reynolds number is the lower one of the actual component Reynolds number, calculated using the length of the engine l and flow conditions, and a cut-off Reynolds number, see Equations (3.4) and (3.12). The value of k is the skin-roughness height. During this thesis, the value from Raymer for production sheet metal is used, so $k = 4 \times 10^{-6}$ m. The formulation of Re_{cutoff} used here is suitable for subsonic Mach numbers. Another version is available for supersonic Mach numbers.

$$C_f = \frac{0.455}{\left(\log_{10} Re\right)^{2.58} \cdot \left(1 + 0.144 M^2\right)^{0.65}} \tag{3.11}$$

$$Re_{cutoff} = 38.21 \left(\frac{l}{k}\right)^{1.053} \tag{3.12}$$

The form factor for an engine nacelle can be calculated using Equation (3.13) and depends on the fineness factor, defined in Equation (3.14). The fineness factor uses the engine's length and diameter or cross sectional area.

$$FF = 1 + \left(\frac{0.35}{f}\right) \tag{3.13}$$

$$f = \frac{l_n}{d_n} = \frac{l_n}{\sqrt{(4/\pi)\, A_n}} \tag{3.14}$$

The mutual interference between the airframe and nacelle increases the generated drag. This effect is captured in the interference factor Q_c. Its value depends on the distance between the engine and airframe, which are:

- 1.5 for engine directly mounted to airframe;
- 1.3 for distances less than an engine diameter;
- towards 1.0 for distances beyond an engine diameter.

The final properties required in Equation (3.10) are the wetted area of the engine and a reference area. The engine's wetted area is obtained from the geometry in the MMG. The reference area is an input that should be provided by the user.

Using the same rationale, the drag of a pylon can be calculated. The form factor of a pylon is given in Equation (3.15). This equation can be used for all winglike components and it uses the relative thickness t/c and the location of the maximum thickness along the chord $(x/c)_m$. The required sweep angle Λ_m is that of the maximum thickness line.

$$FF = \left(1 + \frac{0.6}{(x/c)_m}(t/c) + 100(t/c)^4\right)\left(1.34M^{0.18}\left(\cos\Lambda_m\right)^{0.28}\right) \tag{3.15}$$

3.1.3. Comparison

To compare the results of the two textbook methods, a drag breakdown of the nacelle and pylon drag on an aircraft is required. Stańkowski [16] performed a study on the CRM using a RANS simulation to find the installation effects of an engine. This study showed generally good agreement with wind tunnel data and, therefore, the drag breakdown presented is used as reference in this comparison. The breakdown shows values of 5 counts for $C_{D_{py}}$ and 30 counts for C_{D_n}. These are the values at $\alpha = 0°$ in a configuration with wing, fuselage, tail, pylon, and nacelle. Thus that of a fully integrated aircraft. In this coefficient, the pressure and forces on the surface, as well as interference, are included. The modelled nacelles are TFNs, but since the design of this TFN is made to achieve the same flow as for commercial airliners [17], it is assumed this data can still be used as reference.

The drag contributions of the pylon and nacelle are calculated with the geometric data and flow conditions presented in Tables D.1 and D.2 using the equations presented in Sections 3.1.1 and 3.1.2. The geometrical data of the engine is taken from GTPy, in which an engine of the same size is created. The pylon data is obtained from a STEP model of the CRM.[1] Since no exact flow data is given, the components' Reynolds numbers are scaled with the mean aerodynamic Reynolds number $Re_{\bar{c}}$ and the component's length.

The results of the calculations are tabulated in Table 3.1, together with the reference data introduced above. Due to the formulation of the Torenbeek estimation, the interference drag of this method is provided separately. This interference is included in the values for the Raymer method and the reference data. From this results, it can be seen that both methods underestimate the drag of the nacelle and pylon, with errors in the total drag estimation of respectively 65% and 35%. Both methods estimate the pylon drag with an error of approximately 70%. The nacelle drag is off by 47% in the Torenbeek method, and 30% in the Raymer method. Since in the Torenbeek method an interference correction is added, the error is increased further. The large errors might follow from a missing wave drag contribution, however, the reference data does not show shock waves are present on the nacelle and pylon.

From the results presented above, it can be concluded that the textbook methods do not come close to measured drag values. Also, since they only provide data on the added drag and not the effects of lift, they do not provide all information required to study an engine installation. Therefore, they cannot be used to calculate engine installation effects on their own. However, they provide an opportunity to model only parts of the engine installation and correct for the missing parts. This is can be applied by correcting for the presence of a pylon, since pylon drag is small with respect to the nacelle drag.

[1] Obtained from https://commonresearchmodel.larc.nasa.gov/geometry/stp-files/ on October 7, 2020.

Table 3.1: Results of textbook drag calculations and reference data [16] at $M = 0.83$, $Re = 5 \times 10^6$.

	C_{D_n} (counts)	$C_{D_{py}}$ (counts)	Interference contribution (counts)	Total (counts)
Torenbeek	16	1.4	-5	12.4
Raymer	21	1.7	0	22.7
Reference data	30	5	0	35

3.2. Panel methods

In this section, the two panel methods used in this research, FlightStream and VSAERO, are described. For Flightstream, this is done by a general introduction of the solver in Section 3.2.1, followed by the view in Section 3.2.2. This view is a translation of the geometry created in the MMG into a format the solver can work with. For panel solvers, this is a mesh and a set of boundary conditions. Since FlightStream has not been used with the MMG before, a validation is added in Section 3.2.3. VSAERO and its view are described in Sections 3.2.4 and 3.2.5, respectively. No additional validation is performed for this solver, since its use with the MMG is already established.

3.2.1. FlightStream introduction

The FlightStream 3d panel method uses a vorticity distribution over a body to find the aerodynamic behaviour of the body. Novel to this method is that a conversion is used to go from unstructured vorticity over the body into directional integrated circulation. This circulation can be used to apply the Prandtl lifting-line theory [18]. FlightStream can account for compressibility effects by applying the Prandtl-Glauert, Karman-Tsien, or Laitone model [19–21]. The model choice is made by the solver, a user can only activate or deactivate it. These models do not capture shock waves, which should be kept in mind when simulating high Mach conditions.

Once the vorticity distribution over the aircraft is calculated, the velocity and pressure distributions can be found as well. Within FlightStream, different models for lift, induced drag, moment, and parasite drag can be chosen. These use either the vorticity or pressure distribution. Lift and induced drag offer the choice between pressure and vorticity. The moment can be calculated using either vorticity, linear pressure, or non-linear pressure. The final choice is for the viscous model calculating the skin friction drag. For this, the Reynolds averaged and momentum integral models can be chosen. Using the flow conditions and a reference area, these forces and moments can be converted into coefficients. Furthermore, a flow separation model is available as well. This model can be enabled or disabled.

Results obtained using FlightStream in previous research seem promising, close matches to reference data is found. However, during this thesis, some issues are encountered as well. Since FlightStream is still in active development and the Flight Performance group has close contact with the developers, it is expected that these issues will be resolved in due time. The results in this thesis are obtained using the 2020.2 version of FlightStream, built on November 30, 2020. The two most prominent known issues of this build are: (1) forces after separation are not taken into account properly, resulting in too low values of maximum lift coefficient and underestimated drag; and (2) forces on faces using 'inlet' boundary conditions, which specify their normal velocity (used for engine inlets and exhausts), are all accounted as added drag.

The separation issue limits the usable range of angle of attack during simulation. This issue is known by the developers and fixed in a new release. In the remainder of this research, only a limited range of angles of attack is used, to circumvent this bug. Although this bug was fixed in a new version, released during this thesis, the new version introduced another bug related with bluff body drag. Therefore, it is chosen to continue with the version with the separation bug, since the bug's effects are predictable.

Previous research by King [12] and Ahuja [13] have shown that FlightStream can be used to simulate the flow around engines. Discussions with Ahuja, one of the developers, resulted in modelling both the engine's fan and exhaust faces using 'inlet' boundary conditions. This condition specifies a non-zero normal velocity on faces it is applied to. When applying the inlet and exhaust velocities, obtained from GTPy, to the respective faces, unexpected results are returned. Firstly, only one of the exhaust faces is showing to have forces and moments acting upon it, while these forces are expected on both exhaust

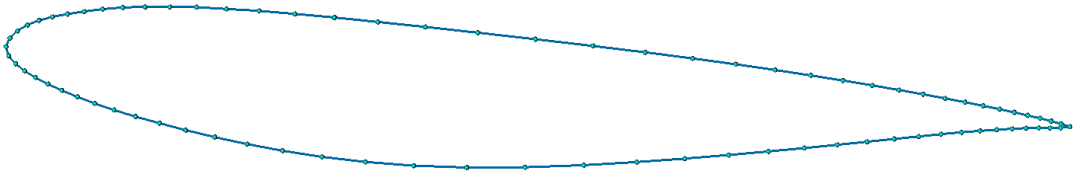


Figure 3.1: Point distribution around a typical structured mesh cross section.

faces. Secondly, the forces are pointing in the same direction and accounted for as a drag contribution, while actually a thrust force should be added. Since it is unsure where this is originating from, it is undesirable to compensate for this during post-processing. This led to the choice to use FlightStream only with TFN. This bug was not encountered by the developers before and is under investigation.

3.2.2. FlightStream view

FlightStream is able to operate with structured, unstructured, and hybrid meshes. The advice of the developers is to use a hybrid mesh, with a structured mesh on lifting surfaces and an unstructured mesh elsewhere. Non-lifting surfaces can be used with a structured mesh as well, as long as no sudden change in mesh face area is made. This advice leads to the following meshing strategy: create a structured mesh on unintersected wings and the front part of the fuselage, and apply an unstructured mesh around intersections in wings and the remaining part of the fuselage. Tip faces of wings and the nose and tail patch of the fuselage are meshed using an unstructured mesh as well. The mesh on engines and pylons is structured when the type of intersection allows it, otherwise an unstructured grid is generated on those parts.

In this section, the available controls are tabulated together with the expected type of the inputs. An example of an input file, containing these inputs is attached in Appendix C.

If a wing is not intersected, the mesh creation procedure is to apply a fully structured grid on it. This mesh is obtained by placing control points around the cross sections of the wing and at the leading and trailing edges. These control points can then be used by the mesher of the MMG to create a mesh. The points around a cross section are placed on both the bottom and top side of the cross section using a geometric distribution. Initial testing showed that 80 points around the cross section are required, using a growth rate for the distribution of 1.1. The point distribution around a cross section is shown in Figure 3.1. Using less points or a larger growth rate results in the leading edge suction peak not being resolved properly. This was also the result of using a cosine distribution for the points. Points on the leading and trailing edge are placed equispaced, using a given value for the pitch of the nodes.

When a wing is intersected, an unstructured patch is placed around the intersection, while on the remaining part of the wing still a structured mesh is applied. The unstructured patch ensures the intersection is not affecting the mesh of other aircraft components. The splitting of the wing to create a patch to apply the unstructured mesh to and the mesh itself can be seen in Figure 3.2. This specific example shows a large jump in face area on the transition from structured to unstructured patch. This jump can be decreased by making the unstructured patch larger. Unfortunately, this was not possible on the used geometry. This is, therefore, an area that should be examined before post-processing the results, to ensure no unnatural flow behaviour is found. The leading and trailing edge of the intersected part still use the same spanwise pitch as on the structured part of the wing. On the unstructured patch, the minimum and maximum linear dimension of the mesh faces, as well as the growth rate of the area between adjacent faces, can be set. These properties are also settable for the wing tip face.

If a wing is in the spanwise direction split into trunks, spanwise refinement controls of specific trunks can be given. This list contains entries using the index of the leading edge segment and the desired pitch at that section. The spanwise controls of the given trunk are updated to use the given pitch. The leading and trailing edge of the neighbouring trunks are adapted to smoothly change from the general pitch to the refined pitch. An example of the refinement is shown in Figure 3.3.

Wings are by default seen as one part, for which one subgrid is generated. This subgrid is used to identify the source of forces in FlightStream. However, sometimes a user wants to break the wing in pieces, e.g. the front, connecting, and rear part of a box wing, to find their contributions. To make this possible, custom subgrids can be defined using pairs of subgrid names and the indices of the trunks they contain. The full set of wing mesh settings, including the expected type, is listed in Table 3.2.

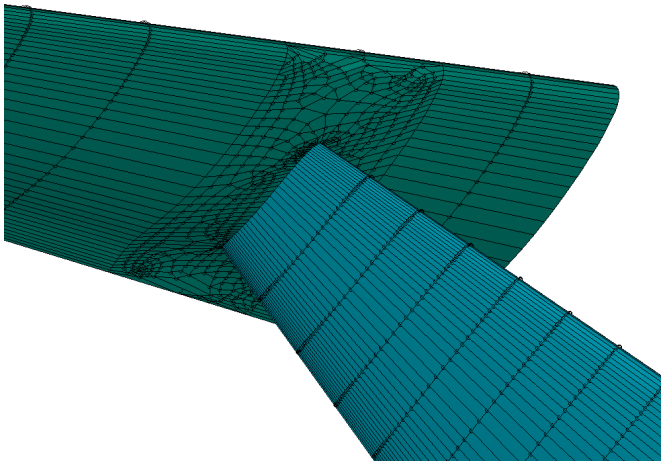


Figure 3.2: Unstructured patch on wing around intersection.

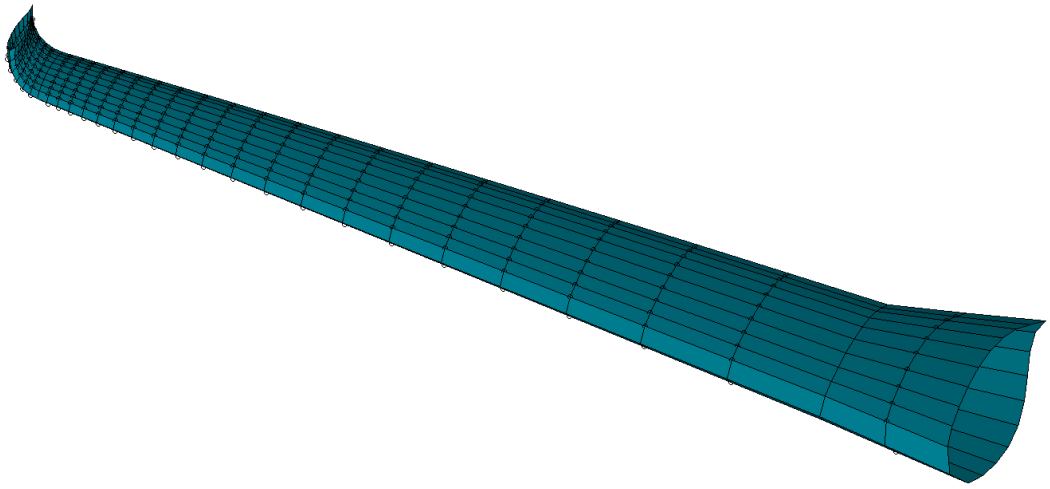


Figure 3.3: Example of wing refinement in a cornered trunk.

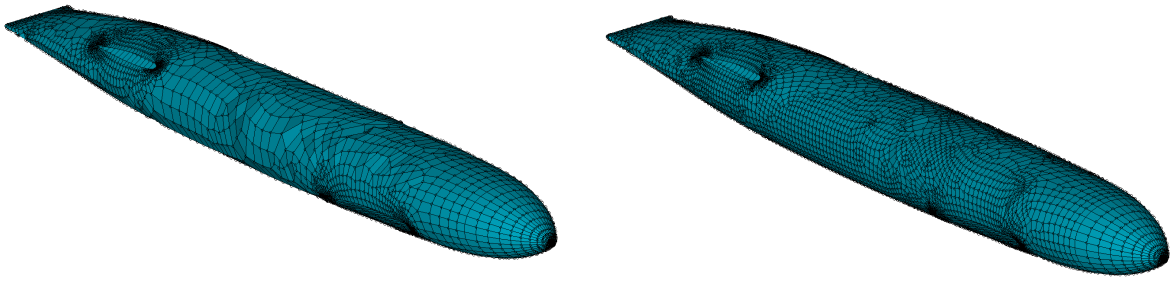
The fuselage is meshed using a hybrid strategy. The front part of the fuselage is using a structured mesh, while the remaining part of the side and the nose and tail faces are using an unstructured mesh. On the front part of the fuselage, the amount of points in circumferential and longitudinal direction can be given. The circumferential points are equispaced and the longitudinal points are placed using a half-cosine distribution, with an increase in element size along the fuselage length.

The unstructured patches of the fuselage are controlled using the minimum and maximum linear size of the faces, and the growth rate of the area between the mesh faces, as in the wing's unstructured patches. Next to this, a setting for the amount of segments per radius is exposed. With this setting, the discretisation of radii can be refined by increasing the value of the setting. The effect of this setting is shown in Figure 3.4.

A final addition to the fuselage is the introduction of refinement patch on the rear of the fuselage. For example, the PARSIFAL fuselage has a sharp side, to resolve this properly a fine mesh is required. When this is applied to the complete fuselage, a lot of extra faces are necessary, increasing the computational costs of the simulations. To prevent this, an extra split can be made. The part in front of this split is meshed using the general fuselage settings, while the patch behind the split is meshed with refined settings. This refinement can be seen in Figure 3.5. All exposed mesh settings for the fuselage are tabulated in Table 3.3.

Table 3.2: Set of wing mesh controls, including expected type.

Setting	Type
Number of chordwise points	Integer
Chordwise distribution type	String
Chordwise distribution growth rate	Float
Spanwise pitch	Float
Unstructured triangle minimum linear size	Float
Unstructured triangle maximum linear size	Float
Unstructured triangle growth rate	Float
Tip face triangle minimum linear size	Float
Tip face triangle maximum linear size	Float
Tip face triangle growth rate	Float
Spanwise refinements	List of dictionaries
Custom mesh groups	Dictionary



(a) Segments per radius = 1. (b) Segments per radius = 5.

Figure 3.4: Effect of changing the amount of segments per radius on the PARSIFAL fuselage.

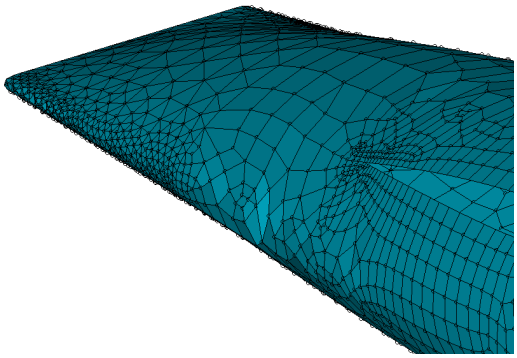
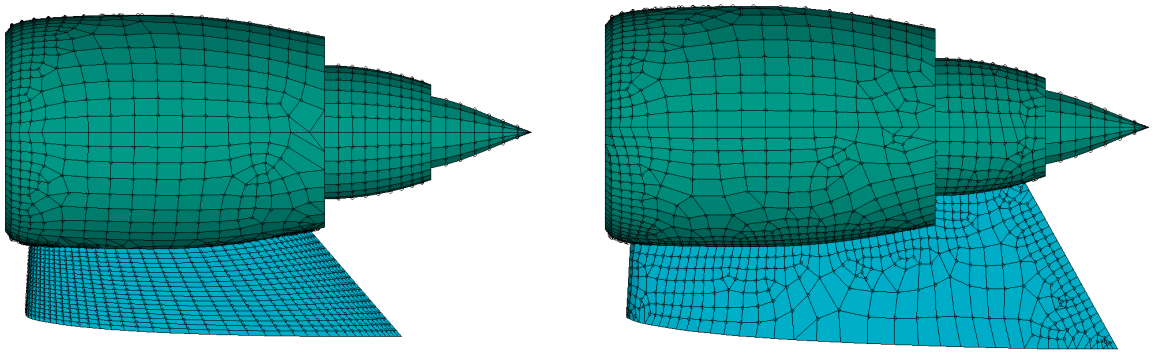


Figure 3.5: Fuselage refinement near the tail.

Table 3.3: Set of fuselage mesh controls, including expected type.

Setting	Type
Number of structured part circumferential points	Integer
Number of structured part longitudinal points	Integer
Unstructured patch minimum linear size	Float
Unstructured patch maximum linear size	Float
Unstructured patch growth rate	Float
Nose face minimum linear size	Float
Nose face maximum linear size	Float
Nose face growth rate	Float
Tail face minimum linear size	Float
Tail face maximum linear size	Float
Tail face growth rate	Float
Unstructured patch refinement settings	Dictionary



(a) Simple connected pylon using structured grid. (b) Complex connected pylon using unstructured grid.

Figure 3.6: Pylon mesh types for FligthStream analysis.

The mesh type used for the pylons depends on how it is connected to the engine. If there is a simple connection, i.e. only one face of the engine is intersected, the pylon is meshed using the strategy of an unintersected wing, thus fully structured. This pylon mesh is shown in Figure 3.6a. The settings for this type of pylon mesh control are the same as for the wing listed in Table 3.2, except no unstructured patch and tip face triangles are present.

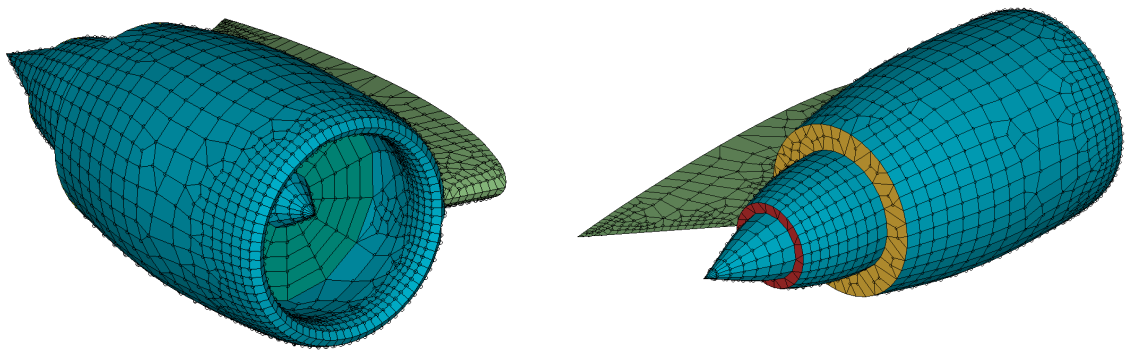
When the pylon is intersecting multiple faces of the engine, as in Figure 3.6b, the structured mesh cannot be created without an intricate splitting strategy. To circumvent this, an unstructured grid is used in this case. The unstructured grid is generated using the same equispaced leading and trailing edge points as for the structured grid. At the airframe intersection side, the same geometric distribution as on a wing is used. On the engine side, the points are placed equispaced. This is because the distribution from the airframe side cannot be copied, due to the corners in the pylon edge. The controls for this type of pylon are the same as in Table 3.2, also including the unstructured patch settings.

The engines are meshed using a hybrid mesh as well. In case of a full engine model, the spinner, fan, and core exhaust cone are discretised using a structured grid. This is possible since these faces are never intersected by a pylon. The other faces, i.e. the outside of the nacelle, inlet, and exhaust faces, use an unstructured grid. This strategy is chosen to ease the mesh generation when nacelle and exhaust faces are intersected. On the inlet face, it allows for a smooth transition from nacelle to fan. An example of this mesh is depicted in Figure 3.7. In case of a TFN, all faces are discretised using an unstructured mesh, as shown in Figure 3.8. The complete set of engine mesh control settings is listed in Table 3.4.

Next to the mesh, boundary conditions have to be provided to FlightStream. These are, amongst others, the trailing edges of lifting surfaces. These have to be indicated to shed vorticity into the wake and, thereby, generate aerodynamic loads. When no trailing edges are indicated, a fully potential flow will be calculated and no loads are generated.

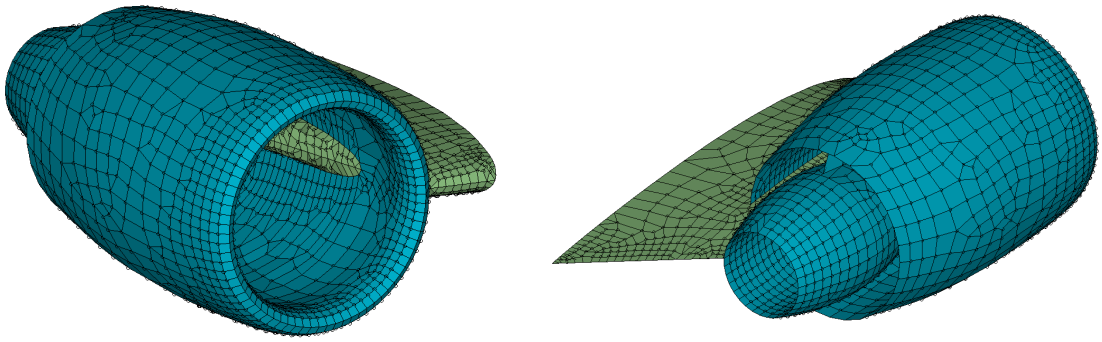
Together with the trailing edges, wake termination nodes can be given. These nodes lie on trailing edges, and would thus be the starting point of a wake strand. However, by making a node a wake termination node, no wake strand is started. This is necessary in the junction of a trailing edge and a non-lifting surface.

Flow conditions are given by providing the density, viscosity, sonic velocity, temperature, and pressure of the fluid under consideration. Together with the fluid free-stream velocity, also given by the user, this forms the flow field that is analysed.



(a) Front side. (b) Rear side.

Figure 3.7: Mesh for a full engine model.



(a) Front side. (b) Rear side.

Figure 3.8: Mesh for a TFN model.

Table 3.4: Set of engine mesh controls, including expected type.

Setting	Type
Number of structured part circumferential points	Integer
Structured part longitudinal pitch	Float
Unstructured patch minimum linear size	Float
Unstructured patch maximum linear size	Float
Unstructured patch growth rate	Float
Intersection edge pitch	Float

Finally, reference data for the calculation of dimensionless parameters has to be given. This data consists of a reference length, reference area, and reference velocity. These are used to calculate the Reynolds number and force and moment coefficients.

3.2.3. FlightStream validation

Since there is no prior use of FlightStream in combination with the MMG, a validation has to be performed to ensure proper functioning. The models created in the MMG have to be transferred properly to FlightStream, a mesh convergence study has to be done to match reference results, and the models for force calculation have to be chosen. How this is done and the results of this validation are shown in the following sections.

Methodology

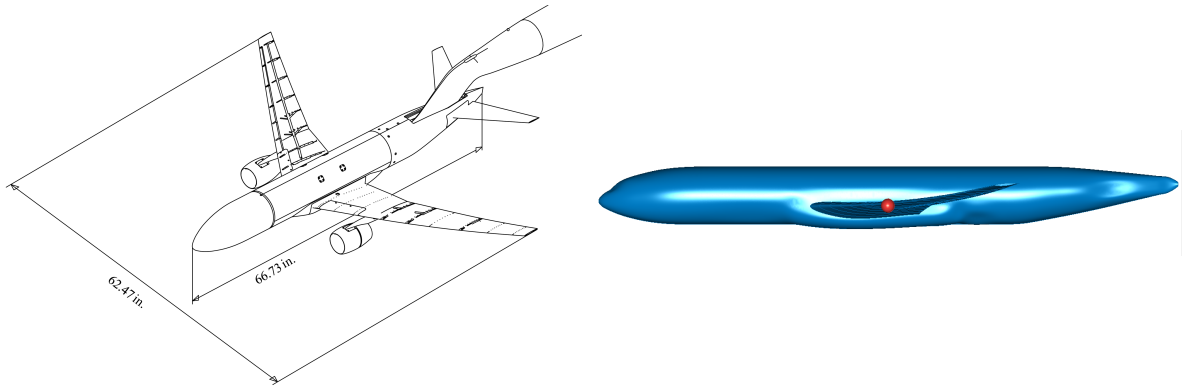
To validate the use of FlightStream, two validation cases are considered. The first is the NASA Common Research Model (CRM), in wing-body (WB) configuration. This is compared with wind tunnel data from the campaign ran by Rivers [22]. Since many aerodynamic solvers are tuned towards the CRM, a second validation case is used. This is a comparison with high fidelity calculations on the PARSIFAL PrandtlPlane box-wing and fuselage, performed by ONERA [23].

For the CRM case, the wind tunnel data from run 39 in the National Transonic Facility (NTF) is used. The validation is done with a simple geometry, only the main wings and body are considered. The geometry of the model is first created at full scale in the MMG, according to the information in [17]. Then, to match the wind tunnel model, a $1/37$ uniform scaling is applied. This scaled model is meshed and exported to FlightStream. The model, as used in the wind tunnel, including horizontal tail, pylons, and nacelles, is drawn in Figure 3.9a [22]. The flow conditions during the test are:

- $M = 0.7$
- $Re = 5 \times 10^6$
- $T = 293$ K
- $P = 162120$ Pa

The full set of fluid properties are calculated using the tables and formulas presented in Appendix E. To simulate the trip strips used in the wind tunnel test, the flow is set to turbulent for the complete domain. The moment coefficient is calculated using a reference point located 90.9 cm behind the nose and 5.2 cm below the center line of the fuselage, as represented in Figure 3.9b.

This validation case is also used to perform a mesh convergence study and to find the best models within FlightStream to use. The mesh convergence is performed by changing the fineness of the wing mesh and the fuselage mesh independently of each other, to see the effects of the meshes on the results, while taking the mesh requirements stated in Section 3.2.2 into account. For both the wings and fuselage, three levels are defined: coarse, medium, and fine. The independent change of the mesh parts is obtained by changing the mesh settings of the wing or fuselage, while keeping the other component's settings fixed at the 'fine' level.



(a) Drawing of the NASA Common Research Model [22].

(b) Location of the moment reference point in the NASA Common Research Model.

Figure 3.9: NASA Common Research Model used for validation.

Table 3.5: Settings used in mesh convergence of wing with resulting number of faces.

Mesh setting	Spanwise pitch	Number of faces on wing
Coarse	1.0	~3,100
Medium	0.75	~4,400
Fine	0.5	~6,000

Since the wing is known to require 80 points around a cross-section, this value is fixed. The settings for the wing tip faces are set to resolve the face without badly shaped elements. The only value changed is the spanwise pitch. The value of the pitch and the number of faces generated on a single wing are tabulated in Table 3.5. On the fuselage, the circumferential number of points, the maximum size of the faces, and the number of segments per radius are changed. The used settings and their corresponding number of faces can be found in Table 3.6. Note that these settings are for the full scale model, the sizes in the mesh are scaled by $1/37$.

The mesh convergence is assessed by the face quality metric in FlightStream, the match of the results with wind tunnel data, and the presence of oscillations in the lift curve. The latter turned out to also be a metric for the quality, after increasing the wing's spanwise pitch induced oscillations in the curve at high angles of attack.

Table 3.6: Settings used in mesh convergence of fuselage with resulting number of faces.

Mesh settings	Circumferential number of points	Face maximum size	Number of segments per radius	Number of faces on fuselage
Coarse	20	1.0	1	~3,100
Medium	25	0.8	3	~5,700
Fine	30	0.6	5	~8,400

Once the mesh convergence study is performed, the mesh of half an aircraft, meshed with the found settings, is tested. A half mesh can be used because only longitudinal flight is considered in this thesis. With the use of a half mesh, the computational costs are reduced if the mesh settings are fixed. However, it does also provide the opportunity to create a more detailed mesh with the same costs as before. The half mesh is created by providing only the faces on one side of the symmetry plane to the mesher. The results of a simulation with a half-model are compared with that of a full model, to see if there is a difference in results.

The choice for the lift model is based on the used boundary conditions. Since only the wakes of lifting surfaces are simulated, the pressure model has to be used. The vorticity model calculates lift by the shed vorticity in the wake, so it does not take the lift force of components without a wake into account. Next to this, the vorticity model does not take separation into account, while the pressure model does. So the pressure model adds extra information on the performance of the aircraft. The models for drag, moment, and viscous drag are chosen based on their match with the reference data. The combination of the medium wing and medium fuselage mesh is used to simulate all combinations of available models and the best matching combination is used further on.

In the second validation case, a half-model of the PARSIFAL MS1 box-wing with fuselage is simulated and compared with the results from a Reynolds Averaged Navier-Stokes (RANS) simulation. As in the reference case, a half mesh is used. This is required to have enough mesh faces to describe the geometry, without running into memory limits of the used computer. The flight conditions are:

- altitude = 11 km
- M = 0.79
- Re = 25×10^6

Using the tables and formulas from Appendix E, again, all required fluid properties are found.

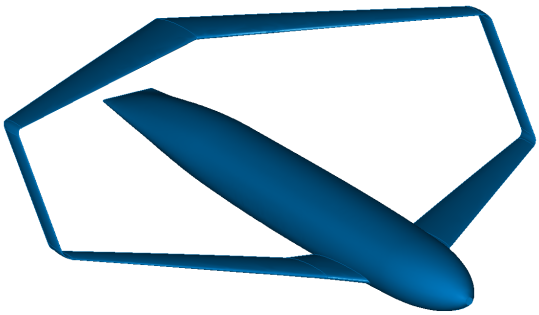
The geometric model and its mesh are shown in Figures 3.10a and 3.10b. The used mesh settings are based on requirements found in the mesh convergence study and are tabulated in Table 3.7. The corners of the box-wing are refined to ensure enough faces are used to describe the curvature, a close-up can be found in Figure 3.10c. The used settings lead to a total of ~11,000 mesh faces. For the calculation of the moment coefficient, the approximate centre of gravity (CG) location is used, this is shown in Figure 3.10d [23].

Results

Iterating over the three wing mesh levels, starting at coarse and increasing the fineness, showed better result when using the medium mesh settings. However, no additional gain was obtained using the fine settings. With the decrease in spanwise pitch, a difference of face sizes on both sides of the kink in the wing became smaller. This was especially visible in the face quality metric in FlightStream. A second effect was that the aspect ratios of the faces at the leading and trailing edge decreased with the decrease in pitch. Herewith, the results stabilised, especially at high angles of attack, where an oscillation was found in the lift curve while using the coarse mesh settings.

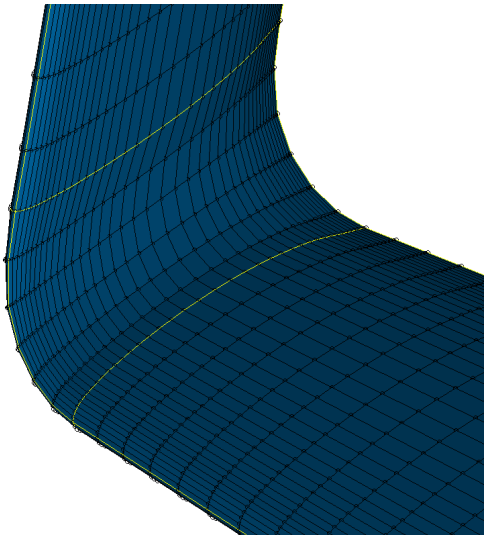
For the fuselage mesh, there is no strong influence on the results found during the convergence study. At least, no clear trend could be found while increasing the fineness. The match of C_L becomes better with an increase in fineness, however, the step from medium to fine only increases the match by less then 1%. C_D is matched equally using the medium and fine settings, and C_M is best matched using the medium settings. Since the finer mesh does not result in a closer match with the reference data, while requiring more resources, it is chosen to continue with the medium fuselage settings.

Next to the mesh convergence study, the available drag, moment, and viscous drag models are compared as well. The comparison showed that using the pressure model for drag underestimated the induced drag, and the vorticity model had a closer match, so vorticity is used further on. The viscous drag model, that calculates the skin drag, was best matching with the momentum integral model with an offset of 3.5%. The Reynolds averaged model returned an overestimation of almost 60%. For the calculation of the moment coefficient, the vorticity model had the best results and is thus used for further analyses.

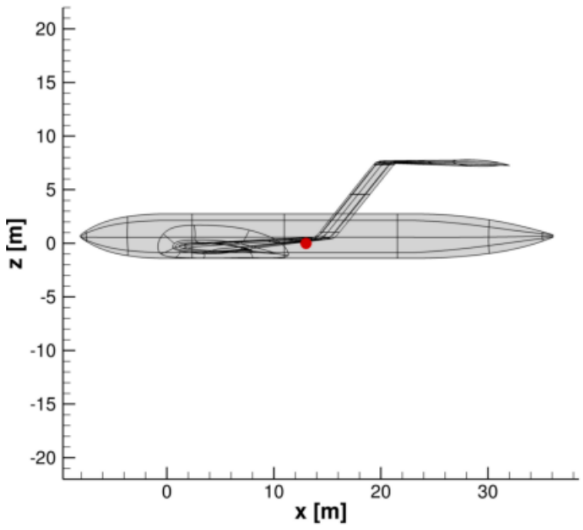


(a) Full geometry of PARSIFAL wing-body configuration.

(b) Half mesh of PARSIFAL wing-body configuration.



(c) Close-up of mesh refinement in box-wing corner.



(d) MS1 moment reference position (red dot) [23].

Figure 3.10: Geometry and mesh used in PARSIFAL validation case.

Table 3.7: Mesh settings used for PARSIFAL validation case.

Component	Setting	Value
Wing		
	Number of chordwise points	80
	Spanwise pitch	1.0
	Refinement pitch	0.2
Fuselage		
	Number of circumferential points	20
	Face maximum size	0.8
	Face minimum size	0.3
	Number of segments per radius	0.1
Fuselage refinement		
	Refinement offset	0.1
	Face maximum size	0.6
	Face minimum size	0.1
	Number of segments per radius	4

With the mesh settings and models selected, the wing-body configuration is simulated and the results are compared with wind tunnel data. The comparison is presented in Figure 3.11. Figure 3.11a shows that the lift is underestimated for $\alpha > 3°$. This is due to a known bug, discussed in Section 3.2.1. Because of this, Figures 3.11b to 3.11d zoom in on the range that is not affected by this bug. In the range of angles of attack where no separation occurs, the lift coefficient is found with a maximum difference of 5%.

The drag polar in Figure 3.11c shows that for negative values of C_L, the value of C_D is underestimated, while it is overestimated for positive values of C_L. The overestimation is approximately 6%. Furthermore, the location of the minimum drag point is shifted with respect to the reference data. Since the value of the minimum drag is found within $1.4 \times 10^{-4}\%$, the difference should follow from the induced drag, when using the two term drag formula from Equation (3.16). The plot shows that the value of C_L at which minimum drag is generated is off. The value of $C_{L_{\min}}$, the lift coefficient for minimal drag, is affected by the wing twist. Rivers showed that there is a mismatch between the wind tunnel model and computational geometry [24]. During the wind tunnel test, the model did not have a fixed twist, as in the computational geometry, but it changed with angle of attack. This change in twist is the likely source for the mismatch in induced drag.

$$C_D = C_{D_{\min}} + \frac{\left(C_L - C_{L_{\min}}\right)^2}{\pi A e} \tag{3.16}$$

The moment curve in Figure 3.11d shows an underestimation of C_M. Next to this, the stability, indicated by C_{M_α}, is mismatched, for $\alpha < 1°$. A breakdown of the moment coefficient into the contributions made by the wings and fuselage shows expected behaviour from the fuselage. For the complete range of angles of attack, it destabilises the aircraft ($C_{M_\alpha} > 0$). The wings have a stabilizing effect over the full range of angles of attack. However, at $\alpha = -3°$ this effect is almost zero and it increases with angle of attack. An investigation of the pressure distributions of wing sections, which allows to find the spanwise lift distribution, shows that for $\alpha < 1°$, the lift generation is too high at the root and too low at the tip of the wing. This is also an attribute of the wing twisting during the wind tunnel test. The distribution mismatch does not affect the total amount of lift produced, as seen in the lift curves, but because of the wing sweep it does influence the pitching moment. Since the root of the wing is located in front of the reference point, while the tip is located aft of the reference point. An offset in

the lift distribution will, because of this sweep, affect the moment coefficient.

The CRM geometry is also analysed using a half mesh, modelling only one half of the aircraft. The three coefficients are found within a 15% difference with respect to the full mesh results. Lift and drag coefficients are overestimated and the moment coefficient is further underestimated with respect to the reference data. The difference is due to the change in fuselage mesh, since it is cut in half. The wing mesh is not changed, and the results for a single wing are the same as in the full mesh case. With this effect in mind, and thus ensuring the fuselage mesh is properly refined at the symmetry plane, a half mesh can be used.

In the second validation case, the PARSIFAL geometry is simulated and compared with data from a RANS-simulation by ONERA [23]. The resulting lift, drag, and moment curves are displayed in Figure 3.12. Figure 3.12a shows that the lift is overestimated up to $\alpha = -0.5°$, where separation of the front wing sets in, by less than 10%. After the flow separation, the lift is increasingly underestimated. The dip at $\alpha = 1.0°$ is due to a loss of lift on the front wing. Probably, this is due to separation effects on this wing.

The drag polar in Figure 3.12b shows an underestimation of the drag coefficient by on average 15%, while the value converges towards the reference with increasing C_L. This is opposite of the expected behaviour. Since with increasing C_L the contribution of wave drag increases, an increase in difference with increasing C_L was expected, because FlightStream does not capture shock waves. FlighStream returns a value for the skin friction drag coefficient of 11 drag counts (1 count is $C_D = 1 \times 10^{-4}$), while the drag breakdown in the reference data shows 9.5 counts. This is a 15% overestimation of the friction drag. The underestimation of the total drag therefore follows from the missing wave drag and a mismatch in induced drag.

The moment coefficient is underestimated in this case, as can be seen in Figure 3.12c. Since no breakdown of contribution per aircraft component is given, no exact cause for the mismatch of 40% can be found. However, the most likely causes are wrong estimation of lift distribution between the front and rear wing and missing drag on the rear wing. Since the rear wing is located high above the reference point, the contribution of its drag towards the moment can be considerable.

FlightStream shows relatively close matching of force coefficients with high fidelity reference data. During the validation, the calculated moment coefficients were off, but for both cases, sources of the mismatch are identified. The two cases presented are on the edge of the operating range of FlightStream. Although FlightStream incorporates compressibility effects, the high Mach numbers found during these studies create effects not captured by FlightStream, i.e. shock waves. Also, the flow separation issues limit the use of FlightStream at this moment. However, when this is all kept in mind, FlightStream can be used for further research.

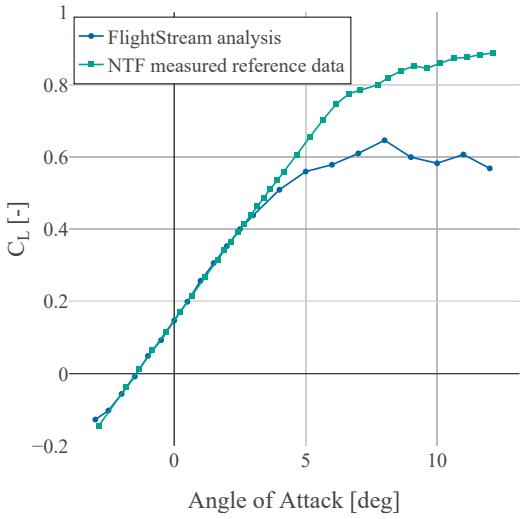
3.2.4. VSAERO introduction

VSAERO is a 3d panel solver that solves the potential flow problem to find the pressure distribution on a body. The wake is calculated using an iterative wake relaxation procedure, while viscous effects are treated in an iterative loop coupling potential flow and integral boundary layer calculations [25]. The program originated in the 1980s at NASA and the currently used version v7.9 released in 2016. VSAERO has been used within the Flight Performance section for a long time. Validations with wind tunnel data have been performed by Raju Kulkarni [26] and Proesmans showed the ability to model powered engines [5].

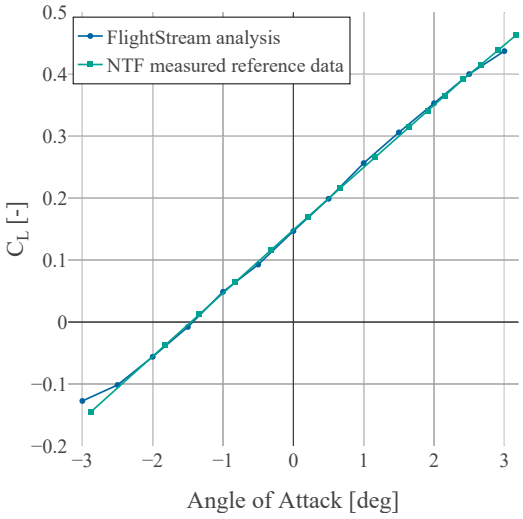
The study of Raju Kulkarni shows that VSAERO does not capture separation effects, maintaining a linear increase of lift coefficient with angle of attack for all used angles of attack. This does limit the usable range of angles of attack. The solved potential flow problem is incompressible, but Karman-Tsien [20] and Prandtl-Glauert [19] corrections are available to simulate compressible flow. These models, however, do not capture shock waves.

3.2.5. VSAERO view

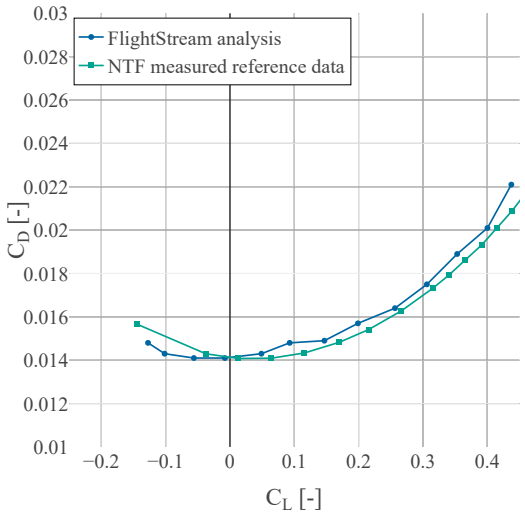
The inputs for a VSAERO analysis consist of solver settings, the geometry discretised as a mesh, and a description of the wake. Within the Flight Performance section, a best practice is formed of using a fully structured mesh, which can require an intricate splitting strategy when multiple wings are intersecting each other and the fuselage. To create a fully structured mesh, the geometry must be split into a set



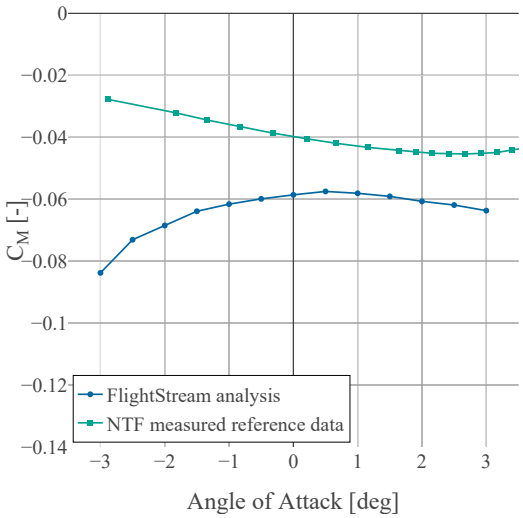
(a) Lift coefficient for full range of measured angles of attack.



(b) Lift coefficient for range of angles of attack not affected by bug.

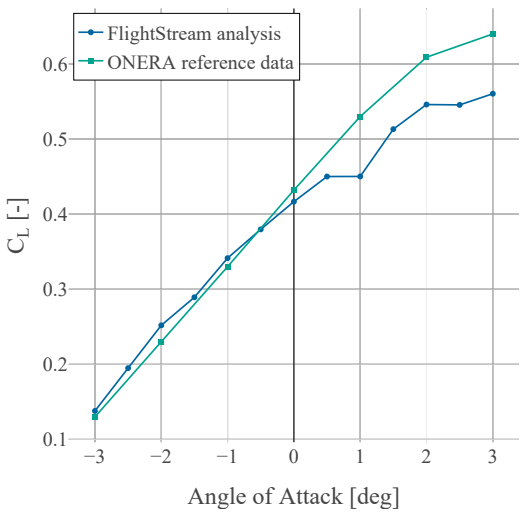


(c) Lift-drag polar for range of angles of attack not affected by bug.

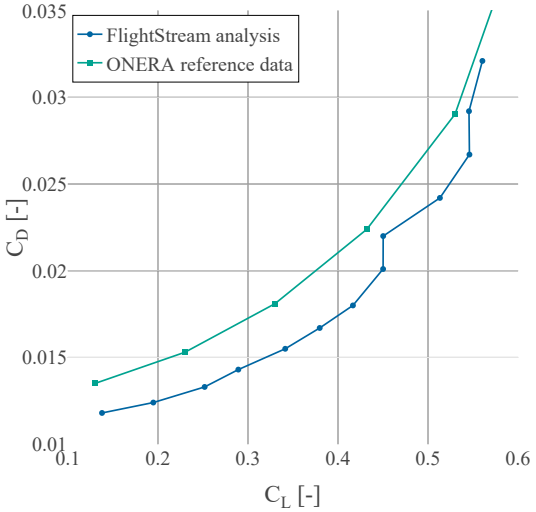


(d) Moment coefficient for range of angles of attack not affected by bug.

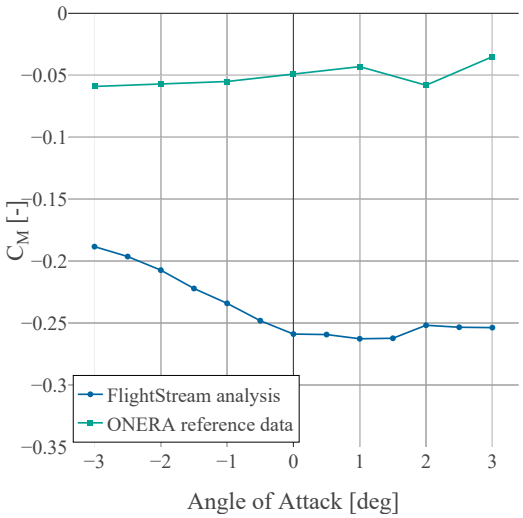
Figure 3.11: Validation curves using CRM wing-body configuration and measured data from [22] at $M = 0.7$, $Re = 5 \times 10^6$.



(a) Lift coefficient curve.



(b) Lift-drag polar.



(c) Moment coefficient curve.

Figure 3.12: Validation curves using MS1 wing-body configuration and simulated data from [23] at $M = 0.79$, $Re = 25 \times 10^6$.

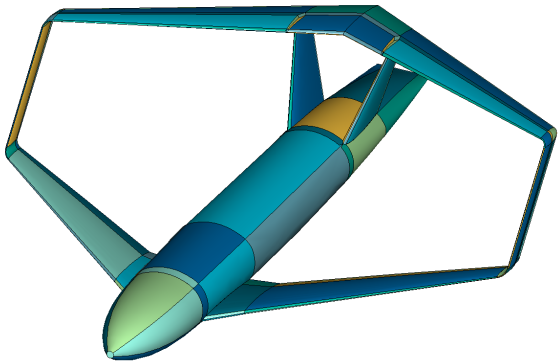


Figure 3.13: Quadrangle patches applied to PARSIFAL geometry.

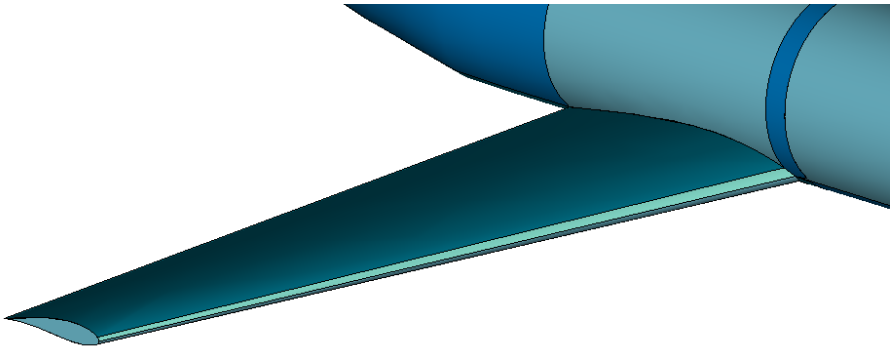


Figure 3.14: Quadrangle faces on unintersected wing.

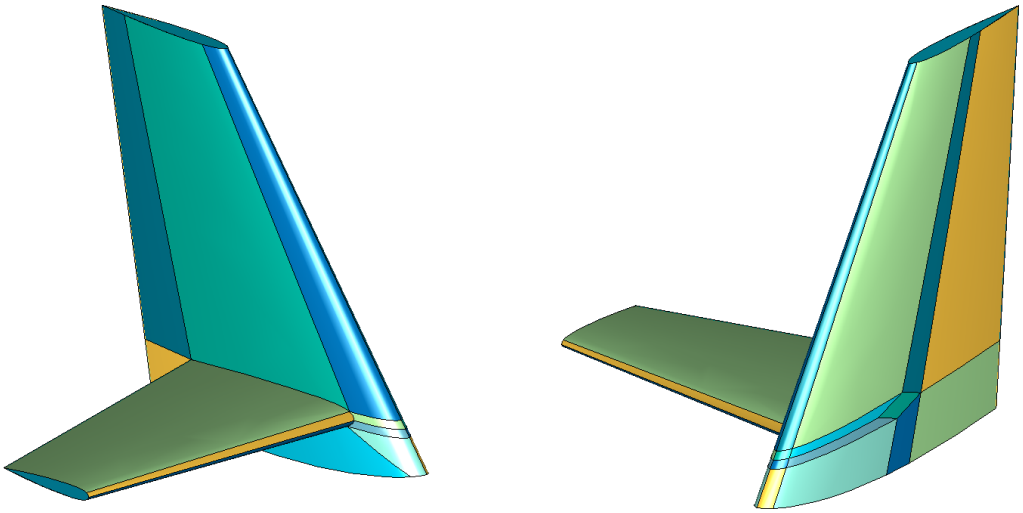
of quadrangle patches. An example of a split aircraft geometry is found in Figure 3.13. Below, the splitting of wings and the fuselage is described. The splitting of engines is described by Proesmans [5].

The geometry of an unintersected wing itself only contains quadrangle faces, and as such, does not have to be split. However, to ensure no sharp angles are created on the fuselage, two splits are added with an offset from the leading edge, one on both the top and button half of the wing. This simple split is shown in Figure 3.14.

When a wing is intersected by another wing, the intersecting wing still uses the splits offset from the leading edge. These are propagated on the intersected wing in the spanwise direction and towards the leading edge of the intersected wing, see Figure 3.15a. This image also shows that at the trailing edge of the intersecting wing, a spanwise split and a split towards the trailing edge of the intersected wing are placed. On the intersected wing, the offset split at the leading edge connects the offset points on the intersected and intersecting wings.

In case of a symmetric intersection, e.g. a vertical tail intersected by a horizontal tail, these splits are present on both sides of the wing. When this is not the case, e.g. a pylon intersecting a wing, a so called splitter diamond is placed on the unintersected side. This is required to ensure that on both sides of the wing the same amount of faces, seen in the spanwise direction, are present. The splitter diamond is shown in Figure 3.15b

To ensure the fuselage consists of quadrangle patches only, splits are applied at the intersections of fuselage and wing. These splits are shown in Figure 3.16. At the leading and trailing edge of the wing, a split is added to the nose and tail of the fuselage. The splits on the wing, that are offset from the leading edge, are propagated in circumferential direction on the fuselage. Also the trailing edge of the wing is added as circumferential split on the fuselage. Furthermore, at the leading edge of the wing, a circumferential split is added. This split starts with a component in the fuselage's longitudinal direction, to ensure no sharp corners are created at the intersection's leading edge.



(a) Intersected side. (b) Unintersected side.

Figure 3.15: Intersected wing with splitter diamond.

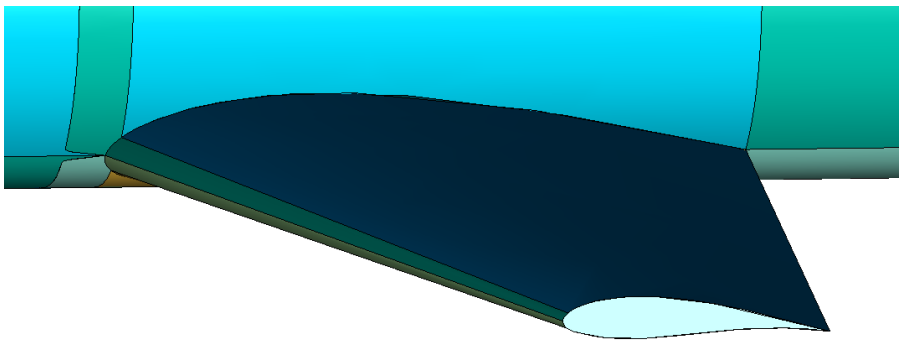
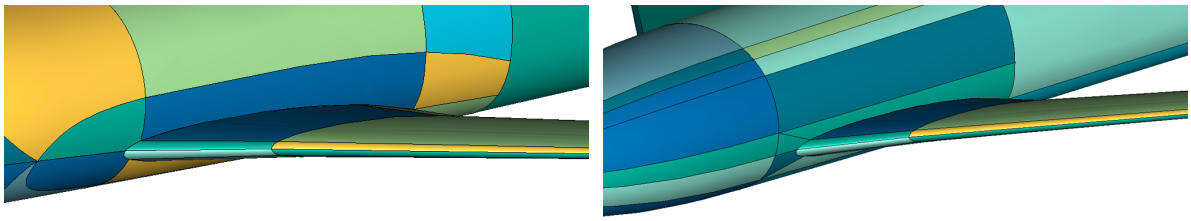


Figure 3.16: Fuselage split near wing leading edge.



(a) Wing shape offset type. (b) Square angled type.

Figure 3.17: Fuselage splits for c-grid creation around intersection.

Another way of ensuring no sharp angles are created around the wing's leading edge is to place either a c-grid around the intersection. Two versions of this are tested. One using a scaled projection of the intersection on the fuselage and a second using a rectangular bounding box. Both versions are depicted in Figure 3.17. During this thesis, an effort is done to use this type of c-grids with different aircraft configurations. However, because of difficulties encountered, this effort is abandoned and the splitting strategy described above was used.

The first issue was the placement of a boundary around a wing that is located very low on the fuselage. This makes the boundary meet with the edge that is at the bottom of the fuselage, located at the symmetry plane. The boundary area on the bottom of the wing had to be adapted to fit between the wing and the symmetry plane. However, this did not provide enough area to create a mesh around the wing, without creating skewed faces.

The second issue was on wings being close to each other, e.g. the vertical tail and pylon of the MS1. The boundaries around the wings were intersecting, running into the same issue as for the case near the bottom of the fuselage. An attempt to merge these intersecting boundaries into one did not return a properly shaped mesh.

The final issue encountered during the implementation of the c-grids was the application of edge controls on circumferential edges. To form a mesh of only quadrangles, each circumferential section of the fuselage requires the same amount of nodes. However, this could not be ensured because of the possible changing amount of faces, thus split edges introducing extra nodes, around a circumference. To fix this, extra longitudinal splits have been added, to ensure the same amount of faces around each circumferential section of the fuselage. This, however, added split lines through the boundary grids of other wings, for some configurations. This, again, broke the creation of a full quadrangle mesh, since triangular patches were created.

Once the body is split into quadrangle patches, a structured mesh is applied to it. This is done by setting the amount of points in chordwise direction and spanwise pitch of wings. On fuselages, the longitudinal pitch and number of circumferential points can be specified by the user. When two components are intersecting each other, the intersecting component defines the amount and distribution of the points on both components. This can be seen in Figure 3.18b, where the wing mesh is projected on the fuselage.

Next to the geometry, discretised as mesh, VSAERO requires a definition of the wake. The wake is a representation of the vorticity shed into the flow by the body. It consist out of wake lines that start at a location given by the user. The wake lines are built up from segments that connect wake grid planes.

The placing of these planes has to done with care. A too coarse spacing can result in badly resolved wake behaviour, which affects the forces that act on the body. Different wake types are available, e.g. a type for wing wakes and types for jet engine wakes.

The freestream conditions are given by a Mach number and a Reynolds number. The Reynolds number is calculated using a reference length given by the user. The user also has to provide a reference area, used for the calculation of force and moment coefficients.

3.3. Best practice trade-off

In this trade-off, five practices using two solvers are compared with each other to find the best option for aerodynamic analysis in the preliminary design phase. This trade-off takes not only engine integration analysis into account, but also looks at the analysis of clean aircraft configurations.

The two solvers used are FlightStream and VSAERO, both introduced in Section 3.2. Since the interference effects between airframe, nacelle, and plyon might be the source of the bad design sensitivity found by Proesmans [5], for both solvers an engine installation with and without pylon is considered. A drag estimation for the pylon found using the Raymer method (see Section 3.1.2) is added to the results of the simulations without pylon, to asses the complete drag addition of the installation. Within the limitations of the solvers, the following practices are selected:

1. TFN with pylon in FlightStream
2. TFN without pylon in FlightStream
3. Powered nacelle with pylon in VSAERO
4. Powered nacelle without pylon in VSAERO
5. TFN without pylon in VSAERO

The FlightStream practices are limited by a bug in the used version of FlightStream, see Section 3.2.1, therefore only TFNs are used. The powered nacelle with pylon in VSAERO is the practice used by Proesmans [5] and considered as a baseline in this comparison. A practice of TFN with pylon in VSAERO is missing because creating a mesh for this practice with the requirements discussed in Section 3.2.5 failed.

In the following sections, first the comparison criteria and their weights are introduced. Then, the results of the comparison of the solvers and practices are discussed and, finally, the solvers and practices are scored to find a best practice.

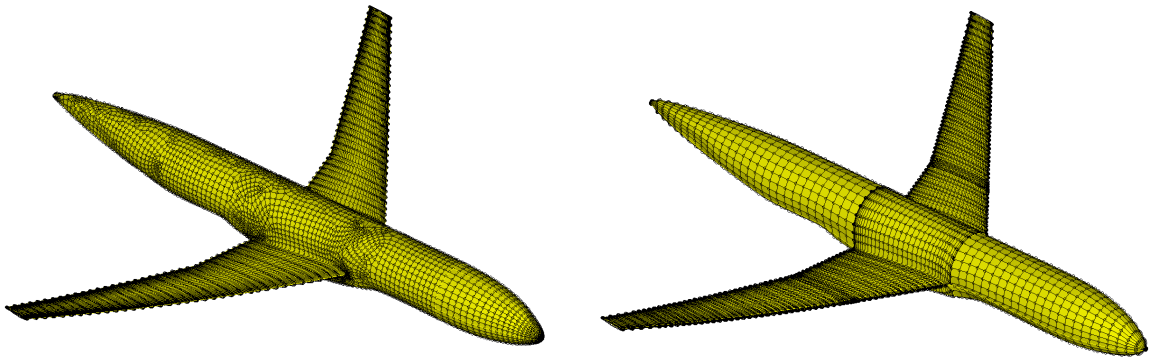
3.3.1. Comparison criteria

The practices under consideration are scored on five criteria, three on tool level and two on practice level. The tools are evaluated for their speed, accuracy of a clean aircraft simulation, and ease-of-use. The different practices are compared on accuracy of the estimation of engine installation effects and the design sensitivity.

The speed of both solvers is quantified as the time it takes the method to return a solution. This is the time from the start of pre-processing, up to when a results file is complete. The time required by the panel methods will be measured using the CRM WB geometry. For both solvers, a mesh with 11,000 faces is created, using their respective meshing methods. The resulting meshes are shown in Figure 3.18. A single angle of attack, $\alpha = 0$, is simulated at $M = 0.25$ and $Re = 3 \times 10^6$. No explicit test is performed for the practices, since it is known that amount of time required for a simulation scales with number of panels. Therefore, it is assumed that a simulation without pylon will require less time than one with pylon.

The accuracy of the clean aircraft performance is obtained by a comparison of the CRM WB simulation results with wind tunnel data [22]. This is the same study as used in Section 3.2.3.

The ease-of-use is broken down into three parts: the ease of pre-processing, the ease of interfacing with the solver, and the ease of post-processing results. The interfacing is how one can communicate with the solver using Python, to make a coupling directly from the MMG. The scores are based on the steps required to pre-process geometry before being able to run the solver, the ease of automating the solver, and the ease of retrieving data from the results files. In the scores for pre- and post-processing, both the manual actions required and ability to automate are taken into account.



(a) FlightStream mesh. (b) VSAERO mesh.

Figure 3.18: Meshes of the CRM containing 11,000 faces used for speed comparison.

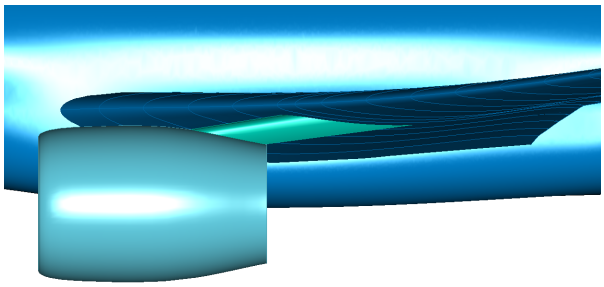


Figure 3.19: Sting pylon used for CRM comparison study.

During Rivers' wind tunnel study, also a wing-body-pylon-nacelle (WBPN) configuration was placed in the wind-tunnel. That is used to find the engine integration effects on the aerodynamic performance. A drawing of the wind tunnel model in the WBPN is shown in Figure 3.9a. Since the MMG is not capable of modelling the pylon used in the wind tunnel test correctly, a simplified sting is used to connect the nacelle to the airframe, shown in Figure 3.19. Because of this different pylon, no exact match with the reference data can be expected. This practice does, however, provide an insight on how the solvers resolve the presence of a pylon. To find the accuracy of the integration effects estimation, the results of a WB(P)N simulation are compared with a simulation of a WB configuration for all practices. The difference between the two are the installation effects. These are compared with the effects found during the wind tunnel campaign.

To find the design sensitivity of the practices, the effects of changing the engine size of the PARSIFAL MS1 aircraft is investigated. The engine size follows from engine cycle parameters, BPR and OPR in this case. Using $8 \leq \text{BPR} \leq 12$ and $39 \leq \text{OPR} \leq 45$, the engine length is in the range of 5.9 m to 6.5 m and the diameter changes between 2.5 m and 3.0 m. The effects on lift, drag, and moment coefficient are compared with that of a clean aircraft, using flow conditions $M = 0.3$ and $Re = 27 \times 10^6$. These conditions are in flight found after take-off or before landing. These conditions are chosen to ensure no shocks are present on the aircraft, to stay within the operating limits of the panel solvers. Since no reference data is available for this case, and no comparable studies are found, this criteria is scored on how the results of the practices match expected results.

Since the engine is located in the downwash of the front wing, see Figure 4.1, it is subjected to a negative angle of attack. Therefore, negative lift production is expected. With an increase in engine

Table 3.8: Weights used for trade-off criteria and subcriteria.

Criterion	Subcriterion	Weight	Subweight
Speed		0.5	
Clean accuracy		1	
Ease-of-use		0.5	
	Pre-processing		1
	Interfacing		1
	Post-processing		1
Installation accuracy		1	
Design sensitivity		1	

Table 3.9: Time required in seconds per solver to solve a case with 11,000 faces.

	Pre-processing	Solving	Total
FlightStream	6	119	126
VSAERO	25	206	231

size, the amount of produced lift is expected to increase. Next to this, an increasing drag coefficient with engine size is expected. The negative lift created by the engine and located aft of the reference point, should induce a pitch up moment that increases with engine size. The contribution of the pylon is neglected, based on the findings in Section 3.1. This expected effect is assuming the nacelle does not influence the forces produced by the airframe significantly, when compared to the forces added by the nacelle and pylon.

Each main and sub-criterion is given a weight factor $w_{criterion}$ and $w_{subcriterion}$, respectively, to take their importance into account. The weights of the criteria are tabulated in Table 3.8. Speed and ease-of-use have a lower weight, since it is worth to wait longer or perform more (difficult) actions to get to more accurate results. Each criterion is scored and the total score of each practice is calculated using Equations (3.17) and (3.18). For each practice, the practice specific criteria and the solver criteria are both taken into account.

$$score = \sum score_{criterion} \cdot w_{criterion} \tag{3.17}$$

$$score_{criterion} = \sum score_{subcriterion} \cdot w_{subcriterion} \tag{3.18}$$

3.3.2. Results

In this section, the results of the comparison of the different analysis solvers and practices are presented per criterion.

Speed

The results of the speed test for the solvers are tabulated in Table 3.9. It shows that for solving a single angle of attack, FlightStream is the faster option. Next to this, FlightStream can reuse solutions during a sweep, e.g. simulating multiple angles of attack, which can lower the time required for additional solutions of the same geometry to approximately 15 seconds. The difference in pre-processing time most likely follows from the extra steps required for VSAERO, e.g. generating the wakes of the aircraft. The solving time difference is likely to come from the fact that VSAERO cannot use multiple processor cores, while FlightStream can use the full computing power.

Table 3.10: Coefficient comparison for CRM WB. Panel solvers compared with wind tunnel data [22] at $M = 0.7$, $Re = 5 \times 10^6$.

	C_{L_α}	$C_{L_{\alpha=0}}$	$C_{D_{\min}}$	$C_{L_{\min}}$	k	c_2	$c1$	c_0
FlightStream	0.0989	0.1499	0.014	-0.032	0.0335	-0.0017	0.0025	-0.0581
	-1.88%	1.49%	0.00%	-200%	-22.45%	-525%	186%	-47.09%
VSAERO	0.098	0.1338	0.0126	0.062	0.0628	-0.0015	-0.0084	-0.0592
	-2.78%	-9.41%	-10.00%	93.75%	45.37%	-475%	-190%	-49.87%
Wind tunnel	0.1008	0.1477	0.014	0.032	0.0432	0.0004	-0.0029	-0.0395

Clean aircraft performance accuracy

To find the accuracy of the panel solvers, three equations are used: Equations (3.16), (3.19) and (3.20). The coefficients of these equations are obtained from interpolations of the FlightStream, VSAERO, and wind tunnel results. In Table 3.10, the result are tabulated. For both simulations, the relative error with respect to the wind tunnel data, determined with Equation (3.21), is added below values of the coefficients.

$$C_L = \alpha C_{L_\alpha} + C_{L_{\alpha=0}} \tag{3.19}$$

$$C_M = c_2\alpha^2 + c_1\alpha + c_0 \tag{3.20}$$

$$\delta = \frac{c_{simulation} - c_{wind\,tunnel}}{|c_{wind\,tunnel}|} \cdot 100\% \tag{3.21}$$

A comparison of the results in Table 3.10 shows that FlightStream has a closer match than VSAERO for most of the coefficients. VSAERO only has a better match for $C_{L_{\min}}$ and the c_2 coefficient of Equation (3.20). The probable sources of the mismatch in the drag and moment coefficients for FlightStream is discussed in Section 3.2.3. For the offset in VSAERO, the same line of reasoning holds.

The large offsets of the moment curves, returned by both solvers, indicate that these panel methods are a bad predictor for the moment coefficient of this aircraft. Therefore, during later comparison, the moment coefficient will be weighted less.

Ease-of-use

FlightStream requires little pre-processing. Since it is able to use structured, unstructured, and hybrid meshes, the meshing process is straight forward, no complex splitting strategies are required. The wake is calculated inside FlightStream, requiring only to indicate with mesh edges do shed a wake.

A case can be run using both the FlightStream GUI manually and the scripting API to automate tasks. The scripts used are plain text files with fully written out named commands, making it easy to understand, write, and modify the file. FlightStream can be started and instructed to run a script file using the command line, which can be called as a sub-process in Python, allowing easy interfacing.

The post-processing of the results consist of reading out force and moment coefficients from a file, or investigating flow properties on and around the aircraft in the GUI. Also, the result files are plain text files that are easy to parse by script. The drag coefficient in this file is divided into a induced drag C_{D_i} and skin friction drag C_{D_0}. This distinction provides more information on where drag changes follow from when comparing different configurations. The visualisation of flow properties can be automated using script commands. However, since the camera cannot be positioned, nor screenshots be exported by script commands, creating images of the flowfield is a manual job.

VSAERO requires more pre-processing. Since best results are obtained using a fully structured mesh, an intricate splitting strategy is required to obtain quadrangle faces all over the geometry. Next to this, wake strands have to be pre-calculated to provide as input to VSAERO, increasing the amount of work required for pre-processing.

The input file given to VSAERO is written in plain text, using cards that indicate the subject of inputs. However, the inputs themselves are not named, making it difficult to understand the file without

Table 3.11: Coefficient comparison for CRM WB(P)N. Analysis practices results and their error with respect to wind tunnel data [22] at $M = 0.7$, $Re = 5 \times 10^6$.

	C_{L_α}	$C_{L_{\alpha=0}}$	$C_{D_{\min}}$	$C_{L_{\min}}$	k	c_2	c_1	c_0
FlightStream with pylon	0.1016	0.1328	0.0155	0.055	0.0527	-0.0011	0.0104	-0.0578
	-2.21%	4.73%	-7.19%	-5.17%	11.89%	-1933%	104%	-11.37%
FlightStream without pylon	0.1017	0.1411	0.0181	0.037	0.0456	-0.0009	0.0089	-0.0563
	-2.12%	11.28%	8.38%	-36.21%	-3.18%	-1600%	74.51%	-8.48%
VSAERO with powered engine	0.1004	0.1202	0.0107	-0.02	0.046	-0.0015	-0.0025	-0.069
	-3.37%	-5.21%	-35.93%	-134%	-2.34%	-2600%	-149%	-32.95%
VSAERO with TFN	0.1013	0.1272	0.0137	0.12	0.0333	-0.0015	-0.0024	-0.067
	-2.50%	0.32%	-17.96%	107%	-29.30%	-2600%	-147%	-29.09%
Wind tunnel	0.1039	0.1268	0.0167	0.058	0.0471	6.00E-05	0.0051	-0.0519

consulting the manual. VSAERO does not offer a GUI, which requires the user to interact with it by using the input file only. Similar to FlightStream, the command line can be used to start VSAERO, which allows for easy interfacing.

The post-processing is the same as for FlightStream. A plain text output file is easy to parse to obtain force and moment coefficients. The flow field can be visualised using OMNI3D program. In this program, no automation of image creation is available, either.

Installation accuracy

Table 3.11 shows the coefficients found using the four practices for an aircraft with engines installed. In Table 3.12, the difference in coefficients following from the engine installation are tabulated. The results are corrected for thrust and pylon drag if necessary. The thrust correction is done by only including the fan cowl and inlet face in the calculation of the aircraft performance. The forces and moments on faces that are not present on a TFN are excluded. The pylon drag correction is that of the Raymer textbook.

In the following comparison, the results of including both powered nacelle and pylon in VSAERO, as done by Proesmans, is not included, but merely used as baseline. The results of Proesmans' study showed good agreement of lift and moment coefficient, but an underestimation of the drag coefficient. This underestimation follows from the method used to model TFN, namely setting inlet and exhaust velocities to match the local flow. In Section 2.2, the source of mismatches using this method is explained.

The coefficients in Table 3.11 show that the slope of the lift curve is best found by the FlightStream practices, although these overestimate the lift at zero angle of attack with 5% and 10%. When combining the results of both slope and lift at zero angle of attack, the VSAERO with TFN practice approximates the lift curve best.

The drag polar is best matched by the FlightStream practices. VSAERO underestimates the zero-lift drag added by the engine with more than 15%, while FlightStream is within 10% error. The value of $C_{L_{\min}}$ is also better estimated by FlightStream. For this, the practice with pylon is much closer to the wind tunnel value than the pylonless practice. The efficiency factor k is best found by VSAERO with a powered engine, FlightStream without pylon being a close second. Taking all coefficients into account, FlightStream with pylon shows the best match with the wind tunnel data.

Closer investigation of the FlightStream results show that the difference between the pylon and pylonless practice follow from the forces on the nacelle. The forces and moments on the airframe are within 5% of each other. The vertical forces on the pylon and nacelle are in opposite direction, such that the force of the two combined is the same as for the nacelle without pylon. The drag difference follows from the different flow on the nacelle. In the practice with pylon, the pylon contribution is very small in comparison to the nacelle contribution.

When looking at the moment curve, FlightStream without pylon has the best match. Followed closely by FlightStream with pylon, and both VSAERO practices are far off.

Table 3.12 shows the change of the coefficients due to the engine installation. In these results, the same outcomes as in Table 3.11 can be found. The practice of FlightStream without pylon has overall

Table 3.12: Change in coefficient comparison for CRM WB(P)N. Analysis practices results and their error with respect to wind tunnel data [22] at $M = 0.7$, $Re = 5 \times 10^6$.

	ΔC_{L_α}	$\Delta C_{L_{\alpha=0}}$	$\Delta C_{D_{\min}}$	$\Delta C_{L_{\min}}$	Δk	Δc_2	Δc_1	Δc_0
FlightStream with pylon	0.0027	-0.0171	0.0015	0.087	0.0192	0.0006	0.0079	0.0003
	-12.90%	18.18%	-44.44%	235%	392%	276%	-1.25%	102%
FlightStream without pylon	0.0028	-0.0088	0.0041	0.069	0.0121	0.0008	0.0064	0.0018
	-9.68%	57.89%	51.85%	165%	210%	335%	-20.00%	115%
VSAERO with powered engine	0.0024	-0.0136	-0.0019	-0.082	-0.0168	0	0.0059	-0.0098
	-22.58%	34.93%	-170%	-415%	-531%	100%	-26.25%	20.97%
VSAERO with TFN	0.0033	-0.0066	0.0011	0.058	-0.0295	0	0.006	-0.0078
	6.45%	68.42%	-59.26%	123%	-856%	100%	-25.00%	37.10%
Wind tunnel	0.0031	-0.0209	0.0027	0.026	0.0039	-0.00034	0.008	-0.0124

the best match. However, the change of some coefficients is better captured by other practices.

Design sensitivity

Figures 3.20 to 3.23 show the sensitivities of C_L, C_D, and C_M on aircraft level for the practices under consideration. For the results of the powered engine with pylon the reader is referred to [5]. The presented results do not contain a correction for pylons, but are corrected for thrust.

With the use of FlightStream, in Figures 3.20 and 3.21, better sensitivity is found when no pylon is included. When a pylon is included in the simulation, the figures show no clear trend with engine size but rather noisy results. Furthermore, the expected behaviour where the largest engines have the largest effect, are reversed. The practice without pylon shows fluctuations in the moment coefficient, but those are in the order of $\Delta C_M = 1 \times 10^{-4}$. The lift and drag coefficient show clear trends in this practice.

The plots of the VSAERO practices, Figures 3.22 and 3.23, show similarly clear trends for the lift and moment coefficient, both including some fluctuations. The powered engines shown in Figure 3.22 are corrected for the produced thrust. The found drag reduction follows from a thrust force generated by the fan cowl.

Overall, all methods are in the same range of effect size per coefficient, except the drag coefficient of the TFN with pylon in FlightStream. This allows to compare the trends as presented in the plots. Clearly, the practices without pylon show better trends than the practice shown here with pylon, as well as in comparison with the result of Proesmans. The practices using a TFN show the highest sensitivity.

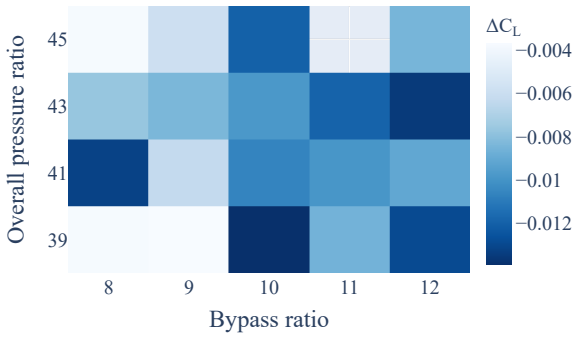
3.3.3. Scoring

Based on the results presented above, the solvers and practices are scored. Each criterion is assigned a score between 1 and 5, with 3 being average, 1 and 2 being below average and 4 and 5 being above average. The given scores are presented in Table 3.13. Using the weights in Table 3.8 and Equations (3.17) and (3.18), the total scores are calculated per practice.

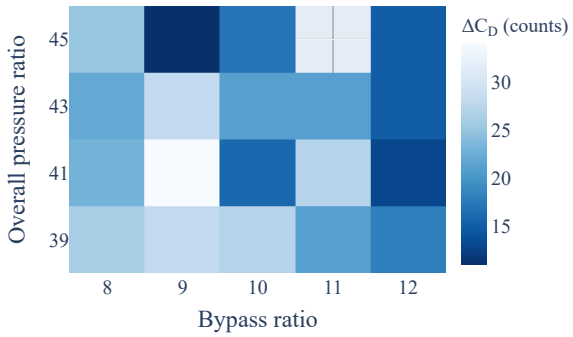
The best practice is a TFN without pylon, simulated in FlightStream. This practice uses Raymer's drag estimation to account for the drag generated by a pylon.

Table 3.13: Analysis practice trade-off scoring.

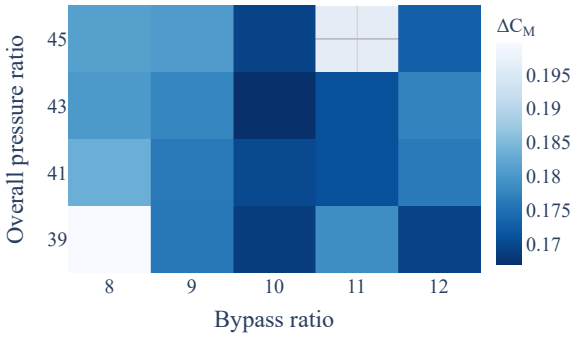
Solver	Practice	Speed	Clean accuracy	Ease-of-use			Installation accuracy	Design sensitivity	Weighted total
				Pre-processing	Interfacing	Post-processing			
FlightStream		4	4	5	5	3			
	TFN with pylon						4	3	19.5
	TFN without pylon						5	5	22.5
VSAERO		3	2	3	3	3			
	Powered with pylon						3	3	14
	Powered without pylon						4	4	16
	TFN without pylon						4	5	17



(a) Lift coefficient.

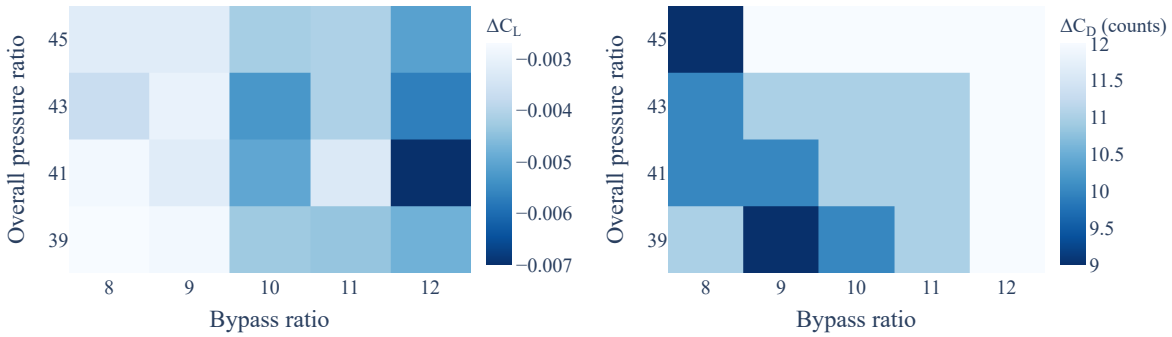


(b) Drag coefficient.



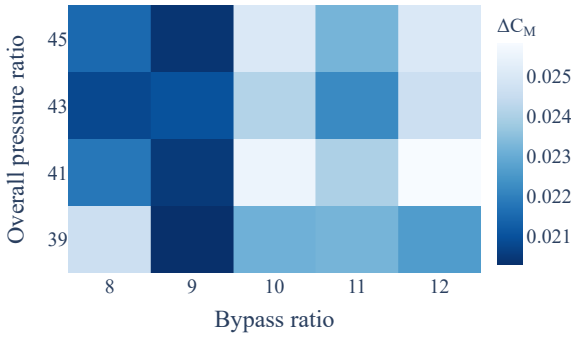
(c) Moment coefficient.

Figure 3.20: Installation effects of FTN type engines with pylon in FlightStream for different engine cycle parameters on the PARSIFAL MS1 aircraft at $M = 0.3$, $Re = 27 \times 10^6$.



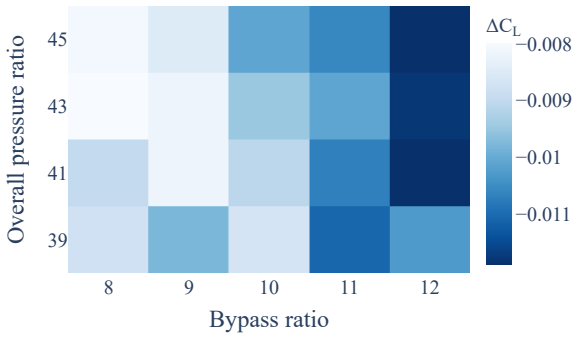
(a) Lift coefficient.

(b) Drag coefficient.

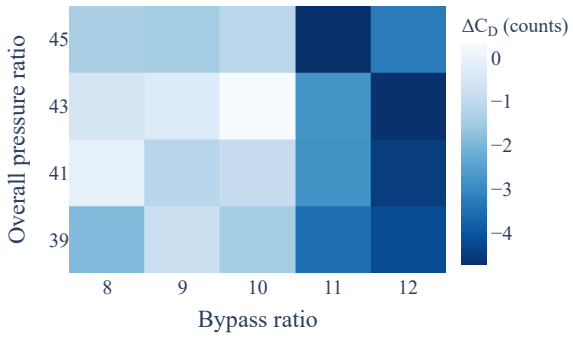


(c) Moment coefficient.

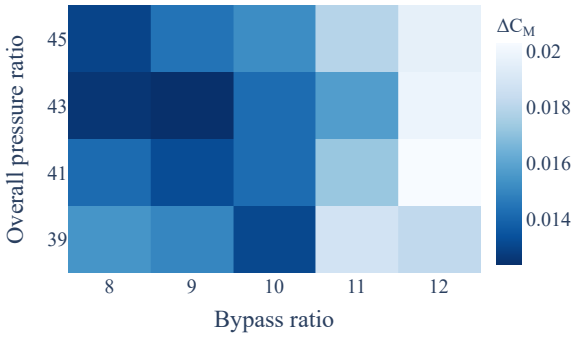
Figure 3.21: Installation effects of FTN type engines without pylon in FlightStream for different engine cycle parameters on the PARSIFAL MS1 aircraft at $M = 0.3$, $Re = 27 \times 10^6$.



(a) Lift coefficient.

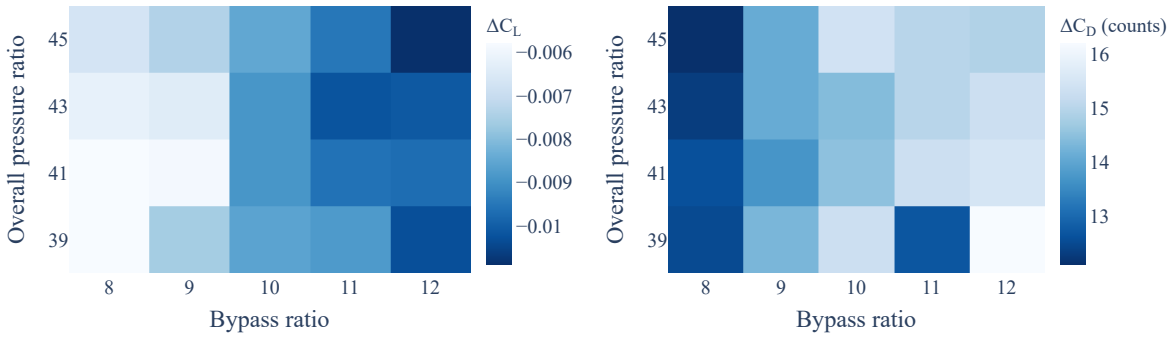


(b) Drag coefficient.



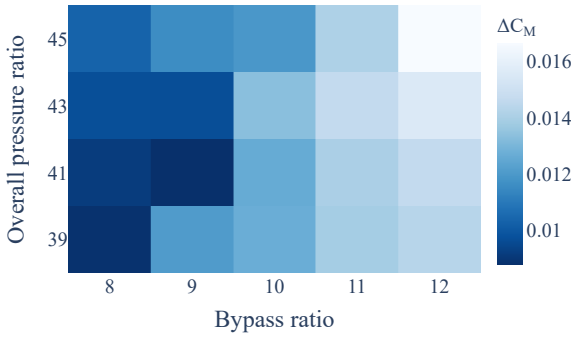
(c) Moment coefficient.

Figure 3.22: Installation effects of powered engines, corrected for thrust, without pylon in VSAERO for different engine cycle parameters on the PARSIFAL MS1 aircraft at $M = 0.3$, $Re = 27 \times 10^6$.



(a) Lift coefficient.

(b) Drag coefficient.



(c) Moment coefficient.

Figure 3.23: Installation effects of TFN type engines without pylon in VSAERO for different engine cycle parameters on the PARSIFAL MS1 aircraft at $M = 0.3$, $Re = 27 \times 10^6$.

4

Parametric study

To evaluate the usability of the best practice found in Section 3.3, two parametric studies are performed. The first study, presented in Section 4.1, aims for a performance gain in the engine installation of the PARSIFAL MS1 by aligning the engine with the local flow. In Section 4.2, an engine installation under the rear wing of the PARSIFAL MS1 aircraft is analysed. This study shows the potential of the best practice for analysing under wing installations.

4.1. Fuselage installation orientation

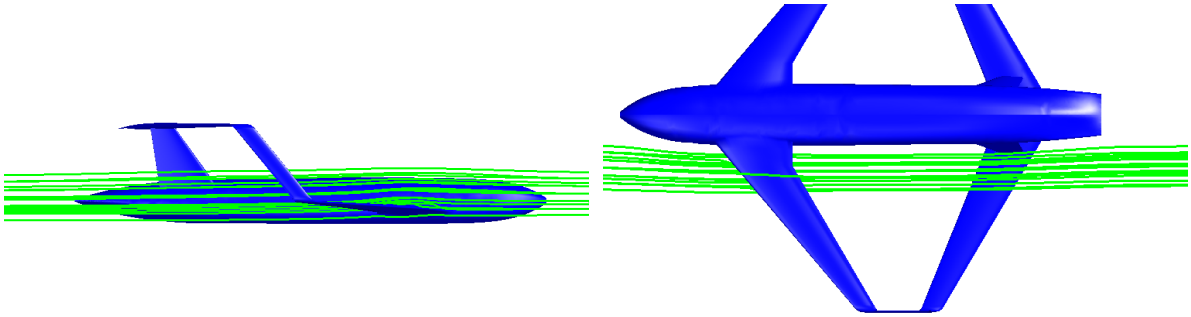
This study looks for an aerodynamic performance gain in the PARSIFAL MS1 engine installation. In the current design, the engines are aligned with the free stream flow at zero angle of attack and zero sideslip angle. However, the engines are located in the downwash of the front wing, which changes the direction of the flow. Moreover, the flow curves inwards, due to the fuselage shape. The local flow, obtained using FlightStream, is shown in Figure 4.1.

To align the engines with the local flow, their pitch and toe angles are changed, while keeping the engine's location fixed in space. This fixation is based on the fact that the current engine location is chosen to align with the connecting structure inside the airframe. The angles' definitions are depicted in Figure 4.2. The minimum and maximum values of the angles are chosen based on the local flowfield of the clean configuration and typical values the textbook of Torenbeek [14]. The pitch angle ranges between $-3°$ and $3°$, while the toe angle is changed between $-3°$ and $0.5°$. Flow conditions are set to $M = 0.30$, $Re = 30 \times 10^6$, as encountered after take-off and during approach. This low Mach number is chosen to ensure no shocks are present on the aircraft. The incoming flow has an angle of attack of $\alpha = 0°$.

The result plotted in Figure 4.3 are on aircraft level, using the engine-less configuration as reference. The lift coefficient in Figure 4.3a is decreased for all engine orientations. Investigation of the lift distribution over the aircraft components shows the nacelle generates negative lift for most orientations. Only for pitch angles larger than $2.25°$, positive lift is generated. The lift generated by the airframe is increased by the presence of the nacelle. The fuselage generates more lift, due to increased suction near the root of the vertical tail. However, this lift increase does not compensate for the negative lift produced by the nacelle. Furthermore, it is expected that the addition of a pylon in this suction area will diminish the added lift. The least lift is lost at an pitch angle of $2.25°$ with a toe angle of $-2°$.

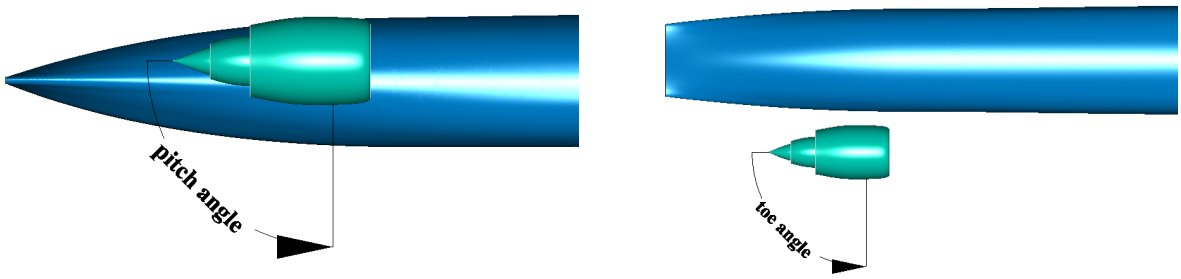
Figure 4.3b shows that the drag increase of installing an engine can be lowered by pitching the engine up and applying an outward toe angle. The drag increase follows from the added drag by the nacelle and pylon, the airframe drag is not changed. To limit the drag increase, the pitch and toe angles found for the minimum lift loss can be used again. This angle combination also shows in Figure 4.3c, it is the combination with the smallest reduction of L/D.

In this optimal position, the value of the aerodynamic efficiency L/D is decreased with $\Delta L/D = -3.4$. A value of $\Delta L/D = -4.1$ is found for the currently used engine orientation. Thus, by aligning the engines with the local flow, a 3% aerodynamic efficiency gain can be obtained. To put these values in perspective, the efficiency of the clean aircraft is $L/D = 29.4$. The results of Proesmans [5] showed



(a) Side view. (b) Bottom view.

Figure 4.1: Local flow streamlines at the engine's position of the PARSIFAL MS1.



(a) Side view. (b) Top view.

Figure 4.2: Angle definition for parametric study of engine installation orientation on the PARSIFAL MS1.

a clean aircraft efficiency of $L/D = 20.8$ and in configuration with engines and pylons $L/D = 15.3$. Differences in both lift and drag between this and Proesmans' study lead to the different efficiencies.

In Figure 4.3d it can be observed that, except for two angle combinations, all orientations provide an additional pitch up moment. The found forces for these angles are neither in line with the result of other angles and thus deemed outliers. The increased pitch up moment follows from the down force generated by the nacelle. The increase in lift on the rear of the fuselage cannot compensate for this down force. With respect to the unaligned engines, a reduction of 17% in the added pitch up moment is obtained by the alignment.

4.2. Under wing positions

To test the use of the best practice with under wing installations, in this study an engine is installed under the rear wing of the PARSIFAL PrandtlPlane. Although this location is already ruled out for the PARSIFAL aircraft, because of the negative effects on aerodynamic performance and structural weight [27], it does provide a good test case for the best practice.

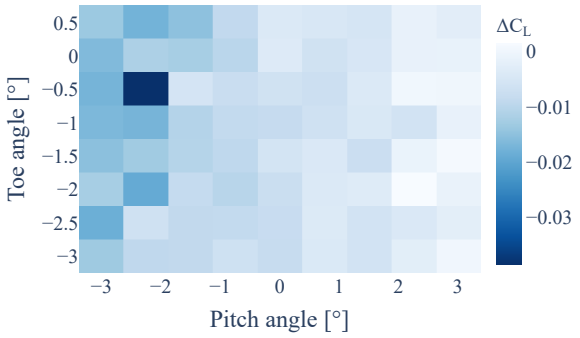
The engine is positioned at a wing section $y = 6$ m, measured from the symmetry plane. At this location the engine will have limited interference with the vertical tail. Within this plane, the engine position is defined by a horizontal distance x/c and vertical distance h/c, both normalized with the local chord. These distances are measured between the leading edge point of the wing and the engine's bypass exhaust's upmost point. The measurements are outlined in Figure 4.4. The engines are aligned with the freestream, so zero pitch and toe angles. The flow conditions are $M = 0.3$, $Re = 30 \times 10^6$, and $\alpha = 0°$.

In Figure 4.5a the change in lift because of the engine installation is shown. It shows that the loss of lift is largest when the engine is horizontally positioned close the the leading edge of the wing. While the nacelle produces a down force, the lift reduction is mostly due to a loss of lift of the rear wing. The presence of the nacelle, changes the flow such that on the bottom side of the wing pressure is lost and on the top side suction is lost. Because the nacelle is close to the wing, on the underside a narrow channel is created, in which flow is accelerated. This acceleration leads to lower surface pressures. A nacelle placed far in front of the wing only lowers the leading edge suction slightly, resulting in a smaller loss of lift.

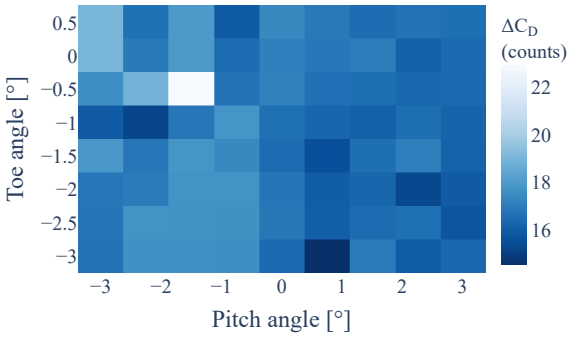
Although the nacelle and pylon add approximately 13 drag counts, Figure 4.5b shows a lower drag increase for the complete aircraft. This reduced drag addition follows from the loss of lift on the rear wing. The lift reduction lowers the induced drag with two to three counts.

From Figure 4.5c, it is observed that the aerodynamic performance is least affected for an installation high and far in front of the wing. This is mostly due to the small loss of lift, with the engine in this position. The loss in lift is also visible in the moment coefficient in Figure 4.5d. The lift loss on the rear wing is reflected in an additional pitch up moment.

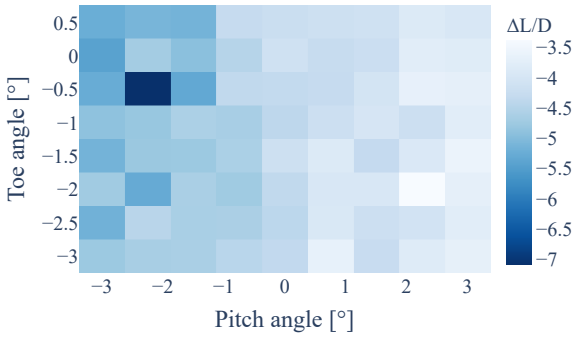
The most efficient position, that is high in front of the wing, shows approximately the same efficiency loss as the fuselage mounted engine in Section 4.1. In the comparison of fuselage and wing mounting by Proesmans [5], the under wing engine was placed further behind the leading edge of the wing. This creates the channel that is detrimental for the aerodynamic efficiency, leading to a efficiency loss of 36%. In the optimal under wing position found by this study, the efficiency loss is only 12%.



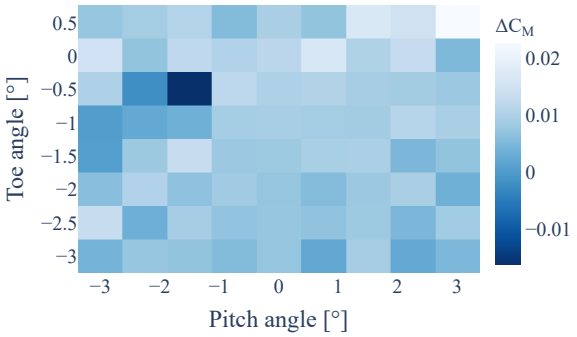
(a) Lift coefficient comparison obtained using FlightStream.



(b) Induced drag coefficient comparison obtained using FlightStream.



(c) L/D comparison obtained using FlightStream and corrected with Torenbeek.



(d) Moment coefficient comparison obtained using FlightStream.

Figure 4.3: Comparison of engine installation effects by changing installation angles with engine-less configuration of the PARSIFAL MS1 aircraft at $M = 0.30$, $Re = 30 \times 10^6$.

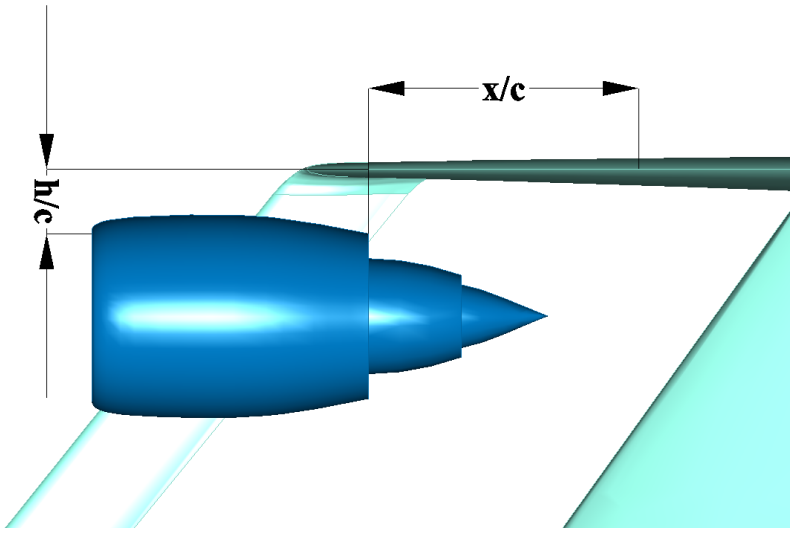
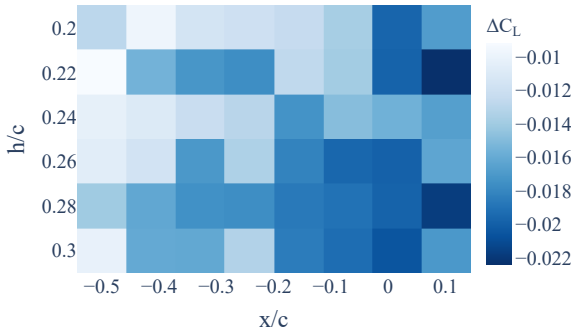
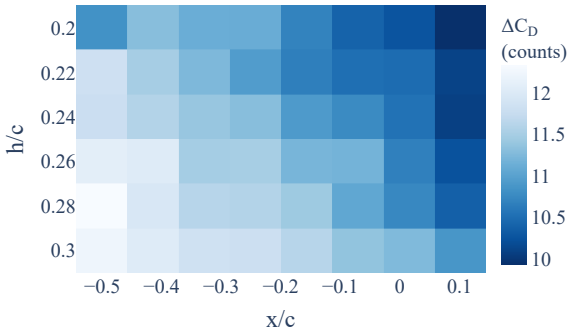


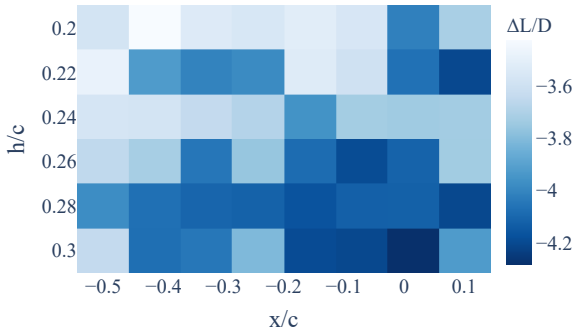
Figure 4.4: Engine location definition.



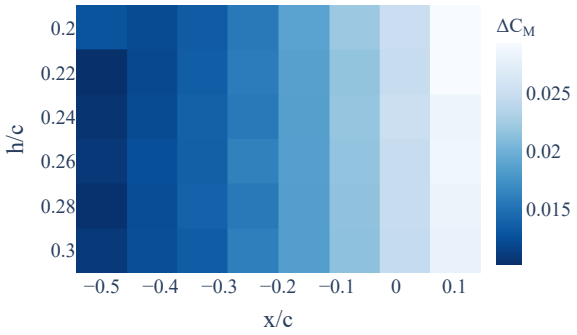
(a) Lift coefficient comparison obtained using FlightStream.



(b) Induced drag coefficient comparison obtained using FlightStream.



(c) L/D comparison obtained using FlightStream and corrected with Torenbeek.



(d) Moment coefficient comparison obtained using FlightStream.

Figure 4.5: Comparison of engine installation effects by changing under wing installation location with engine-less configuration of MS1 aircraft at $M = 0.30$, $Re = 30 \times 10^6$.

5

Conclusions and Recommendations

To finalise this report, conclusions from the presented research are drawn to answer the research questions. Opportunities for further work are added as well.

5.1. Conclusions

By aligning the engines of the PARSIFAL aircraft with the local flow, while keeping their current location fixed, a 3% aerodynamic efficiency gain is obtained with respect to the current orientation. This gain follows from reductions in lift loss and drag addition. Moreover, this alignment reduces the added pitching moment with 17%.

These results are obtained using the panel solver FlightStream. The engine installation is analysed by simulating the airframe with through flow nacelles (TFN) and adding pylons through a textbook correction by Raymer. This analysis practice showed to be best in a trade-off study that used speed, accuracy of a clean aircraft simulation, ease-of-use, accuracy of engine installation effects prediction, and design sensitivity as scoring criteria.

In this trade-off study, analysis practices to find the effects on lift, drag, and pitching moment by an engine installation are compared. Two textbook methods are considered, those by Raymer and Torenbeek. However, these are not satisfactory on their own, since they only provide information on drag. Therefore, also panel methods are considered. Two solvers are tested, FlightStream and VSAERO. Practices of simulating airframe, nacelle, and pylon and simulating pylon and nacelle, corrected with pylon drag from a textbook method are compared. The Raymer textbook method is used for this correction, since it showed to be a better drag predictor than the Torenbeek method.

Although the focus of the research was on PARSIFAL's fuselage mounted engines, it is shown that with the found best practice also under wing installations can be analysed. Effects on lift, drag, and pitching moment can be predicted and show design sensitivity for different locations of the engine with respect to the leading edge of the wing. Moreover, the results show that with proper engine positioning the same aerodynamic efficiency as with a fuselage mounting can be obtained.

Before the trade-off study could be performed, changes to the MMG had to be made. Three primitives, the building blocks that create an aircraft geometry, had to be adapted. Firstly, the current wing primitive DARwing is replaced with a new one, ParaWing. This was required because of incompatibility of the old primitive with a new version of the underlying platform of the MMG. The new primitive uses common geometrical characteristics, e.g. dihedral angle, sweep angle, and chord length, to parametrise a wing. Furthermore, the new primitive can create n^{th}-order rails, defining the planform of the wing, that are not limited by dihedral angle. Comparison with the results of the old primitive shows differences of less than 1% in leading and trailing edge locations. However, direct control over the stagger of box-wings is lost if they are modelled as a single wing.

Secondly, to enable proper simulation of TFNs, the engine primitive is extended with this type of nacelle. In previous research, a TFN was simulated by setting the velocity of the fan and exhaust to values following from the free stream. However, because of the area ratio between fan and exhaust faces, this leads to a simulation that does not stroke with physics. By adding a geometry that models

the external shape of the nacelle, but has an open stream tube instead of a full engine geometry, TFNs can be modelled properly.

Thirdly, the design space of the pylon is extended by the implementation of a new primitive. This primitive provides the user with more options to shape a pylon. Pylons with leading edges close to perpendicular to the flow, e.g. fuselage mounted pylons, can be formed correctly. However, pylons with highly swept leading edges, often found in under wing installations, are a worse match with reality. Their planform can be shaped correctly, but the created 3d shape does not feature the smooth curvature found on real pylons.

5.2. Recommendations

Although the trade-off in this research provides a best practice for engine integration analyses, the selection of practices is limited by a bug in the used version of FlightStream. This bug prevented the modelling of powered engines, so only TFN can be used. Once this issue is resolved, powered engines should be investigated and scored using the same criteria as in this research, to find how they compare to the TFN in the found best practice. Furthermore, since FlightStream is still in active development, new features might be added that increases its usability, so an eye should be kept on this development.

In order to replace DARwing with ParaWing completely, further work is required. To better control the stagger of box-wings, the construction method of the box-wing should be changed. One option is to implement a utility that takes the wing stagger and desired sweep and dihedral as inputs. The utility should then find a dihedral and sweep distribution to form a box-wing. This allows the box-wing to be modelled as one shape. A second option is to separate the box-wing in three parts. The two horizontal wings can then be placed as desired. A connecting element should be created to close the box-wing system. If the latter option is chosen, primitives in the MMG can be reused by updating them.

To enable a flight mechanics analysis, movable control surfaces should be added to ParaWing. To use the multi-element capabilities of ParaWing in the MMG, the MMG should take inputs that define these multi-element sections and pass them down to ParaWing.

The pylon primitive should be refined as well. To create a realistic fairing, the placement of sections has to be adapted to realistically shape highly swept pylons. This is probably best done by turning the orientation of the sections, so their normals are tangent with the flow, instead of perpendicular as they are now. This, however, also requires a redefinition of the rails that form the planform of the pylon. A better fairing geometry will return a more accurate estimation of the wetted area. This will improve the results of the pylon drag estimation.

There is also an opportunity to include the internal structure of the pylon. When this is first sized based on the forces it should transfer from the engine into the airframe, a fairing can be created around it. This improves the sizing of the fairing, and will thus lead to a better drag prediction.

In this thesis, it is shown that with the found best practice meaningful parametric studies can be performed to find the effect of engine installations on aerodynamic performance. This practice can serve as a base for a more extensive engine integration analysis, in which also the engine's efficiency and flight mechanics analyses are included. The combination of these analyses allows to make a well informed engine selection and decision on installation location, in an early design stage.

Bibliography

[1] M. Darecki, C. Edelstenne, T. Enders, E. Fernandez, P. Hartman, J.-P. Herteman, M. Kerkloh, I. King, P. Ky, M. Mathieu, G. Orsi, G. Schotman, C. Smith, and J.-D. Wörner, *Flightpath 2050 Europe's Vision for Aviation* (2011).

[2] L. Prandtl, *Induced drag of multiplanes,* (1924).

[3] A. Frediani, L. B. Crema, G. Chiocchia, G. L. Ghiringhelli, and L. Morino, *Development of an Innovative Configuration for Transport Aircraft,* XVII Congresso Nazionale AIDAA (2003).

[4] A. Frediani, *Prandtlplane Architecture for the Sustainable Improvement of Future AirpLanes (PARSIFAL),* (2016).

[5] P. Proesmans, *Preliminary Propulsion System Design and Integration for a Box-Wing Aircraft Configuration,* (2019).

[6] G. La Rocca and M. J. L. van Tooren, *Description of the ICAD Multi-model generator interaction tools* (TU Delft, Faculteit Luchtvaart- en Ruimtevaarttechniek, Delft, 2002).

[7] J. Koning, *Development of a KBE application to support aerodynamic design and analysis: Towards a next-generation multi-model generator,* (2010).

[8] J. Wei, *Parametric modelling for determining aircraft stability & control derivatives,* (2016).

[9] R. Groot, *Stability & control derivatives prediction for box wing aircraft configuration,* (2019).

[10] A. Sóbester and A. I. Forrester, *Aircraft Aerodynamic Design: Geometry and Optimization*, Aerospace Series (Wiley, Chichester, England, 2014) pp. 1–246.

[11] B. Malouin, M. Gariepy, J. Y. Trepanier, and E. Laurendeau, *Internal drag evaluation for a through-flow nacelle using a far-field approach,* Journal of Aircraft **52**, 1847 (2015).

[12] L. King, V. Ahuja, and R. Hartfield, *Aerodynamic optimization of integrated wing-engine geometry using an unstructured vorticity solver,* in *33rd AIAA Applied Aerodynamics Conference* (2015).

[13] V. Ahuja, J. Burkhalter, and R. J. Hartfield, *Optimizing engine placement on an aircraft wing using bio-mimetic optimization and flightstream™,* in *AIAA SciTech Forum - 55th AIAA Aerospace Sciences Meeting* (2017).

[14] E. Torenbeek, *Synthesis of Subsonic Airplane Design* (1982).

[15] D. Raymer, *Aircraft Design: A Conceptual Approach, Sixth Edition*, 4th ed. (American Institute of Aeronautics and Astronautics, 2018).

[16] T. P. Stákowski, D. G. MacManus, C. T. Sheaf, and R. Christie, *Aerodynamics of aero-engine installation,* Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering **230**, 2673 (2016).

[17] J. C. Vassberg, M. A. DeHaan, S. M. Rivers, and R. A. Wahls, *Development of a common research model for applied CFD validation studies,* in *Collection of Technical Papers - AIAA Applied Aerodynamics Conference* (2008).

[18] V. Ahuja and R. J. Hartfield, *Aerodynamic loads over arbitrary bodies by method of integrated circulation,* Journal of Aircraft **53**, 1719 (2016).

[19] J. D. Anderson, *Fundamentals of aerodynamics.* (2011).

[20] H. S. Tsien, *Two-Dimensional Subsonic Flow of Compressible Fluids,* Collected Works of Hsue-Shen Tsien (1938-1956) **6**, 92 (2012).

[21] E. V. Laitone, *New Compressibility Correction for Two-Dimensional Subsonic Flow,* Journal of the Aeronautical Sciences **18**, 350 (1951).

[22] M. B. Rivers and A. Dittberner, *Experimental investigations of the NASA common research model,* in *Journal of Aircraft*, Vol. 51 (2014) pp. 1183–1193.

[23] M. Carini, M. Méheut, L. Sanders, S. Kanellopoulos, C. Varriale, G. La Rocca, K. Abu Salem, V. Cipolla, and V. Binante, *Aerodynamic and Acoustic Analysis of the baseline PrandtlPlane*, Tech. Rep. 723149 (2017).

[24] M. B. Rivers, C. A. Hunter, and R. L. Campbell, *Further investigation of the support system effects and wing twist on the NASA common research model,* 30th AIAA Applied Aerodynamics Conference 2012 , 1960 (2012).

[25] J. K. Nathman, *VSAERO Version 7.9,* (2016).

[26] A. R. Kulkarni, C. Varriale, M. Voskuijl, G. La Rocca, and L. L. Veldhuis, *Assessment of sub-scale designs for scaled flight testing,* in *AIAA Aviation 2019 Forum* (American Institute of Aeronautics and Astronautics (AIAA), 2019) pp. 1–21.

[27] R. Elmendorp, *Feasibility study on the use of very large bypass ratio turbofan engines for the PrandtlPlane*, Tech. Rep. 723149 (2020).

A

ParaWing input

Listing A.1: ParaWing input file for MS1 box-wing and vertical tail.

```
{
  "boxwing": {
    "is_closed": true,
    "is_mirrored": true,
    "flatten_root": true,
    "sections": [
      {
        "type": "selig",
        "filename": "./resources/profile_wing_lower_00_id.dat",
        "eta": 0.0,
        "flip_airfoil": false
      },
      {
        "type": "selig",
        "filename": "./resources/profile_wing_lower_01_id.dat",
        "eta": 0.029556,
        "flip_airfoil": false
      },
      {
        "type": "selig",
        "filename": "./resources/profile_wing_lower_02_id.dat",
        "eta": 0.103259,
        "flip_airfoil": false
      },
      {
        "type": "selig",
        "filename": "./resources/profile_wing_lower_03_id.dat",
        "eta": 0.424697,
        "flip_airfoil": false
      },
      {
        "type": "selig",
        "filename": "./resources/profile_wing_lower_03_id.dat",
        "eta": 0.44415,
        "flip_airfoil": false
      },
      {
        "type": "selig",
        "filename": "./resources/profile_wing_upper_01_id.dat",
        "eta": 0.605703,
        "flip_airfoil": true
```

```
      },
      {
        "type": "selig",
        "filename": "./resources/profile_wing_upper_01_id.dat",
        "eta": 0.62179,
        "flip_airfoil": true
      },
      {
        "type": "selig",
        "filename": "./resources/profile_wing_upper_00_id.dat",
        "eta": 1.0,
        "flip_airfoil": true
      }
    ],
    "position": {
      "point": [
        7.95857063,
        0.0,
        -1.00055185
      ]
    },
    "rails": {
      "span": 18.0,
      "chord_lengths": [
        8.0,
        8.0,
        5.35,
        1.95,
        1.9,
        1.858,
        1.823,
        1.795,
        1.779,
        1.892,
        1.903,
        1.919,
        1.939,
        1.963,
        1.99,
        5.3
      ],
      "chord_etas": [
        0.0,
        0.029556,
        0.103259,
        0.424697,
        0.428588,
        0.432479,
        0.436369,
        0.44026,
        0.44415,
        0.605703,
        0.60892,
        0.612137,
        0.615355,
        0.618572,
        0.62179,
        1.0
      ],
      "dihedral_angles": [
        0.0,
```

```
        0.0,
        4.96,
        4.96,
        2.72,
        2.72,
        13.2493,
        28.1456,
        48.3358,
        70.8141,
        90.0,
        90.0,
        107.7807,
        131.3811,
        153.923,
        169.9102,
        180.0,
        180.0
      ],
      "dihedral_etas": [
        0.0,
        0.029555,
        0.029557,
        0.103258,
        0.10326,
        0.424697,
        0.428588,
        0.432479,
        0.436369,
        0.44026,
        0.44415,
        0.604013,
        0.60892,
        0.612137,
        0.615355,
        0.618572,
        0.62179,
        1.0
      ],
      "sweep_angles": [
        0.0,
        0.0,
        32.05,
        32.05,
        38.015,
        38.015,
        39.7778,
        42.5038,
        44.2068,
        42.8635,
        38.866,
        38.866,
        43.127,
        43.5066,
        38.2214,
        29.9828,
        24.225,
        24.225
      ],
      "sweep_etas": [
        0.0,
        0.029555,
```

```
        0.029557,
        0.103258,
        0.10326,
        0.424697,
        0.432479,
        0.436369,
        0.44026,
        0.44415,
        0.444305,
        0.604013,
        0.60892,
        0.612137,
        0.615355,
        0.618572,
        0.62179,
        1.0
      ],
      "twist_angles": [
        2.97,
        2.97,
        3.87,
        1.47,
        0.0,
        0.0,
        1.37,
        3.67
      ],
      "twist_etas": [
        0.0,
        0.029556,
        0.103259,
        0.424697,
        0.444305,
        0.604013,
        0.62179,
        1.0
      ],
      "curve_distribution_points": [
        [
          0,
          0.42,
          50
        ],
        [
          0.42,
          0.45,
          50
        ],
        [
          0.45,
          0.58,
          10
        ],
        [
          0.58,
          0.625,
          50
        ],
        [
          0.625,
          1,
```

```
          20
        ]
      ]
    }
  },
  "verticalTail": {
    "is_mirrored": true,
    "parameters": [
      {
        "eta": 0.0,
        "type": "selig",
        "filename": "./resources/profile_vtp_00_id.dat"
      },
      {
        "eta": 1.0,
        "type": "selig",
        "filename": "./resources/profile_vtp_01_id.dat"
      }
    ],
    "position": {
      "point": [
        30.67704797,
        1.881330114,
        1.045121184
      ],
      "orientation": {
        "x_vector": [
          1.0,
          0,
          0.0
        ],
        "y_vector": [
          0,
          0.25881971718949,
          0.9659256462036571
        ]
      }
    },
    "rails": {
      "span": 5.989732512786134,
      "dihedral_angles": [
        0.0,
        0.0
      ],
      "twist_angles": [
        0.0,
        0.0
      ],
      "sweep_etas": [
        0.0,
        1.0
      ],
      "dihedral_etas": [
        0.0,
        1.0
      ],
      "sweep_angles": [
        29.14769325485116,
        29.14769325485116
      ],
      "chord_lengths": [
```

```
        5.552589999999999,
        2.2819240000000036
      ],
      "chord_placement_factor": 0.0,
      "chord_etas": [
        0.0,
        1.0
      ],
      "twist_axis_factor": [
        0.0,
        0.0
      ]
    }
  }
}
```

B

Pylon input

Listing B.1: Inputs for fuselage mounted pylon.

```
{
  "parentUID": "fuselage",
  "root_airfoil": "./airfoil/NACA0012.dat",
  "tip_airfoil": "./airfoil/NACA0012.dat",
  "fuselage_front_section": 33.6,
  "fuselage_aft_section": 36.1,
  "fuselage_vertical_offset": 0.1,
  "le_longitudinal_offset": [
    -1.5,
    -1.0
  ],
  "te_longitudinal_offset": [
    2.5,
    0.6
  ]
}
```

Listing B.2: Inputs for wing mounted pylon.

```
{
  "parentUID": "NASA_CRM_wing1",
  "root_airfoil": "NACA0012",
  "tip_airfoil": "NACA0012",
  "le_longitudinal_offset": [
    -3.0,
    -1.0
  ],
  "te_longitudinal_offset": [
    1.8,
    2.0
  ],
  "wing_front_spar_location": 0.2,
  "wing_aft_spar_location": 0.8,
  "wing_vertical_offset_ratio": 0.03,
  "additional_sections": [
    {
      "span_fraction": 0.25555722377970685,
      "airfoil": "NACA0012",
      "position": "absolute",
      "le_position": [26.32315,
      9.6,
      4.116549079647776]
    },
    {
      "span_fraction": 0.36968730498541913,
      "airfoil": "NACA0012",
      "position": "absolute",
      "le_position": [26.13315,
      9.6,
      3.916549079647776]
    }
  ]
}
```

C

Mesh input

Listing C.1: Inputs FlightStream mesh of PARSIFAL aircraft.

```
{
  "use_half_mesh": true,
  "wings": {
    "boxwing": {
      "number_chordwise_points": 60,
      "spanwise_pitch": 1,
      "tri_max_size": 0.50,
      "tri_min_size": 0.10,
      "tri_growth_rate": 0.30,
      "tip_max_size": 1.00,
      "tip_min_size": 0.30,
      "tip_growth_rate": 1,
      "chordwise_distribution": "geometric",
      "chordwise_growth_rate": 1.1,
      "spanwise_refinements": [
        {
          "le_index": 2,
          "pitch": 0.2
        },
        {
          "le_index": 4,
          "pitch": 0.2
        }
      ],
      "mesh_groups": {"front": [0, 1], "connector": [2, 3, 4], "rear": [5, 6, 7]}
    },
    "verticalTail": {
      "number_chordwise_points": 60,
      "spanwise_pitch": 0.6,
      "tri_max_size": 0.50,
      "tri_min_size": 0.10,
      "tri_growth_rate": 0.20,
      "tip_max_size": 1.00,
      "tip_min_size": 0.30,
      "tip_growth_rate": 1,
      "chordwise_distribution": "geometric",
      "chordwise_growth_rate": 1.1
    },
    "engine_pylon": {
      "number_chordwise_points": 80,
      "spanwise_pitch": 0.1,
```

```
      "tri_max_size": 0.30,
      "tri_min_size": 0.05,
      "tri_growth_rate": 0.10,
      "tip_max_size": 1.00,
      "tip_min_size": 0.30,
      "tip_growth_rate": 1,
      "chordwise_distribution": "geometric",
      "chordwise_growth_rate": 1.1
    }
  },
  "fuselage": {
    "fuselage_lateral_control": 40,
    "lateral_max_size": 0.6,
    "lateral_min_size": 0.3,
    "lateral_growth_rate": 0.25,
    "lateral_nb_seg_per_radius": 3,
    "tail_max_size": 0.25,
    "tail_min_size": 0.05,
    "tail_growth_rate": 0.20,
    "nose_max_size": 0.25,
    "nose_min_size": 0.05,
    "nose_growth_rate": 0.2,
    "nose_longitudinal_control": 0.5,
    "tail_refinement": {
      "offset_factor": 0.1,
      "growth_rate": 0.2,
      "max_size": 0.4,
      "min_size": 0.1,
      "nb_seg_per_radius": 5
    }
  },
  "turbofans": {
    "Turbofan_0": {
      "number_circumferential_points": 15,
      "longitudinal_pitch": 0.2,
      "max_size": 0.4,
      "min_size": 0.1
    }
  }
}
```

D

Engine geometrical data

Table D.1: Nacelle geometry parameters used in textbook analysis.

Variable	Value
M_∞	0.83
$Re_{\bar{c}}$	5×10^6
$\bar{c}$	7.005
β	0.40
S_{wet_f}	67.89 m^2
$\left(C_{D_p}\right)_{\beta_f}$	0.0176
d_n	3.93 m
d_h	3.32 m
l_n	5.74 m
d_o	6.69 m
M_f	0.87
l_g	1.01 m
S_{wet_g}	5.66 m^2
$\left(C_{D_p}\right)_{\beta_g}$	0.0160
d_g	1.89 m
M_g	0.91
l_p	1.43 m
S_{wet_p}	2.67 m^2
S_{ref}	383.3m^2
k	4.05×10^{-6} m
Q_c	1.3

Table D.2: Pylon geometry parameters used in textbook analysis.

Variable	Value
M_∞	0.83
$Re_{\bar{c}}$	5×10^6
t/c	0.06
$\Lambda_{0.25c}$	81°
$S_{wet_{py}}$	16.8
S_{ref}	383.3 m^2
$(x/c)_m$	0.5
Λ_m	78°
k	4.05×10^{-6} m
Q_c	1.5

E

Flow conditions calculations

Table E.1: Standard atmosphere for used altitudes [19].

Altitude [m]	Temperature [K]	Pressure [Pa]	Density [kg m^{-3}]
0	288.16	$1.013\,25 \times 10^5$	1.2250
11,000	216.78	2.2700×10^4	0.3648

The following equations are taken from [19]

$$a = \sqrt{\gamma R T} \tag{E.1}$$

$$\mu = \left(\frac{T}{T_0}\right)^{3/2} \frac{T_0 + 110}{T + 110} \mu_0 \tag{E.2}$$

with $T_0 = 288.16$ K and $\mu_0 = 1.7894 \times 10^{-5}$ kg m s^{-1}

$$\rho = \frac{P}{TR} \tag{E.3}$$