A Knowledge-based Engineering Application for Wing–Nacelle–Intake Integrations in Fuel Cell Aircraft

Erik Regeling

Delft University of Technology

TU Delft

A Knowledge-based Engineering Application for Wing–Nacelle–Intake Integrations in Fuel Cell Aircraft

by

Erik Regeling

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday 22nd of January at 14:00.

Student number:	4864794	
Project duration:	September, 2024 – January, 2026	
Principle supervisor:	Prof. dr. ir. L.L.M. Veldhuis	
Additional supervisor:	Dr. ir. G. la Rocca	
Thesis committee:	Dr. ir. T. Sinnige	TU Delft, chair
	Prof. dr. ir. L.L.M. Veldhuis	TU Delft, principle supervisor
	Dr. ir. G. la Rocca	TU Delft, additional supervisor
	Prof. dr. M. Kotsonis	TU Delft, external examiner

Cover: ”Universal Hydrogen first flight” by F. Zera, used with written permission. [1]

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

This work marks the end of my student career at TU Delft, and with it the closing of an important chapter in my life. It has been a long and demanding journey, filled with ups and downs, challenges, long hours, and the freedom to fully apply myself and discover who I am professionally. During this time, I have learned an immense amount, technically, socially, and emotionally, for which I am incredibly grateful. Although this journey concludes with a diploma that reflects my technical abilities, there is far more that I have gained than what can be captured on a single piece of paper.

Throughout my studies, I learned to always give 100% of myself to every project I worked on, and this thesis is no exception. In fact, it may be the project I am most proud of. As I write these final words at the end of this thesis, I feel a strong sense of pride in what I have achieved. I fully stand behind every sentence, discussion, and motivation presented in this work.

That being said, I genuinely believe that sustainable flight is possible in the future. The idea of fully green aviation is a dream come true, not only because of the remarkable technological advancements it requires, but also because of its potential impact on our world. We live on a planet that we exploit every day, and the prospect of contributing, even in a small way, to a more sustainable future is deeply motivating. For this reason, I have put the fullest of my effort into this project, hoping that one day the work done here will matter.

However, I was not alone on this journey. First, I would like to thank Gianfranco La Rocca, who inspired my interest in Knowledge-Based Engineering and has likely shaped my future career path. I would also like to thank Leo Veldhuis for introducing me to this project. It was only fitting that both Leo and Gianfranco served as my supervisors throughout this thesis. They provided the proper guidance and support and comfortably guided me through this project, ultimately contributing greatly to its success. Every meeting with them was enjoyable, filled with enthusiasm and shared interest in the work.

I would also like to thank the people at ParaPy B.V. for their continuous support. In particular, I thank Reinier van Dijk and Max Baan for providing me with a workplace, proper resources, and valuable input on the content of this work, and for welcoming me with open arms during my time there. They made me feel truly part of the team, even after I stopped my part-time work to fully focus on this thesis. Special thanks go to San Kilkiş, who helped me through many technical challenges during the development of my tool and always took time out of his personal schedule whenever I needed assistance.

Finally, and most importantly, I would like to thank my family and friends. I am deeply grateful to my parents, Peter and Jacqueline, for giving me every opportunity to pursue my ambitions without hesitation, and for supporting me with complete trust and freedom to do what I needed and wanted, not only during this project, but throughout my entire personal and professional journey. They shaped me into the person I am today. I would also like to thank my significant other, Bo, for always supporting me. You were the one I could lean on whenever I needed it most

Thank you.

Erik Regeling
Delft, January 2026

Executive Summary

This thesis presents the development and assessment of WIN-Gen (Wing–Intake–Nacelle Generator), a knowledge-based engineering tool for the parametric generation and evaluation of outer mould line (OML) geometries for wing-nacelle-intake (WNI) configurations of fuel cell-powered and electrified aircraft. The work addresses the growing need for systematic propulsion integration studies enabled by the design freedom inherent to fully electrified propulsion systems.

WIN-Gen enables the automated generation of a wide range of WNI configurations and supports early-stage assessment of geometric feasibility and indication of aerodynamic behaviour. Using a Model-Based Systems Engineering (MBSE) approach, stakeholder needs identified from literature are translated into a structured and verifiable set of system requirements, which guide the tool architecture and functionality.

A key methodological contribution of this thesis is the introduction of the *parametrisation-as-composition* paradigm. In this approach, the parametrisation of a component is explicitly decoupled from its geometric definition, allowing multiple parametrisation strategies to be interchanged without modifying the underlying component model. Used extensively throughout this work, this significantly improves modularity, reusability, and extensibility of the design framework and enables systematic expansion of the tool to new parametrisations, components and integration strategies.

In addition, this work introduces a novel parametric blending strategy, the `BlendSolid` algorithm, to address the challenge of smoothly integrating complex geometric components. The algorithm enables quasi-tangent blending between low- to medium-complexity solids while maintaining intuitive and fully parametric control over the intermediate blending surface. This capability is essential for the robust integration of wings, nacelles, and intakes and forms a critical enabler for automated generation of integrated WNI geometries.

The tool provides fully parametric modelling of the wing, nacelle, and scoop intake and allows independent component design and analysis prior to integration. Through extensive geometric verification, WIN-Gen demonstrates robust behaviour during manual design, where configurations can be interactively corrected. In automatic design generation, a parametric robustness of approximately 30–35% is achieved across a large and realistic design space, while still enabling the generation of a diverse set of distinct and valid geometries. The primary sources of failure are identified as limitations in Gordon surface divergence, geometric blending limitations, and implemented modelling methodology.

A geometric validation study reproducing a reference wing-nacelle configuration (TUD-PWF) shows excellent agreement, with an geometrical distance RMS deviation of 0.13% and a maximum deviation of 0.49%. Preliminary aerodynamic validation indicates promising agreement in lift characteristics and local pressure distributions. However, the fidelity of automatically generated aerodynamic results is currently limited by surface meshing quality which is out of scope in this work, indicating the need for further study.

Overall, WIN-Gen constitutes a comprehensive and extendable parametric framework for wing-nacelle-intake configurations, capable of generating a wide range of OML designs for fuel cell and electrified propulsion concepts. Nevertheless, limitations remain in the modelling of fully blended and highly integrated WNI concepts, the maturity of the aerodynamic analysis framework for propulsion integration, and the inclusion of innovative intake configurations. Owing to its inherently extensible architecture, targeted future work is expected to address these limitations. Recommended developments include improved geometric algorithms, a comprehensive meshing and aerodynamic simulation study, an expanded design validation effort, and further enhancement and extension of component-level modelling capabilities.

Contents

Preface i

Executive Summary ii

List of Figures vi

List of Tables xi

Nomenclature xii

1 Introduction 1
- 1.1 Integration of Fuel Cell Thermal Management Systems in Aircraft . . . 2
- 1.2 Unconventional Intake Configurations . . . 3
- 1.3 Propeller and Nacelle Integration . . . 3
- 1.4 Research topic . . . 3
  - 1.4.1 Scope of the Research . . . 4
  - 1.4.2 Research objective . . . 4
  - 1.4.3 Hypotheses . . . 4
  - 1.4.4 Research questions . . . 5
  - 1.4.5 Research Structure . . . 6

2 Literature Review: Key Findings and Research Scope 8
- 2.1 Hydrogen Fuel Cell Technology in Aircraft . . . 8
  - 2.1.1 Scope of Fuel Cell Sizing and Thermal Management in the Model . . . 9
- 2.2 Intakes . . . 9
  - 2.2.1 Scope of Intakes in the Model . . . 11
- 2.3 Propeller and Nacelle Integration . . . 11
  - 2.3.1 Scope of Propeller and Nacelle Integration in the Model . . . 12

3 Methodology and Requirements 13
- 3.1 Parametric Modelling Methodology . . . 13
  - 3.1.1 Knowledge-Based Engineering . . . 13
  - 3.1.2 KBE and augmented CAD . . . 13
  - 3.1.3 Tools and Software . . . 14
  - 3.1.4 Selection of Modelling Methodology and Tool . . . 14
- 3.2 MBSE-for-KBE . . . 15
  - 3.2.1 Knowledge Model Ontology . . . 15
  - 3.2.2 Scope of MBSE-for-KBE in the Tool . . . 16
- 3.3 Requirements Package . . . 16

4 Parametric Model 18
- 4.1 Model Architecture . . . 18
- 4.2 Parametrisation as Composition . . . 20
  - 4.2.1 Descriptors and Construction . . . 22
- 4.3 Capability Modules . . . 23
  - 4.3.1 Data Import . . . 23
  - 4.3.2 Geometry Export Module . . . 24
  - 4.3.3 Automatic Meshing . . . 26
  - 4.3.4 Simulation Interface and Aerodynamics Module . . . 27
- 4.4 Gordon Surface . . . 31
  - 4.4.1 Previous Work and Rationale . . . 31
  - 4.4.2 The Algorithm . . . 32

4.5 Blending of Solids Algorithm . . . 32
4.5.1 The Algorithm . . . 33
4.5.2 Blending Curve . . . 33
4.5.3 Shape Control . . . 36
4.5.4 Continuity Assessment . . . 38
4.5.5 Conclusion and Limitations . . . 41

5 WIng Package Modelling 43
5.1 Architectural Overview . . . 43
5.2 Profiles and Airfoils . . . 45
5.2.1 Profiles and Cross-sections . . . 45
5.2.2 Airfoils . . . 46
5.2.3 Curve for Profiles and Airfoils . . . 47
5.2.4 Imported Airfoils . . . 48
5.2.5 Airfoil on Rails and Transform Parametrisations . . . 49
5.3 Lifting Surface . . . 51
5.3.1 Wings Skeleton and Planform . . . 51
5.3.2 Lifting Surface . . . 52
5.3.3 Clean Wing . . . 53
5.4 Notable Improvements from Previous Work . . . 55
5.5 Matched Lifting Surface . . . 55
5.5.1 Algorithm . . . 55
5.5.2 Inserted Airfoil . . . 57
5.5.3 Limitations and Recommendations . . . 57
5.6 Capability Module Implementations . . . 58

6 Nacelle 59
6.1 Architectural Overview . . . 59
6.2 Spinner . . . 60
6.2.1 Shape Parametrisations . . . 61
6.2.2 Positioning . . . 61
6.3 Internal Subsystems . . . 61
6.3.1 Sizing . . . 62
6.3.2 Positioning of Internals . . . 62
6.4 Nacelle Cross-sections . . . 63
6.4.1 Curves for Nacelle Cross-sections . . . 63
6.4.2 Imported Cross-sections . . . 64
6.4.3 Nacelle Cross-sections Positioning . . . 65
6.5 Nacelle . . . 66
6.5.1 Concrete Nacelle . . . 67
6.5.2 Nacelle from Cross-sections . . . 67
6.6 Matched Nacelle . . . 68
6.6.1 Algorithm . . . 68
6.6.2 Inserted Cross-section . . . 70
6.6.3 Limitations and Recommendations . . . 70
6.7 Wing-Nacelle Integration . . . 71
6.7.1 Nacelle Positioning . . . 71
6.7.2 Wing-Nacelle Geometry Matching . . . 72
6.7.3 Wing-Nacelle Geometrical Integration . . . 73
6.7.4 Limitations and Recommendations . . . 75
6.8 Capability Module Implementations . . . 76

7 Scoop Intake 77
7.1 Architectural Overview . . . 77
7.2 Lips . . . 78
7.2.1 Clean Lip and Lip Curve Parametrisations . . . 79
7.3 2D Scoop and Intakes . . . 80
7.3.1 Scoop Profile . . . 80

7.3.2 Clean Scoop Profile . . . 80
7.4 3D Scoop Intake . . . 81
7.4.1 Direct Scoop . . . 81
7.5 Nacelle-Scoop Integration . . . 83
7.5.1 Algorithm . . . 83
7.5.2 Limitations and Recommendations . . . 85
7.6 Capability Module Implementations . . . 86

8 Verification and Validation 87
8.1 Tool Verification . . . 87
8.1.1 Geometry Assessment . . . 87
8.1.2 Manual Design Feasibility . . . 89
8.1.3 Design Space Coverage and Robustness . . . 93
8.1.4 Conclusion and Recommendations . . . 96
8.2 Validation . . . 97
8.2.1 Validation Case . . . 97
8.2.2 Geometry Validation . . . 100
8.2.3 Aerodynamic Validation . . . 102
8.2.4 Conclusion and Recommendations . . . 104

9 Conclusions and Recommendations 106
9.1 Conclusion . . . 106
9.1.1 Research Questions Answers . . . 108
9.1.2 Evaluation of Hypotheses and Research Objective . . . 110
9.2 Recommendations . . . 111
9.2.1 Algorithms and Model Architecture . . . 111
9.2.2 Aerodynamics and Meshing . . . 112
9.2.3 Validation and Design Applications . . . 112
9.2.4 Model Features and Component Extensions . . . 113

References 114

A Fuel Cell technological evolution 119

B Preliminary Parametrisations and Visual Aids for the Intake Configurations from the Literature Review 121
B.1 General Intake Parametrisation . . . 121
B.2 Leading-edge Intake . . . 122
B.3 Scoop Intake . . . 123
B.4 Ring Intake . . . 124
B.5 Underschlung Intake . . . 125
B.6 Flush Intake . . . 126

C STEP Export Algorithm 127

D Full Parameter Specifications for Robustness Study 129
D.1 Examples Generated Designs . . . 131

E Needs and Requirements 133
E.1 Needs . . . 133
E.1.1 Internal Needs . . . 133
E.1.2 External Needs . . . 137
E.2 System Requirements . . . 137
E.2.1 System Requirements 1 . . . 137
E.2.2 System Requirements 2 . . . 144
E.3 Assessment of System Requirements . . . 145
E.3.1 Requirement Satisfaction Scale . . . 145
E.3.2 Requirement Satisfaction Matrix . . . 152

List of Figures

1.1 Projection of CO_2 emissions from aviation [3] . . . 1
1.2 Nacelle of Universal Hydrogen's De Havilland Canada Dash-8 [1]. . . . 3
1.3 Geometrical model of FC TMS used in [16]. . . . 3
1.4 Illustrative overview of the thesis research and tool-development workflow. . . . 7

2.1 Schematic front view and internal geometry of a podded duct and radiator with a preliminary parametrisation. . . . 9
2.2 Schematic front view and internal geometry of a scoop intake with a preliminary parametrisation. . . . 10
2.3 Schematic side view of a propeller and nacelle integrated into the wing with a preliminary parametrisation. . . . 12
2.4 Schematic side view of a spinner with area-ruling. . . . 12

3.1 Definition of the Knowledge Ontology for KBE applications [41] . . . 15
3.2 Requirement stereotypes from SysML 1.0 . . . 16
3.3 Overview of entire requirements diagram with needs and requirements package structure 17

4.1 Generated OML of the`SystemAssembly` of *baseline design* with each component coloured. 19
4.2 Schematic UML diagram of `SystemAssembly` . . . 19
4.3 Flow diagram of a typical design flow of `SystemAssembly`, where the blue processes are tool-integrated processes and grey are optional processes for an example design workflow. 20
4.4 UML representation of the Parametrisation-as-Composition architecture applied to an airfoil. Implemented classes are shown in blue and not implemented classes in grey. The design realises the Strategy Pattern, with classes annotated by their respective pattern roles (Context, Strategy, ConcreteStrategy). . . . 21
4.5 UML representation of the inheritance-based parametrisation approach for an airfoil. Implemented classes are shown in blue and example classes in grey. . . . 21
4.6 UML representation of `TransformDescriptor`, a descriptor capable of creating an `AirfoilTransformParametri` by injecting design data and context. Each class is annotated with its respective pattern role. . . . 23
4.7 Coordinate Data Import Overview . . . 24
4.8 UML representation of the `GeometryExportBase` class implemented by both the `Airfoil` and `LiftingSurface`. The attributes are annotated as root, helper, or child. This diagram serves as a conceptual example; in practice, only references to geometrical attributes are implemented rather than direct instances. . . . 25
4.9 Comparison of the baseline design in ParaPy and the corresponding geometry imported into FreeCAD using the geometry export module. . . . 26
4.10 Unstructured surface mesh generated for the baseline design using 'Moderate' fineness. The zoomed-in region shows alignment between mesh edges and geometrical boundaries (in blue). . . . 27
4.11 Schematic UML diagram of the abstract `Simulation` interface with related objects, implemented for the aerodynamics module. . . . 28
4.12 Schematic UML diagram of the `AeroAnalysis` class, illustrating its key components and their interaction with the FlightStream interface. The `SystemAssembly` and `Wing` classes are shown as representative analysable objects. Colour coding denotes implementation origin: **dark blue** indicates classes and functionalities directly reused from ParaPy, **light blue** represents elements imported with modifications, and **white** denotes newly created and implemented components. . . . 29

4.13 Illustrative aerodynamic outputs generated through the integrated FlightStream workflow. The plots and visualisation serve solely as examples of the tool's automated analysis and export capabilities, not as validated aerodynamic results. 30
4.14 Construction of the Gordon surface using its three interpolation surfaces, adapted from [60]. 32
4.15 Illustrative workflow of the `BlendSolid` algorithm showing the merging of cube and a cylinder into a single continuous blended geometry. 34
4.16 Illustrative workflow of the `BlendCurve` algorithm showing the construction of a Bézier curve from two points on the nacelle and scoop boundary for the baseline design. With two views of the same situations, an iso-view and the same view rotated upwards with a significant amount. 35
4.17 Cylinder-cube blending using `BlendSolid`. Top: uniform trimming distance of 0.15 m in both directions. Bottom: asymmetric trimming with 0.25 m toward the cylinder and 0.05 m toward the rectangular face. 36
4.18 Cylinder-cube blend using an interpolated σ-law starting at 2.0, to 0.5 halfway and back to 2.0 along the intersection curve. 37
4.19 Effect of blend-curve direction control in nacelle-scoop blending. Top row: fully *direct* tangent direction. Bottom row: aligned with the adjacent-face boundary normal. 38
4.20 Sampling convergence of continuity deviations for the baseline nacelle-scoop blend with 10 guides. 39
4.21 `BlendSolid` (orange blending surface) used in different blending cases with identical blending parameters, σ = 1.0, offset = 0.1 m and 10 blending guides (green) with default direction factors = 0.0 and max angle to 45°. 39
4.22 Convergence of continuity deviations of the `BlendSolid` for the simple geometry, with 100 sample points. 40
4.23 Convergence of continuity deviations of the `BlendSolid` for the baseline design at nacelle-scoop integration, with 100 sample points. 40
4.24 Convergence of continuity deviations of the `BlendSolid` for the baseline design at wing-nacelle integration, with 100 sample points. 40

5.1 UML representation of all wing-related classes. Abstract classes (gray), configurable parametric classes (green), and specialisation classes (blue) illustrate the structural and functional relationships between lifting-surface and profile entities. Not all attributes are shown as the diagram serves primarily as a structural overview. 44
5.2 Scaling and transformation of an arbitrary `Profile`. 45
5.3 Example closed `CrossSection` with counter-clockwise orientation. The computed Centre-of-Gravity (CoG) is annotated and horizontal/vertical characteristic points are shown in red. 46
5.4 Clark-Y airfoil with computed characteristics: $t/c_{\max} = 0.118$, LE radius $= 0.011$ m, and $x_{t/c_{\max}} = 0.303\,c$. visualised in the ParaPy viewer 46
5.5 Airfoil constructed using the CST parametrisation. The baseline airfoil (grey) uses shape coefficients of 0.1 and -0.1. The perturbed airfoil (black) modifies the mid-surface coefficient on the upper surface (5 coefficients) to 0.3, and the corresponding coefficient on the lower surface (33 coefficients) to -0.6, demonstrating local controllability. 47
5.6 Grumman K-1 transonic airfoil: CST-fitted curve (black) and imported data (red). The optimisation yields upper-surface coefficients [0.170, 0.140, 0.160, 0.263, 0.170] and lower-surface coefficients [-0.167, -0.096, -0.223, -0.013, 0.095]. 49
5.7 Definitions for the placement of *airfoil-on-rails* [57]. 49
5.8 Illustrative workflow for computing the reference airfoil position for *airfoil-on-rails*, showing the determination of the coordinate system from the LE and TE rails with span line for a span and chord ratio of 0.5. 50
5.9 Wing skeleton for two 3D elliptical rails, with computed MAC $= 5.10\,\mathrm{m}$ and spanline length $= 9.89\,\mathrm{m}$. 51
5.10 Rectangular wing planform with $b = 10\,\mathrm{m}$, $\Lambda = 10°$, $\lambda = 0.4$, $AR = 8$, and a twist axis varying from $0.25c$ at the root to $0.75c$ at the tip, resulting in a computed MAC of $2.59\,\mathrm{m}$. 52

5.11 Wing skeleton from figure 5.9 instantiated as a concrete lifting surface with four NACA0012 airfoils placed along the span using the airfoil-on-rails transform parametrisation. 53
5.12 Procedure for constructing the leading- and trailing-edge rails from a planar planform [57]. 53
5.13 Procedure for applying dihedral to a rail [57]. 54
5.14 Procedure for applying twist to sampled points along a rail [57]. 54
5.15 Example of a lifting surface constructed with the clean wing parametrisation, showing the rails (red), twist axis (green), airfoils (black), and 2D planform (grey). 54
5.16 Illustrative workflow for modifying an original wing into a matched wing by providing a transition region and deformation factor. 56
5.17 Grumman K-1 transonic airfoil as inserted airfoil into a wing with *deformation factor* of 1.5 was used (blue) and compared to an inserted wing of the unscaled wing (black), leading to an actual scale increase of 1.446. 57

6.1 UML representation of all abstract (gray), configurable parametric (green) and specialised (blue) nacelle-related classes, illustrating the structural and functional relationships. . . 59
6.2 Spinner with elliptical shape parametrisation, medium AR (=2) and radius of 1 m. 60
6.3 Internal subsystems embedded in the *baseline* nacelle design for a total power requirement of 4 MW, using estimates of 10 kW/kg for the electrical motor and 2.5 MW/m^3 for FC system. 62
6.4 Computed nacelle cross-section points for a CST-based cross-section ($N_u = 0.5$ and $N_l = 0.1$ with uniform shape parameters) generated within the tool, showing the crown, MHB, and keel points (red). 63
6.5 Examples of nacelle cross-sectional families defined using CST, where $NC1$ and $NC2$ denote the upper and lower class factors, respectively [64]. 64
6.6 Optimised CST nacelle cross-sections for increasing wobble constraint levels ($n = 16$, $N = 5$ data points per side seen in red). 65
6.7 Positioning of two nacelle cross-sections along the nacelle middle line, including relative x- and y-offsets, cant angle (φ), and dihedral angle (Γ). The local cross-section origin is shown in red, the middle line in grey, and the nacelle start in blue. 66
6.8 Concrete nacelle from the *baseline* design showing the nacelle end (red), the middle line (bold black), the nacelle's local coordinate system (and thus start), and the spinner (solid yellow). 67
6.9 Side view of the nacelle-from-cross-sections parametrisation of *baseline* design, showing keel guide behaviour (blue) under different tangency options (from left to right): spinner tangent, smooth, streamwise (local y), and smooth. 68
6.10 Illustrative workflow for constructing a `MatchedNacelle` by defining a new nacelle start location and transition cut. 69
6.11 Inserted nacelle cross-section (blue) at 50% nacelle length in the baseline design, uniformly scaled by a factor of 0.8 and compared to the original nacelle contour (black). . . 70
6.12 Nacelle positioning on a straight wing for the *baseline* design: placement via reference location and offsets, followed by 3D orientation control. 72
6.13 Wing–nacelle geometry matching in the *baseline* configuration, performed by scaling of the `MatchedLiftingSurface` and `MatchedNacelle` based on the geometry at the blend location and specified blending parameters. 73
6.14 Comparison of three wing–nacelle integration strategies on the matched *baseline* configuration. 74
6.15 Wing–nacelle integration of the *baseline* configuration using a 0.5 m fillet radius, 0.5 scaling distribution, chordwise blend at maximum airfoil thickness location, nacelle transition length 0.5, and wing blend width 0.3 m. 75
6.16 Local airfoil-shape preservation mid-blend for fillet-based integration on the *baseline* configuration. 75

7.1 UML representation of all abstract (gray), configurable parametric (green) and specialised (blue) scoop-related classes, illustrating the structural and functional relationships. . . . 77

7.2 Example B-spline used as a *direct* lip curve, with automatically computed highlight (tip), leading-edge radius (blue), throat point (red), maximum internal wall angle location (green), droop angle and camber line. 78
7.3 Euler centerlines compared to a baseline design (black; $l_{\text{lip}} = 1$ m, $\varphi_\ell = 20^\circ$, $\sigma_{\text{euler}} = 0.0$), with variations in *length* (red; 1.2 m), *shape parameter* (blue; $\sigma_{\text{euler}} = 1.0$) and *droop* (green; $\varphi_\ell = 30^\circ$). 79
7.4 Example *clean lip* design showing centerline (gray), highlight-based LE radius (blue, 0.025 m), droop (20°), length (1.0 m), aft outer-wall thickness (0.05 m), aft inner-wall thickness (0.1 m), and CST-based shaping of both inner and outer surfaces. 80
7.5 *Direct scoop profile* with $h_d = 0.5$ m and outer lip reference, constructed from two clean lips. Intake parameters (θ_s, h_c, h_h, h_d, l_c) are computed from the lip geometry. 81
7.6 *Clean scoop profile* with $\theta_s = 20^\circ$, $h_c = 0.5$ m, $l_c = 0.3$ m, droop (0°/ 30°), and radii (0.01 m/ 0.025 m). CST shaping parameters are omitted for brevity. 81
7.7 *Direct scoop* parametrisation built from the profile in figure 7.6 and a CST intake-area curve with class exponents $N_1 = N_2 = 0.5$ and shape coefficients $[0.5, -0.5]$. Coordinate system represents the intake local reference frame and thus position. 82
7.8 Effect of inserting intermediate cross-sections on LE fidelity for the scoop in figure 7.7. . 82
7.9 Illustrative workflow for constructing the nacelle-scoop integration by positioning of the scoop, definition of reattachment cross-section, external duct construction via Gordon surface, and final transition blending using `BlendSolid`. 84

8.1 Parametric model of the *baseline* design, generated using the developed tool. 90
8.2 Reference photograph of the De Havilland Canada DHC-8 turboprop aircraft used for the conceptual comparison. 91
8.3 Parametric reference model resembling the De Havilland Dash 8, generated using the developed tool. 91
8.4 Reference photograph of the battery-electric Elysian concept. 92
8.5 Parametric reference model resembling the Elysian battery-electric concept, generated using the developed tool. 93
8.6 Histogram of failure codes during parametric robustness study, coloured by subsystem category. 95
8.7 NLF-Mod22(B) airfoil with nested flap [72] (modified). 98
8.8 Technical drawing of the TUD-PWF reference model with key dimensions (mm) [73]. . . 98
8.9 TUD-PWF nacelle reference geometry, visualised in the ParaPy viewer. 99
8.10 TUD-PWF validation model within the tool. 100
8.11 Histogram of point-to-surface distances computed by CloudCompare. 100
8.12 Point-to-surface distances between tool-generated geometry and reference CAD for TUD-PWF model, visualised in CloudCompare. 101
8.13 Zoomed views of key regions showing point-to-surface distances in CloudCompare. . . 102
8.14 Simulation setup for the TUD-PWF configuration generated by the tool. 103
8.15 Simulated aerodynamic data from FlightStream compared with uncorrected windtunnel data [72]. 104

B.1 Schematic front view and internal geometry of a podded duct and radiator with a preliminary parametrisation. 121
B.2 Three-quarter view of a LE model, inverted showing fairing between ducted and plain airfoil sections. [77] 122
B.3 Schematic front view and internal geometry of a LE intake and radiator with a preliminary parametrisation, on top of the original airfoil of the wing. 123
B.4 Example of a scoop intake on a turboprop engine. [78] 123
B.5 Schematic front view and internal geometry of a scoop intake with a preliminary parametrisation. 124
B.6 Full nacelle with ring intake for cooling of a fully electrical propulsion system. [79] 125
B.7 Schematic side internal view of a ring intake with a preliminary parametrisation. 125
B.8 View of a rear underslung fuselage duct with internal BL diverter [80] 125
B.9 Drawing of an underslung intake with external BL diverter [81] 125

B.10 A NACA intake used as primary intake on the YF-93. [82] 126
B.11 Plan view of flush intakes including the NACA intake. [83] 126

C.1 Flow diagram of the compound-construction algorithm for collecting all geometry and children in the model tree during STEP export. 128

D.1 Some designs from the parametric robustness study. 132

E.1 Needs diagram for internal needs based on Hypothesis 1 133
E.2 Needs diagram for internal needs based on Hypothesis 2 135
E.3 Needs diagram for external needs 137
E.4 System Requirements 1 package (1/3) 138
E.5 System Requirements 1 package (2/3) 140
E.6 System Requirements 1 package (3/3) 142
E.7 System Requirements 2 package 144

List of Tables

4.1 Continuity performance summary of `BlendSolid` for three geometric configurations. . . . 41

8.1 Geometry verification failure codes used in the automated robustness assessment. . . . 89
8.2 Geometric deviation metrics between tool-generated and reference geometry 100

A.1 Fuel Cell Technological Evolution predictions from the last 20 years.[1] 119

B.1 Initial design variables for intakes in general 122
B.2 Initial design variables for Leading-Edge intakes. 123
B.3 Initial design variables for scoop intakes. 124
B.4 Initial design variables for ring intakes. 125
B.5 Initial design variables for underslung intakes. 126

D.1 Parametric robustness sampling design space overview. 129

E.1 Requirement satisfaction matrix. NA = not assessed; NS = not satisfied; MS = marginally satisfied; PS = partially satisfied; LS = largely satisfied; FS = fully satisfied. 152

Nomenclature

Abbreviations

Abbreviation	Definition
API	Application Programming Interface
ATP	`AirfoilTransformParametrisation`
BL	Boundary Layer
B-rep	Boundary representation
CAD	Computer-Aided Design
CADx	augmented Computer-Aided Design
CFD	Computational Fluid Dynamics
CM	Capability Module
CoG	Centre-of-Gravity
CPACS	Common Parametric Aircraft Configuration Schema
CST	Class-Shape Transformation
CSV	Comma-Separated Values
FC	Fuel Cell
FFD	Free-Form Deformation
FS	FlightStream
ICE	Internal Combustion Engine
KBE	Knowledge-Based Engineering
LE	Leading Edge
MAC	Mean Aerodynamic Chord MBSE
Model-Based Systems Engineering	
MDO	Multi-disciplinary Design Optimization
MHB	Maximum Half Breath
MMG	Multi-Model Generator
MW	MegaWatt
NACA	National Advisory Committee for Aeronautics
OCC	Open Cascade
OML	Outer Mould Line
RMS	Root Mean Squared
SLSQP	Sequential Least-Squares Programming
STEP	Standard for the Exchange of Product model data file
UML	Unified Modelling Language
TE	Trailing Edge
TMS	Thermal Management System
TU Delft	Technical University Delft
WIN-Gen	Wing-Intake-Nacelle Generator
WNI	Wing-Nacelle-Intake

Symbols

Symbol	Definition	Unit
A_c	Area at intake face	[m^2]
A_h	Area of intake highlight	[m^2]
CR	Contraction ratio	[-]
$C_{L_{max}}$	Maximum lift coefficient	[-]
d_n	Nacelle diameter	[m]
G^0	Location continuity	
G^1	Tangential continuity	
h_c	Intake height	[m]
h_{dim}	Dimple height	[m]
h_e	Exit height	[m]
h_h	Highlight height	[m]
l_c	Intake length	[m]
l_d	Cowl/duct length	[m]
l_{dim}	Dimple length	[m]
R	Radius	N.A.
r_l	Lip radius	[m]
t/c	Thickness over chord ratio	[-]
w_c	Intake width	[m]
x_c	Intake axial location	[m]
x_p	Propeller axial location	[m]
$\overline{y}$	Camber	[-]
z_p	Propeller vertical location	[m]
Γ	Dihedral	[°] δ_{max}
Maximum `BlendCurve` angle	[°]	
ζ	CST shape ordinate	[-]
θ_s	Stagger angle	[°]
κ_e	External curvature	N.A.
κ_i	Internal curvature	N.A.
κ_{int}	Nacelle-wing integration curvature	N.A.
κ_n	Nacelle-prop integration curvature	N.A.
κ_s	Spinner curvature	N.A.
σ	`BlendSolid` sharpness parameter	[-]
φ_w	Wall angle	[°]
φ_l	Lip droop angle	[°]
φ_p	Peripheral extend	[°]
ψ	Non-dimensional chordwise coordinate	[-]
Ω_c	Cross-sectional shape intake	N.A.
Ω_d	Cross-sectional shape duct	N.A.

1

Introduction

The global shift towards sustainable energy has grabbed the world by storm with the urgent effort to reduce emissions in aviation, which currently accounts for approximately 12% of global transport-related greenhouse gas emissions [2]. Despite annual improvements in fuel efficiency of around 2%, the continued growth of air traffic, projected at 3.5% per year, has led to a rise in overall CO_2 emissions, as figure 1.1 indicates. Achieving the aviation sector's 2050 sustainability goals will require the adoption of green fuels, among which fuel cell-powered (FC-powered) aircraft have emerged as a promising alternative with several feasibility studies indicating their potential for regional and short-range transport [3–5]. However, these studies indicate the general consensus that the success of hydrogen technology in aviation is highly dependent on technological advancements in the field.

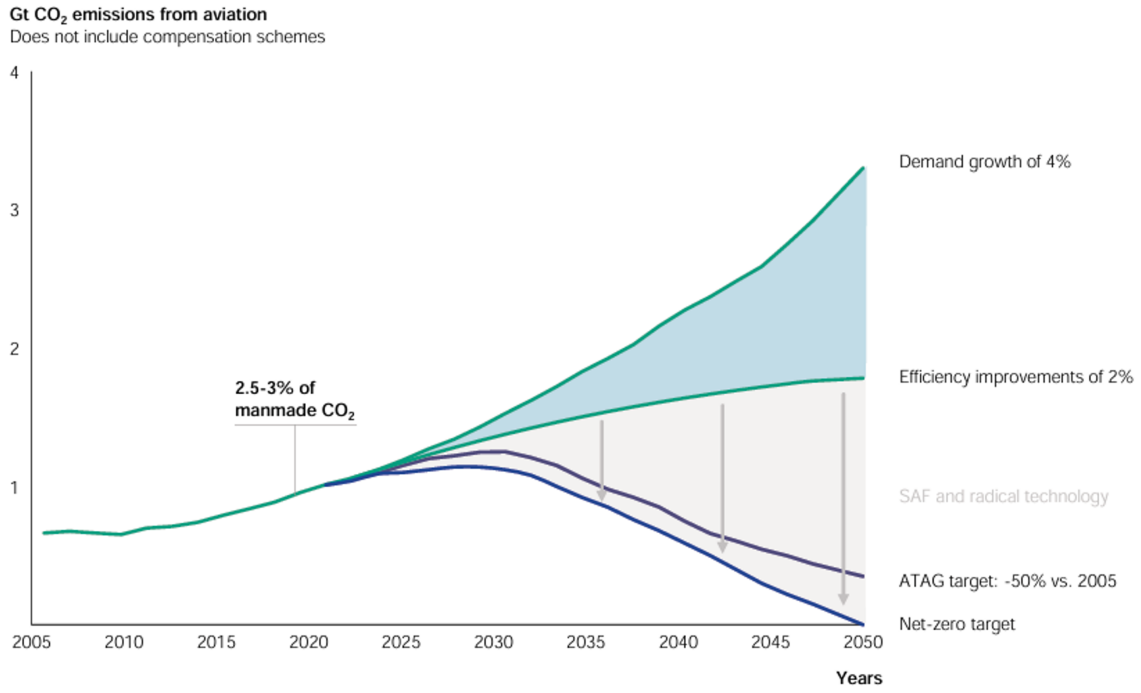


Figure 1.1: Projection of CO_2 emissions from aviation [3]

One key challenge lies in propulsion system integration. For conventional internal combustion engines (ICE) aircraft, engine installation drag can contribute to roughly 10-15% of total aircraft drag [6], emphasizing the importance of engine integration and nacelle design. Traditional ICE engines have followed a trend toward larger fan diameters, necessitating podded engine configurations to accommodate the increased engine and thus nacelle size. Consequently, for turbofans the nacelle profile drag constitutes

up to 70% of total propulsion installation drag that rises exponentially with increasing engine diameter [7].

In contrast, FC propulsion systems employ electrified propulsion architectures that offer increased flexibility in propulsor size, placement, and number, creating opportunities to rethink engine installation entirely for FC aircraft [8, 9]. Novel concepts such as over-the-wing nacelles, boundary layer (BL) ingestion and distributed propulsion have demonstrated measurable aerodynamic and aero-propulsive performance benefits [10]. For FC-powered and other electrified propulsion systems, thermal management system (TMS) design becomes a key driver, as intake airflow is primarily required for cooling rather than thrust generation, often resulting in relatively large intake requirements [11]. Consequently, retrofitting conventional ICE nacelle architectures for FC propulsion generally leads to suboptimal designs, as such approaches fail to exploit the available design flexibility and do not adequately address the fundamentally different intake, cooling and volumetric requirements of FC systems.

Since FC technology in aviation is novel technology, limited research exists on the aerodynamic integration of such propulsion systems. Many studies remain conceptual and lacks methodologies that capture the geometric and aerodynamic interactions essential for external and internal flow aerodynamics. This indicates a clear need for systematic approaches and tools that enable the design and analysis of new and innovative FC propulsion integrations within the aircraft's framework. Such tools could facilitate research into these new aerodynamically attractive integration concepts, while still enabling (re)design of existing wing-nacelle-intake (WNI) integrations.

1.1. Integration of Fuel Cell Thermal Management Systems in Aircraft

FC-powered aircraft require electrical power outputs on the order of several megawatts to sustain flight [12]. With projected overall powertrain efficiencies between 50–70% [13], a substantial portion of this power is dissipated as heat, making TMS design an important driver. The sizing and integration of the TMS directly affect the aerodynamics and performance of the aircraft. Numerous methodologies have been proposed for the preliminary design and sizing of FC thermal management systems [14–19][1], often incorporating simplified aerodynamic models to estimate cooling drag and integration aerodynamics. However, these approaches typically rely on empirical or incomplete aerodynamic formulations, sometimes considering only drag effects and neglecting full aerodynamics phenomena like lift effects or interference. To conclude and further detailed in section 2.1, no comprehensive framework currently exists that fully addresses the aerodynamic effect of TMS integration for FC-powered aircraft [13].

Several methods have been developed for the sizing of FC-powered aircraft [20, 21][1], as well as for the sizing of the individual FC subsystem [17–19][1]. These approaches primarily focus on estimating power requirements, component efficiencies, and overall system mass, while largely neglecting the geometric implications of system integration. A limited number of studies attempt to include geometric aspects of the FC subsystem [15, 16][1], of which the most complex geometrical model can be seen in figure 1.3 which lacks the geometrical and integration fidelity seen in a real design of a FC propulsion integration system in figure 1.2. Consequently, there is no established methodology that accounts for whether the FC system can be realistically integrated into the aircraft, nor one that relates the external and internal aerodynamic geometry to the size, shape, or placement of the FC subsystem within a WNI configuration.

[1] Each model is extensively addressed with scope and limitations in section 2.1.

Figure 1.2: Nacelle of Universal Hydrogen's De Havilland Canada Dash-8 [1].

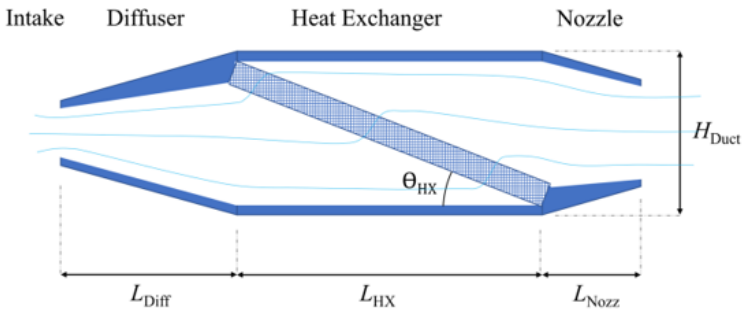


Figure 1.3: Geometrical model of FC TMS used in [16].

1.2. Unconventional Intake Configurations

The unique design scenario outlined at the beginning of this chapter introduces new opportunities to explore intake configurations[2] other than the conventional podded nacelle-under-wing design that may offer superior aerodynamic performance for FC-powered aircraft. Such concepts have received little attention since the early 20th century, when smaller frontal area intakes were prevalent [22]. Following the advent of larger commercial aircraft in the post-1950s era, research interest shifted toward conventional podded nacelle and intake architectures better suited to high-bypass ratio turbofans [23]. As a result, modern research on these potentially high-performing intake concepts remains scarce. However, with the availability of advanced design methodologies and techniques, such as Computational Fluid Dynamics (CFD) and Multidisciplinary Design Optimization (MDO), these *unconventional*[3] intake configurations can now be revisited to develop more aerodynamically efficient and innovative solutions tailored to the topological and cooling requirements of FC-powered aircraft.

1.3. Propeller and Nacelle Integration

Fully electric propulsion systems typically employ propellers for thrust generation, which is highly coupled to aerodynamics of the wing, nacelle, and intake and therefore design (section 2.3). Since the nacelle geometry is highly coupled to the placement and shaping of the FC subsystem, the aerodynamic interaction between the propeller and the WNI system becomes a critical factor in determining overall system performance and external geometry. For that reason, propeller and specifically nacelle integration is a critical aspect in this work.

1.4. Research topic

Based on the summarised findings (full literature review can be found in chapter 2) from previous sections, three problems definitions, and corresponding gaps in literature, can be identified:

1. Electrified FC propulsion fundamentally alters nacelle placement, intake function, and TMS-driven sizing requirements compared to conventional ICE architectures. As a result, retrofitting ICE engine nacelles for FC propulsion leads to aerodynamically suboptimal configurations.
2. No generic or systematic methodology currently exists for assessing the integration of FC systems with respect to outer mould line (OML) and overall aerodynamic performance.
3. *Unconventional*[3] intake configurations, which may offer significant aerodynamic advantages for FC-powered aircraft, have received little attention in recent research as they may be part of the solution making FC-powered flight possible.

These identified challenges define the motivation and direction of this thesis. Together with the research scope outlined in the following section, they form the foundation of the work in this thesis, of which the overall structure is detailed in section 1.4.5.

[2] Listing and qualitative aerodynamic analyses of various configurations and intakes in general are presented in section 2.2.

[3] In this thesis, *unconventional* WNI configurations denote propulsion–airframe integration concepts enabled by electrified propulsion, such as distributed propulsors and wing-integrated or over-the-wing nacelles, which deviate from the *conventional* single- or twin-engine under-wing podded nacelle architectures.

1.4.1. Scope of the Research

The scope of this research is defined as follows:

- The research is **limited to the external geometry**, or outer mold line (OML), of WNI configurations representative of FC-powered aircraft.
- **Parametric geometry generation valid for aerodynamic analyses** is the primary focus, enabling the automatic creation of robust and geometrically valid[4] WNI configurations to be used directly in aerodynamic analyses
- **Sufficient fidelity for preliminary design** is included in the geometries, capturing global flow characteristics and aerodynamic interaction effects while omitting fine-scale geometric details.
- The modelling framework remains **extendable with the initial focus on external aerodynamics**, making it extendable to other disciplines within a multidisciplinary design context.
- Parametrised **representations of the wing, nacelle, and intake** are included, encompassing both conventional and unconventional WNI concepts to support innovative configuration exploration and redesign.
- **Simplified (FC) subsystem geometries** are included to enable investigation of the coupling between external aerodynamic design and internal system placement and sizing.
- **Propeller-wing-nacelle aerodynamic interaction effects** are incorporated, as their strong aerodynamic coupling is essential for capturing realistic integration effects in FC aircraft configurations.

1.4.2. Research objective

From the problem statements and scope defined previously, a solution in terms of a single research objective can be formulated:

"Develop an extendable parametric tool capable of generating a vast selection of computationally valid external geometries for wing-nacelle-intake studies of fuel cell-powered aircraft."

The emphasis of this objective lies in the development and its extendability of the tool. Given the limited scientific literature available on *unconventional*[3] intake configurations and FC propulsion integrations, it is neither feasible nor intended for this first version to encompass a fully comprehensive parametrisation that captures all possible design variations. Different design methodologies inherently influence how configurations are defined and parametrised; therefore, the tool must not be constrained to a single formulation or modelling approach. Instead, it should provide a flexible foundation that allows for future modular extension as knowledge evolves.

Furthermore, the scope of this thesis focuses on geometry generation for external aerodynamics. The developed tool should therefore facilitate the inclusion of additional disciplines, such as structural, thermal, or internal layout analyses, while ensuring that included aerodynamic analyses modules can be expanded and refined in subsequent research.

1.4.3. Hypotheses

In the pursuit of the previously defined research objective, two major hypotheses can be defined. Correct implementation of the tool as described, aims to verify these hypotheses. In addition and most importantly, when these two hypotheses are confirmed, this will lead to solving the problem statements defined in the intro of section 1.4 and confidently the accompanying research gaps.

Hypothesis 1 A parametric design tool, capable of generating valid outer mould lines for wing-nacelle-intake systems throughout a large design space will enable the identification of aerodynamically effective configurations through the systematic exploration of both conventional and unconventional propulsion integration concepts.

This first hypothesis relates directly to the core objective of developing a parametric tool for geometry generation. It focuses on the tool's ability to produce valid[4] external geometries for WNI configurations physically representative of FC-powered aircraft. By doing so, a large design space beyond conventional under-the-wing single podded nacelle architectures can be explored. This directly addresses

[4] Exact definition of *valid geometry* in this context is presented in section 8.1.1.

problem 1, by providing an alternative to the retrofitting of internal combustion engine nacelles. Additionally, the tool can be used to systematically analyse and explore such configurations, addressing the second problem statement in section 1.4. Fulfilling this hypothesis therefore does require that the generated geometries are not only diverse but also physically valid, aerodynamically meaningful, and suitable for use in downstream analyses such as aerodynamic or multidisciplinary simulations.

Hypothesis 2 An extendable and centrally structured design tool can serve as a primary medium for capturing, structuring, and reusing aerodynamic and geometric knowledge related to FC propulsion integration, enabling systematic exploration and understanding of intake design and engine integration concepts beyond conventional architectures.

The second hypothesis emphasizes the function of the tool as a centralized and extendable environment that supports systematic knowledge collection and design understanding of FC propulsion integration. This directly contributes to overcoming the third problem statement in section 1.4, where limited research on *unconventional*[3] intake configurations and FC propulsion integration has restricted current aerodynamic design knowledge. Furthermore, by enabling modular extension of parametrisations and disciplines, the framework allows the continuous refinement and iteration of FC propulsion integration methods as new insights emerge. In turn, these new insights support the development of more efficient and better integrated designs, resolving the first problem statement in section 1.4 by providing improved solutions rather than adaptations of conventional architectures. Ultimately, this hypothesis positions the tool not only as a design generator but also as a critical tool for continuously evolving research environment of FC aircraft technology.

1.4.4. Research questions

In order to evaluate these hypotheses and guide the development of the tool, several research questions are formulated, directly linked to the hypotheses and thus research objective. Providing answers to these questions will lead to confirmation of the hypotheses, and solving the problem statements and thus ultimately achieving the research objective.

RQ 1 - Parametric Framework and Methodology

How can a parametric modelling tool be formulated to automatically generate physically valid OMLs for WNI systems representative of FC-powered aircraft?

This question forms the foundation of the research, and thus subsequent research questions, and directly addresses the objective defined in section 1.4.2, within the boundaries of the scope outlined in section 1.4.1. It is directly related to Hypothesis 1, as it establishes the methodological basis for developing the parametric tool.

RQ 2 - Identification of Aerodynamically Attractive Designs

How can the developed tool be used to identify WNI configurations for FC-powered aircraft that exhibit improved aerodynamic performance metrics compared to conventional podded nacelle designs?

This question directly relates to the first problem statement and to Hypothesis 1, particularly the assertion that the tool will "…enable the identification of aerodynamically effective configurations…". It examines whether the parametric tool can be applied to systematically evaluate wing–nacelle–intake configurations that demonstrate quantifiable improvements in aerodynamic performance, such as reduced drag penalties or improved aerodynamic efficiency, relative to conventional podded nacelle arrangements.

RQ 3 - Assessment Methodology for Fuel Cell Propulsion Integration

To what extent can the developed tool be applied to systematically analyse the aerodynamic performance and spatial integration of fuel cell subsystems in aircraft?

This question addresses the second problem statement concerning the absence of systematic methods for FC subsystem integration. It evaluates to what extent the tool can serve as a generic aerodynamic and geometric analysis framework for FC propulsion and thermal-management systems, and identifies the practical limitations that currently prevent full applicability.

RQ 4 - Parametrisation of Intake Configurations

Which geometrical parameters must be incorporated into the tool to enable research and into configurations of WNI systems?

Addressing the third problem statement, this question focuses on the correct implementation of key geometric and aerodynamic parameters for the unconventional intake concepts introduced in section 1.2, as well as propeller-nacelle interactions discussed in section 1.3. It is also linked to Hypothesis 2, as improving *"...understanding of intake design and engine integration..."* is highly dependent on a relevant and flexible parametrisation approach.

RQ 5 - Valid Geometry Generation

What level of geometric fidelity is required for the generated geometries to be considered usable for aerodynamic analysis and other downstream analyses or design tools?

This question is critical in validating the tool's functionality and results and corresponds to Hypothesis 1, which requires the generation of *"...valid external geometries for wing–nacelle–intake systems..."*. It addresses the geometric fidelity to ensure that the output geometries are both physically meaningful and compatible with aerodynamic or multidisciplinary analysis tools.

RQ 6 - Extendability of Parametrisations and Disciplines

How can the tool architecture be designed to remain extendable, both in terms of additional parametrisations and the inclusion of new disciplinary modules?

Given the limited research on *unconventional*[3] WNI configurations, as noted in the third problem statement, it is not yet possible to define at this stage, a complete parametrisation that captures the full relevant design space. Moreover, different research and designing contexts may require distinct parametrisations as for example a different design objective may require a different and more relevant parametrisation of a component. Consequently, as indicated in Hypothesis 2, which emphasises an *"...extendable design framework..."* this question examines how modularity and scalability can be achieved in the tool architecture to allow for new parametrisations for the included components and disciplinary extensions.

RQ 7 - Knowledge and Understanding of Fuel Cell-powered Aircraft

To what extend can the tool contribute to systematic knowledge reuse and improved understanding of propulsion integration for FC-powered aircraft?

This question relates to the limited research on the aerodynamic and geometric integration of FC propulsion systems highlighted in the second problem statement. It directly supports Hypothesis 2, which states that the framework *"...will facilitate systematic knowledge collection and reuse..."* by evaluating how the tool can facilitate in the understanding of FC-powered aircraft and the accompanying aerodynamic phenomena and contribute to the broader body of design knowledge.

1.4.5. Research Structure

The research methodology follows a structured six-phase workflow, illustrated in figure 1.4. It begins with the formulation of an idea, as introduced in chapter 1. Based on a comprehensive literature review [24], three problem statements are established by identifying specific knowledge gaps summarised in section 1.1, section 1.2, and section 1.3. The first *Context & Problem Formulation* phase concludes with the definition of preliminary parametrisations, presented in the literature report [24] and summarised in chapter 2, together with a clear definition of the scope per component. The subsequent *Research Design* phase establishes the foundations for the remainder of the thesis through the formulation of the research objective, decomposition into two hypotheses, and the introduction of supporting research questions, all presented in this section.

The *Methodology* phase begins with the selection of a suitable modelling framework in chapter 3, leading to the derivation of internal needs and concrete system requirements aligned with the hypotheses. The fourth phase, *Tool Development*, is initiated by defining the complete system architecture in chapter 4, with shared functionalities and introducing key modelling concepts used throughout the thesis. The full component geometry generation and integration capabilities are subsequently implemented:

the wing module in chapter 5, the nacelle and enclosed FC subsystems in chapter 6, and the scoop intake[5] in chapter 7.

Using these components and integration routines, the *Product Assessment* phase evaluates the tool through verification and validation procedures, presented in chapter 8. Finally, the research is synthesised in the *Research Evaluation* phase in chapter 9, which summarises key findings, answers the research questions, evaluates the hypotheses, and assesses fulfilment of the research objective. The thesis concludes by reflecting on the original problem statements and identifying recommendations and priorities for future work.

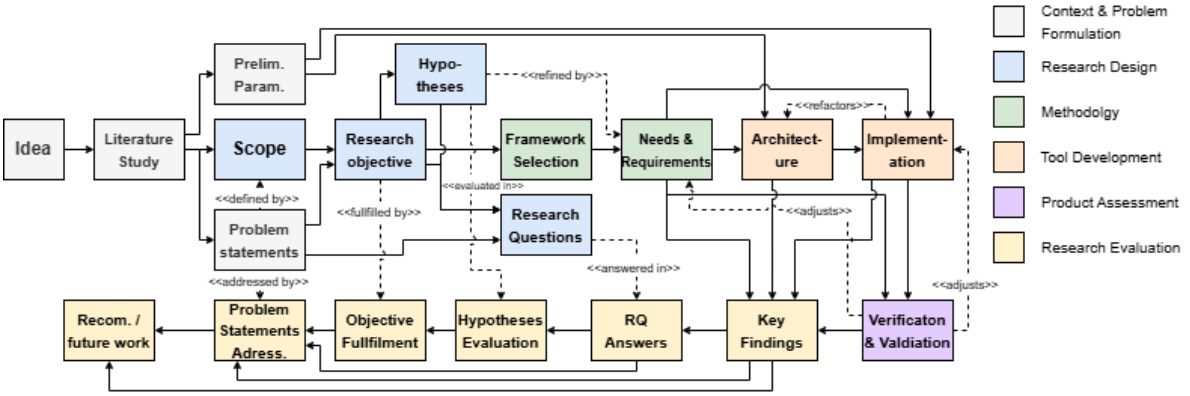


Figure 1.4: Illustrative overview of the thesis research and tool-development workflow.

[5]The scoop intake is currently the only implemented intake configuration due to project scope and time constraints. The broader intake modelling scope is discussed in section 2.2.1.

2

Literature Review: Key Findings and Research Scope

This chapter establishes the foundational basis required for developing a parametric design tool for WNI systems in FC-powered aircraft. The tool is built upon key aspects of FC technology and intake and propeller–wing aerodynamics to address the geometric challenges of FC propulsion integration.

As a full literature study has been conducted and documented in an extensive literature review report [24], all information presented in this chapter is derived from that report. This chapter therefore serves as a concise summary of the key findings directly relevant to this research, and no extensive individual literature references are included here. The reader is referred to the literature review report for full referencing and detailed discussion of the sources.

An overview of fuel cell technology in aircraft, focused on associated system sizing methodologies, is presented in section 2.1. A summary of intake aerodynamics, discussing various intake configurations including preliminary parametrisations, is given in section 2.2. Finally, section 2.3 summarises the aerodynamic interactions between the propeller, wing, and nacelle, highlighting their influence on propulsion integration. Each section concludes with a scope definition and modelling approach adopted for the respective subsystem, thereby forming the conceptual basis for the parametric tool developed in subsequent chapters.

2.1. Hydrogen Fuel Cell Technology in Aircraft

The use of hydrogen in aviation began with early hydrogen-fuelled gas turbines in the first half of the 20th century, but renewed interest has emerged only recently as environmental pressures intensified and attention shifted toward FC-based electric propulsion, the focus of this thesis. Demonstrators such as ZeroAvia's Dornier 228 (2023) [25] and Universal Hydrogen's Dash 8 (2023) [26] have now shown that hydrogen-powered FC flight is technically feasible, and numerous ongoing projects[1] collectively indicate that hydrogen-propelled aircraft are likely to become technologically viable by 2050 [13].

The architecture of an FC propulsion system, consisting of the FC stack, balance-of-plant subsystems, hydrogen storage, and a megawatt-scale electric powertrain, introduces significant thermal loads that must be aerodynamically integrated. However, current literature on thermal-management integration remains limited as most studies rely on simplified empirical drag formulations or highly case-specific geometries, highlighting that a comprehensive aerodynamic model for FC thermal-system integration is still missing. Likewise, existing sizing approaches at both aircraft level and subsystem level tend to prioritise mass and power estimates over geometric fidelity, offering limited guidance on how internal FC architecture couples with the OML and aerodynamics. This gap motivates the need for more detailed, geometry-aware sizing and integration methodologies for FC propulsion systems.

[1] A full overview of recent FC developments is provided in chapter A.

2.1.1. Scope of Fuel Cell Sizing and Thermal Management in the Model

Because detailed topological constraints and robust geometric-aerodynamic coupling strategies for high-powered FC systems are not yet well established in the literature, this thesis incorporates simplified internal FC subsystem geometries. These subsystems are included primarily as geometric references as their exact interaction with the OML geometry is outside the scope of this work which is thus a known limitation in this work. Furthermore, the model does not incorporate an Initiator-type sizing module [27] for determining subsystem dimensions or component layouts based on mission requirements or aircraft configuration. All subsystem sizes and geometric constraints are instead treated as user-specified inputs to the model.

2.2. Intakes

As state-of-the-art thermal-management methodologies for FC aircraft address external aerodynamic integration only partially if at all, this thesis scopes FC subsystem integration primarily through the lens of external aerodynamics. For the TMS, the dominant external element is the intake, making intake aerodynamics a central driver of many design choices in this work.

An intake must supply an air-breathing system with the required mass flow while maintaining air quality and minimising aerodynamic losses. Following the duct concept, mass flow is governed mainly by the exit area, while internal aerodynamics focuses on pressure recovery and flow quality, driven by friction, turbulent mixing, and susceptibility to BL ingestion. These losses are strongly influenced by geometric parameters such as internal curvature (κ_i), wall angles (φ_w), contraction ratio (CR), lip radius (r_l), and stagger (θ_s). External aerodynamics, in turn, seeks to minimise drag originating from friction, pressure, spillage, and installation effects. Interference accelerates the local flow and thickens the BL, while upwash alters the incidence angle at the intake face. Since FC cooling performance is highly sensitive to inlet flow quality, preventing internal separation is essential. Collectively, these aerodynamic mechanisms motivate the general set of geometric design variables illustrated in figure 2.1. These variables form the basis for the parametrisation of the scoop intake in chapter 7 where more implementation details are presented specifically regarding restrictions, coupling and ranges of the parameters.

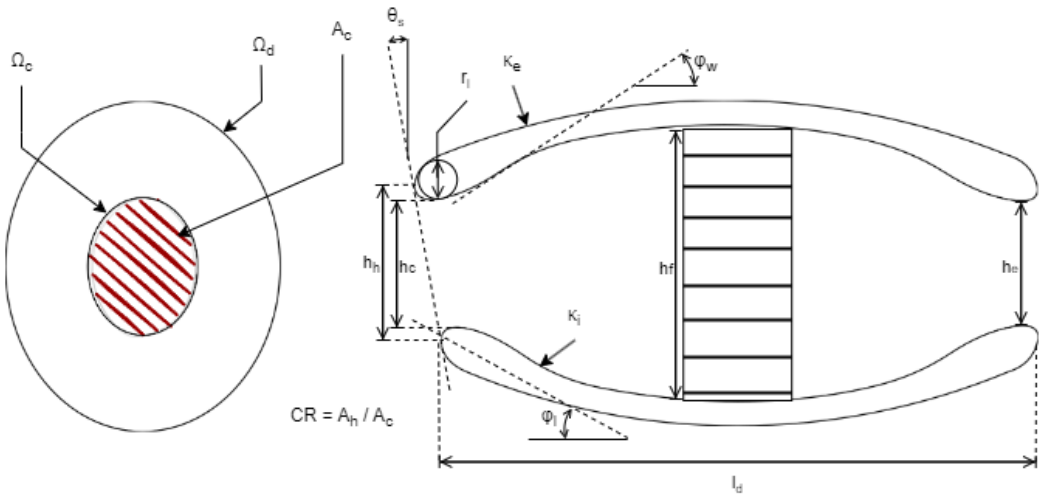


Figure 2.1: Schematic front view and internal geometry of a podded duct and radiator with a preliminary parametrisation.

Leading-edge Intake

Leading-edge (LE) intakes are integrated directly into the wing leading edge and therefore strongly interact with the wing aerodynamics. They must tolerate varying flow ratios and upwash-induced inlet incidence, but if carefully designed they add virtually no external wetted area and introduce minimal interference, especially when the intake face is placed near the stagnation point and canted favourably toward the propeller slipstream. Their main limitations are geometric and structural as the intake height is constrained by wing thickness, meaning larger mass-flow needs quickly drive spanwise width and reduce wing efficiency, which, together with structural complexity, explains why LE intakes have seen

little modern use despite promising historical NACA results[2]. Experiments nonetheless show that suitable combinations of stagger, upper and lower lip radii and thickness, camber, and opening height can yield LE intakes with high $C_{L_{\max}}$ and good pressure recovery. A more recent study on such a design showed significant aerodynamic challenges remain for this concept to be viable for FC-powered flight due to the high cooling requirements [11], emphasizing the need of subsequent research into the full geometrical integration of the concept.

Scoop Intake

Scoop intakes, by contrast, are forward-facing projections attached to the nacelle or fuselage that deliberately intake flow outside the BL, improving pressure recovery relative to intakes located behind large forebody wetted areas. This comes at the cost of added frontal area, drag, and weight. Their external drag depends on intake length (l_c), cross-sectional shape (Ω_c) and blending into the body, while internally the strong bend into the nacelle increases internal and external curvature (κ_i and κ_e), friction losses, and separation risk. BL ingestion is a dominant issue, mitigated by diverters or bleed systems whose height must scale with the local BL thickness, and by placing the scoop further forward to reduce forebody losses. Cross-sectional shape (height; h_c, width w_c, or peripheral extent φ_p), lower-lip camber ($\overline{y}$), and axial location (x_c) collectively control both BL ingestion and separation tendencies. Primary geometrical parameters influencing these characteristics can be seen in the preliminary parametrisation in figure 2.2.

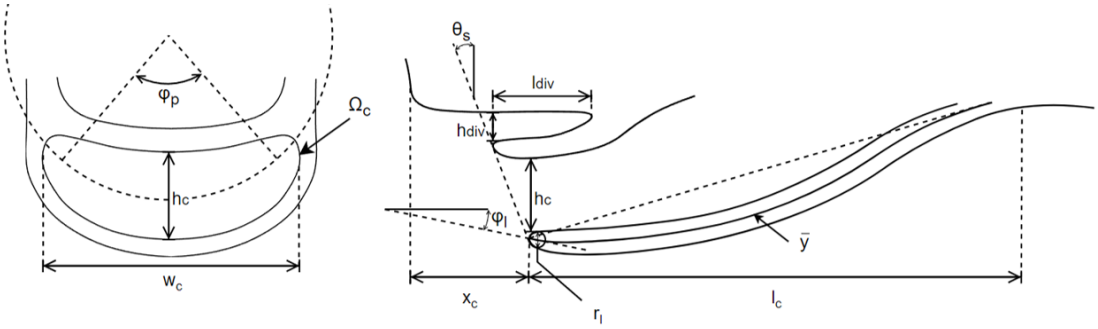


Figure 2.2: Schematic front view and internal geometry of a scoop intake with a preliminary parametrisation.

Ring Intake

Ring (annular) intakes wrap around the nacelle like a torus and behave similarly to podded inlets but with a much larger central body. They offer excellent external performance with low added wetted area and spillage drag, and can achieve high critical Mach numbers when the outer ring is smoothly cambered, yet suffer from poor internal efficiency due to high wall friction on both inner and outer ducts and strong sensitivity to BL ingestion, aggravated by low intake heights and significant forebody area. Their performance is therefore governed by intake height, axial location, and lip geometry, and although internal layouts can sometimes provide relatively straight ducts, ring intakes remain vulnerable to incidence effects, making careful design of lips and placement essential.

Exit

The exit controls the mass flow through the duct and therefore the cooling capability, with larger exit areas enabling higher throughflow. After the radiator, the flow must be re-accelerated through a converging nozzle to minimise the drag penalty associated with internal losses, making short duct lengths (i.e. exits positioned close to the intake) preferable. Depending on the configuration, exits may be placed on the nacelle (for scoop or ring intakes) or on the wing (for LE or underslung intakes). Experiments show that the low-energy outflow can induce separation when it merges with the freestream, particularly near maximum expansion, making lower-surface exits aerodynamically favourable. Conversely, exits located near the trailing edge (TE) can negatively affect both drag and maximum lift, underscoring the sensitivity of exit placement to overall aerodynamic performance.

[2]Full list of all NACA studies between 1915-1949 can be found at [28]

Underslung and Flush Intake

Underslung intakes resemble scoop intakes but are positioned aft on the wing or fuselage, with the duct exiting directly beneath the body rather than feeding into an internal duct. Their placement allows favourable high-incidence performance and reduced compressibility issues, yet the large forebody wetted area makes them highly prone to BL ingestion, requiring diverters or bleed systems that increase drag. Overall, they tend to underperform compared to LE intakes.

Flush or NACA-type intakes, while offering minimal frontal area and spillage drag, suffer from severe BL ingestion, lip separation at high flow ratios, and strong sensitivity to ramp angle, which necessitates long geometries that conflict with nacelle and wing integration. Although NACA's vortex-based BL diversion reduces approach losses, it introduces additional bleed drag and can significantly disturb the wing flow. Due to their spatial integration problems, NACA intakes are primarily positioned on the fuselage or used as smaller auxiliary intakes.

2.2.1. Scope of Intakes in the Model

Based on the intake aerodynamics discussed previously, all intake components in the model follow the general preliminary parametrisation in figure 2.1. For each intake type, the corresponding preliminary parametrisations shown in chapter B are used to enable parametric studies of their aerodynamic behaviour.

Although multiple intake configurations are reviewed in the literature, only a subset can be implemented within the scope and time constraints of this thesis. Nevertheless, the full ordering of intake types is retained below because it reflects their expected modelling priority and intended long-term integration into the tool. The highest-priority components, those most likely to be implemented in this thesis, appear first:

1. **Scoop intake**: primary intake type for (turbo)propeller aircraft and most relevant for FC cooling.
2. **Exit**: required to model full in-outflow aerodynamic behaviour of the duct system.
3. **Leading-edge intake**: aerodynamically favourable when geometric constraints allow specifically for FC-powered aircraft.
4. **Ring intake**: commonly used in propeller installations with low external drag where inflow requirements are minimal.
5. **Underslung intake**: less common but aerodynamically relevant for certain layouts.
6. **Flush intake**: generally unsuitable for FC systems due to integration limitations.

Only the scoop intake is included in the present implementation of the tool. The remaining configurations are kept in the ordered list to document the intended future development path and to ensure the framework remains extensible. Exclusion of a component in the current version does not imply scientific irrelevance. It merely reflects the scope of the version of the tool delivered in this thesis. Exact recommendations for extending the tool with additional intake types are provided in section 9.2.4 where the preliminary parametrisations based on literature are provided in chapter B.

2.3. Propeller and Nacelle Integration

The use of electrical propulsion in aircraft adds high flexibility in moving power around in the aircraft, leading to various innovative propulsive concepts, like distributed propulsion and BL-ingestion [8]. These propulsion system contains either a propeller or a (ducted) fan leading to the importance of propeller-wing-nacelle interactions scoped to tractor configuration.

Propeller-Wing-Nacelle Interactions

A propeller can be modelled as an actuator disk generating a helical slipstream with increased axial and tangential velocities, higher total pressure, significant vorticity and helicity, and a slight contraction of the wake. This non-uniform slipstream strongly affects and is affected by its installation on a wing-nacelle system. *Upstream*, the nacelle accelerates the inflow near the blade roots (blockage) and wing upwash induces an effective incidence on the propeller, modifying blade loading. *Downstream*, the increased dynamic pressure enhances lift behind the propeller—particularly near $3/4R$—while swirl introduces asymmetric upwash and downwash, altering both lift and induced drag distributions. The

slipstream also increases friction drag, promotes turbulent BL development, and can degrade intake pressure recovery due to non-uniform forebody flow, especially when blade roots sit ahead of the inlet. Finally, the altered wake and cross-flows influence longitudinal, lateral, and directional stability, although stability effects fall outside the scope of this thesis.

Propeller Position and Nacelle Integration

The aerodynamic interaction between a propeller slipstream and the wing is strongly governed by propeller placement and orientation of which a sketch is shown in figure 2.3. The axial distance controls how fully the slipstream develops before intersecting the wing. The vertical position modifies the local effective angle of attack and axial velocity through upwash, with high-mounted propellers experiencing greater mass flow and potentially smaller, more efficient diameters, while low-mounted propellers suffer the opposite trend. Because upwash inclines the incoming flow, the propeller may be tilted by an angle to restore axial inflow, though this alters the thrust line and slipstream alignment, affecting interference loads. Additionally, nacelle sizing and curvature near the spinner influence blockage and installation penalties, and tilting the intake plane toward the propeller can improve pressure recovery without increasing drag.

Spinner Design Considerations

A propeller spinner functions as an aerodynamic fairing over the hub, reducing drag from the hub itself and from interference with the propeller and nacelle. Its geometry is primarily defined by diameter and length, which must balance minimal wetted area with sufficient internal volume for the hub and a smooth transition to the nacelle. Excessive curvature increases adverse pressure gradients and BL separation, harming pressure recovery in the downstream intake. Experimental and CFD studies [29, 30] show that smooth, moderately slender shapes (typically elliptical or conical) provide the most favourable performance. Additionally, applying area-ruling near the blade roots by introducing shallow dimples reduces blockage and improves propeller efficiency, particularly at higher Mach numbers. An overview of key spinner parameters is illustrated in figure 2.4.

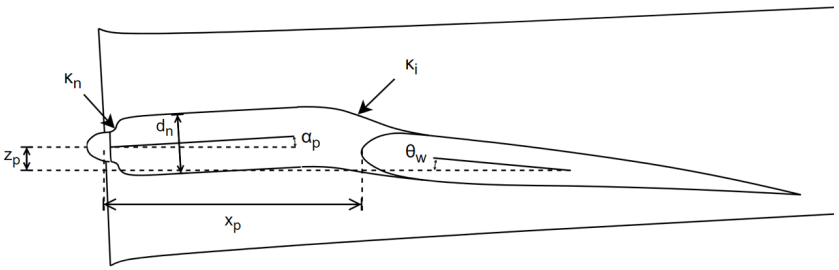


Figure 2.3: Schematic side view of a propeller and nacelle integrated into the wing with a preliminary parametrisation.

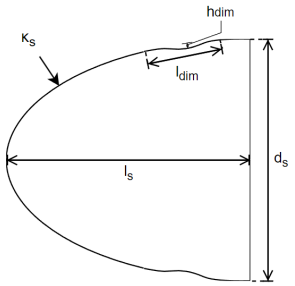


Figure 2.4: Schematic side view of a spinner with area-ruling.

2.3.1. Scope of Propeller and Nacelle Integration in the Model

The tool will include a nacelle integrated into the wing, capable of housing FC subsystem internals as scoped in section 2.1.1. Since nacelle sizing requirements for FC systems may lead to substantially smaller nacelles compared to conventional turbofan engines, the nacelle-wing integration will include blending capabilities to enable research into more aerodynamically attractive designs. Due to the novelty of this concept, little research on this concept or its potential has been conducted.

For completeness of external aerodynamics of the nacelle, a spinner will also be included into the tool. Propeller blade design and modelling of the propeller disk is left out of scope for this thesis and thus the tool, since that is an extensive and complex topic on its own. However, subsequent versions of the tool could integrate a proper propeller disk and blades to enable parametric studies of the propeller and its interactions with full system.

3

Methodology and Requirements

A well-defined modelling methodology is essential for developing the tool. This chapter first compares two parametric approaches, Knowledge-Based Engineering (KBE) and augmented Computer-Aided Design (CADx), to identify the most suitable framework for the tool's development. Based on this methodological foundation and the research topic defined in section 1.4, a structured set of requirements is derived using the MBSE-for-KBE approach introduced in section 3.2. Finally, section 3.3 presents an overview of the complete requirements model, which forms the basis for the tool's implementation and the architectural developments discussed in the subsequent chapters.

3.1. Parametric Modelling Methodology

Parametric modelling captures design intent by expressing knowledge through parameters and rules, enabling automated geometry generation with variation of design options. In contrast, direct modelling relies on manual manipulation of geometry and therefore produces only single, fixed designs. By formalising geometric rules into parametrisations, large design spaces can be explored efficiently through changes in input values. This section focuses on two principal parametric approaches (KBE and CADx) compares their capabilities, and concludes with the selected modelling methodology and software environment for this thesis.

3.1.1. Knowledge-Based Engineering

KBE is defined according to La Rocca [31] as "engineering using product and process knowledge that has been captured and stored in dedicated software applications, to enable its direct exploitation and reuse in the design of new products and variants." The main principle of KBE is to capture engineering design logic into a centralized knowledge system allowing for the automation of repetitive processes while supporting multi-disciplinary design. The capturing of design intent enables the automation of routine engineering tasks, ultimately leading to significant reduction in time and costs of product development. The reduction in time opens the door for more time invested in innovative engineering tasks, instead of the repetitive work which take up to 80% of the design process [32].

3.1.2. KBE and augmented CAD

The first KBE systems were created due to the lack of functionalities and capabilities of the then-current CAD software. As an answer to this, many CAD systems now employ also means of parametrisation by integrating user-defined macros, parametrized geometries, and more. These new extended CAD systems are in this thesis refered to as augmented CAD (CADx) systems.

Using augmented CAD

When there is a need for rapid and detailed geometry generation and manipulation, CAD systems are often more suitable than KBE systems [33]. Verhagen et al. [34] demonstrated that KBE applications are less effective for products undergoing frequent fundamental changes, where designs demand quick and intensive geometric adjustments and major configuration iterations. In such scenarios, design

knowledge and intent is constantly changing and revised leading to CAD systems providing a distinct advantage.

Additionally, CAD systems are preferred for straightforward design tasks [34], particularly when single-purpose modelling is sufficient [35]. They also support a ”what you see is what you get” modelling approach, offering users an intuitive and immediate visual representation of their designs. In contrast, KBE systems often involve abstract and complex modelling processes, which require knowledge engineers with advanced software engineering expertise [35].

Using KBE

When the primary modelling objective is to capture the design intent, KBE is the preferred solution [33]. Unlike CAD programs, which output data models that capture the result, KBE systems deliver a comprehensive product model [32]. Chapman and Pinfold [32] also highlighted that the automation capabilities of KBE enable the evaluation of a wide range of existing and novel designs. This capability is challenging to achieve at scale with CAD systems due to their significant overhead when dealing with parametrisation, mitigated by KBE systems [36].

An additional advantage of KBE is its exceptional integration flexibility. The communication rules employed by KBE systems facilitate seamless interoperability with other tools. By comparison, integrating tools with a CAD software solution often requires extensive and laborious interaction with its Application Programming Interface (API) [36]. This flexibility positions KBE systems as excellent candidates for multidisciplinary design optimization (MDO) applications [33]. Finally, emerging trends suggest significant potential for the application of KBE within web frameworks, further expanding its usability and reach [35, 37].

3.1.3. Tools and Software

Several KBE platforms exist in industry, including Adaptive Modelling Language[1], General-purpose Declarative Language[2], and ParaPy[3]. ParaPy, developed at TU Delft and founded in 2016, is the most recent and actively maintained of these tools. It provides Python-based libraries for geometry generation, meshing, and automated aerodynamic and structural analysis [38]. Its geometry engine is built on the open-source Open Cascade (OCC) kernel[4], enabling the creation of custom geometric extensions. Since TU Delft already holds a comprehensive license with ParaPy B.V., selecting ParaPy as the KBE environment incurs no additional costs for this thesis.

Augmented CAD frameworks such as Siemens NX KnowledgeFusion, Autodesk Intent, and Dassault Systèmes CATIA V5 Knowledgeware also support rule-based modelling. However, these systems operate strictly within traditional CAD environments, making the captured knowledge geometry-centric and limiting integration with non-geometric data or multidisciplinary workflows [33]. Their use may also require new licensing agreements with external vendors, further reducing their suitability for this project.

3.1.4. Selection of Modelling Methodology and Tool

As stated in section 1.4.2, the objective of this thesis is to develop an extendable tool capable of generating external geometries for WNI integrations representative of FC-powered aircraft.

The geometries targeted by the tool are inherently complex, featuring smooth surface transitions, blended integrations, and complex parameter relationships, as illustrated by the example configurations in chapter 2. Consequently, the modelling environment must support advanced geometric construction methods and precise surface control. CADx systems provide these capabilities natively and are therefore considered a low-risk option in terms of geometric robustness. However, CADx environments are typically rigid, offering limited flexibility for process automation, multidisciplinary coupling, and parametrisation capabilities.

KBE, by contrast, offers a development paradigm that directly supports the aims of this research. The automation of geometry generation within a KBE framework eliminates repetitive modelling tasks and

[1] https://www.technosoft.com/application-software/adaptive-modeling-language/
[2] https://genworks.com/
[3] https://parapy.nl/
[4] https://dev.opencascade.org/doc/overview/html/

enables researchers to concentrate on concept exploration and aerodynamic innovation. Furthermore, KBE environments can accommodate highly scalable and interconnected parametrizations, which are essential for capturing the aerodynamic and geometric complexities identified in chapter 2. Finally, KBE models are inherently modular and interoperable, facilitating integration with external analysis, optimization, and design workflows, essential for systematic research on propulsion integration and FC aircraft technology.

Based on these considerations, KBE is selected as the modelling methodology, with ParaPy chosen as the implementation platform. Although the geometric capabilities of ParaPy may initially appear less mature than those of commercial CADx systems, the platform's open and extensible architecture, combined with its underlying open-source geometry kernel OCC, mitigates this risk. Any required geometric enhancements can be incorporated within this work or future developments by extending the ParaPy platform or the underlying OCC kernel, ensuring that the chosen approach remains both robust and adaptable to evolving research needs.

3.2. MBSE-for-KBE

Since KBE is used to develop the parametric tool (section 3.1.4), a structured application-development methodology is required to ensure quality and consistency. A common approach is MOKA [39], but it has notable limitations, including reliance on specialised knowledge engineers, a non-intuitive knowledge model, and the absence of an industry-standard neutral schema [40]. To address these issues, the Model-Based Systems Engineering for Knowledge-Based Engineering (MBSE-for-KBE) approach was introduced [41].

3.2.1. Knowledge Model Ontology

In the MBSE-for-KBE framework, the first step is defining a Knowledge Ontology as an explicit description of the key domain concepts and their relationships [41], illustrated in figure 3.1. This ontology corresponds to the MBSE system view, adapted for KBE application development. It is organised into three packages: the *Requirements* package, which captures stakeholder needs and system requirements, the *Process* package, representing system behaviour, and the *Product* package, describing the system structure and its subsystems.

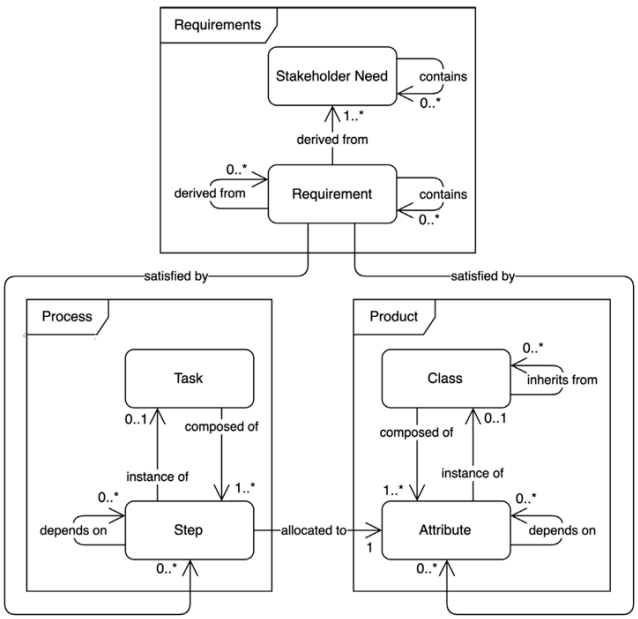


Figure 3.1: Definition of the Knowledge Ontology for KBE applications [41]

This thesis focuses primarily on the *Requirements* package, clarification for this choice is made in section 3.2.2. The Requirements package consists of a Stakeholder Need package and a System

Requirement package. Stakeholder Needs capture external, high-level expectations, while System Requirements formalise the technical constraints derived from those needs, as shown by the *derived from* relation in figure 3.1.

Three requirement types are used: *Physical Requirements*, *Performance Requirements*, and *Component Requirements*. Their relationship to the standard SysML *Requirement* and *Need* stereotypes is shown in figure 3.2.

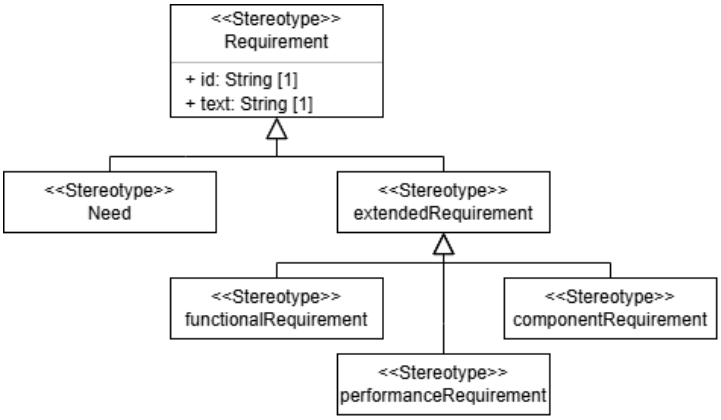


Figure 3.2: Requirement stereotypes from SysML 1.0

3.2.2. Scope of MBSE-for-KBE in the Tool

MBSE-for-KBE is a very promising framework for creating KBE applications, as it allows for a more formalized and standardized method of capturing and reusing knowledge. However MBSE relies heavily on capturing knowledge in digital models. At this stage, knowledge on propulsion integration for FC aircraft technology is very limited as indicated in the problem statements in section 1.4. For that reason it is very difficult to create extensive knowledge models, particularly for the Process and Product package. In conclusion, MBSE-for-KBE is a very promising framework but will not be used in full for the design and development of the tool in the thesis.

However, one particular aspect of it will be used and that is the Requirements package. This aspect provides a systematic process of defining the various desired functionalities of the tool and expresses them in terms of concrete requirements. This initial step can carried out and will provide a solid basis for development and tool architecture, even if knowledge is limited. Before development of the tool, a full Requirements package, is made according to the Knowledge Ontology in figure 3.1.

3.3. Requirements Package

The Requirements package consists of Needs and System Requirements, as shown in figure 3.3. Needs define the functionality of the tool in terms of its interaction with the external environment, while System Requirements specify the internal implementation requirements that satisfy these needs and thereby realise the desired functionality [42]. The Needs package consists of two sub packages, namely the Internal Needs and External Needs. In this thesis, the philosophy is that Internal Needs are functionalities directly related to research topic, hypotheses and research questions presented in section 1.4. This means that the Internal Needs are coming from inside the thesis own research and therefore called Internal, highlighted by the *refined by* link in figure 1.4. On the contrary, External Needs are any functionality not directly related to any research and originate from outside the thesis *Research Design*, therefore called External.

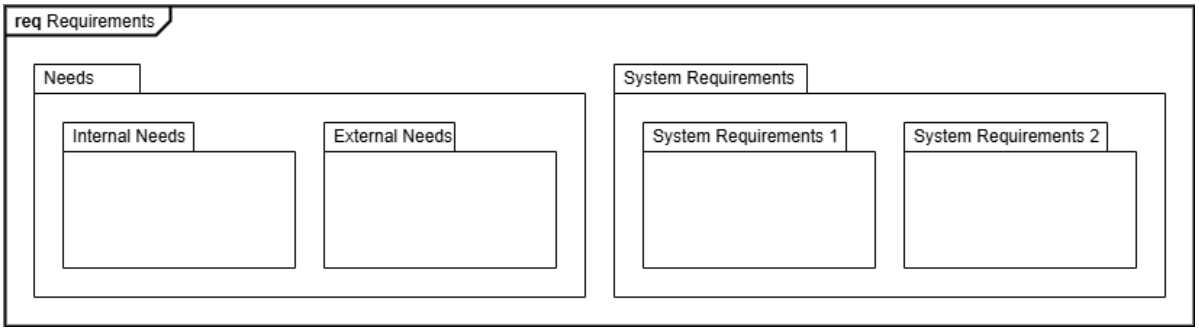


Figure 3.3: Overview of entire requirements diagram with needs and requirements package structure

The Internal Needs are derived directly from the two hypotheses introduced in section 1.4.3. Consequently, these hypotheses serve as the two top-level Internal Needs from which all subsequent Needs originate. Lower-level Needs further decompose these into more specific requirements through containment or derivation. As all problem statements and research questions are inherently linked to the hypotheses, fulfilling these two top-level Needs ultimately satisfies the research objective. Therefore, if all lower-level needs are met, it follows that the top-level Needs, and thus the hypotheses, are also fulfilled. External Needs originate from stakeholders rather than from the literature and are therefore not directly tied to the research topic in section 1.4. In this project, the primary stakeholders are TU Delft and the author.

Building on the Needs, the corresponding Requirements are defined within the System Requirements package and split into two packages System Requirements 1 and 2, directly related to Hypothesis 1 and Hypothesis 2 respectively. Whereas the Needs describe the intended functionality of the tool, the Requirements specify the internal, or white-box, behaviour and implementation details. These represent more concrete and verifiable definitions of the tool's capabilities, each derived from a lower-level Need. Following the same rationale as for the Needs, the Requirements are further decomposed into more specific sub-requirements through containment and derivation. Consequently, satisfying all low-level Requirements ensures that the Needs are met, thereby confirming both hypotheses and the research objective. The low-level Requirements thus form the foundation for the tool's development.

Finally, although the Needs and System Requirements provide the formal foundation for the tool's development, their full detail is not required for understanding the remainder of this thesis. For clarity and brevity, the complete requirement set, including SysML diagrams, full listings, and their assessments, is therefore provided in chapter E.

4

Parametric Model

Building on the system requirements defined in chapter 3, this chapter introduces WIN-Gen (Wing-Intake-Nacelle Generator), an extendable parametric model capable of generating simulation-ready geometry for WNI configurations for FC-powered aircraft.

Section 4.1 introduces the overall architecture, followed by section 4.2, which presents the concept of *parametrisation as composition* for constructing complex and replaceable parametrisations. Generic capability modules are discussed in section 4.3, and key geometric operations are introduced in section 4.4 and section 4.5.

4.1. Model Architecture

The primary function of the WIN-Gen is to generate external geometries of WNI systems for FC-powered aircraft. The `SystemAssembly` class forms the core of the model, encapsulating all subsystems, integration routines, and general tool functionality. The full `SystemAssembly` including components is shown in figure 4.1 of which a simplified Unified Modelling Language (UML) overview is shown in figure 4.2, where the final geometry is represented by the `solid` property, Only the high-level structure is shown as detailed component logic is discussed in the following chapters.

The assembly includes a wing as instance of `LiftingSurface` (chapter 5), an single `Nacelle` instance (chapter 6) with internal subsystems and spinner, and an intake of `Scoop` type (chapter 7). Due to time constraints, only the scoop intake configuration from section 2.2.1 is implemented. As required by SR-4.3 - Component Control, each component can be individually enabled or disabled via Boolean parameters such as `has_wing`, `has_nacelle`, and `has_intake`.

All input parameters for the components and integrations are stored in a unified `data` container, composed of multiple concrete subclasses of `Descriptor` objects. The `Descriptor` class (section 4.2.1) serves as a structured data container designed to systematically collect and manage all input parameters for a specific component. Each model component has a corresponding `Descriptor` subclass containing parameters relevant to the parametrisation of that element. The centralised collection of all `Descriptor` objects is motivated by SR-6.2-D8 - Parameter Collection, as users interact with the model solely through the unified `data` interface.

In addition to the individual components, the model incorporates several integration classes responsible for combining the resulting individual geometric entities of each component into a single configuration. These integration classes perform the necessary blending, trimming, and boolean operations to yield an integrated and aerodynamically consistent system geometry. The model architecture also includes multiple capability modules (section 4.3) that perform analytical and export functions for both the individual components and the full assembly.

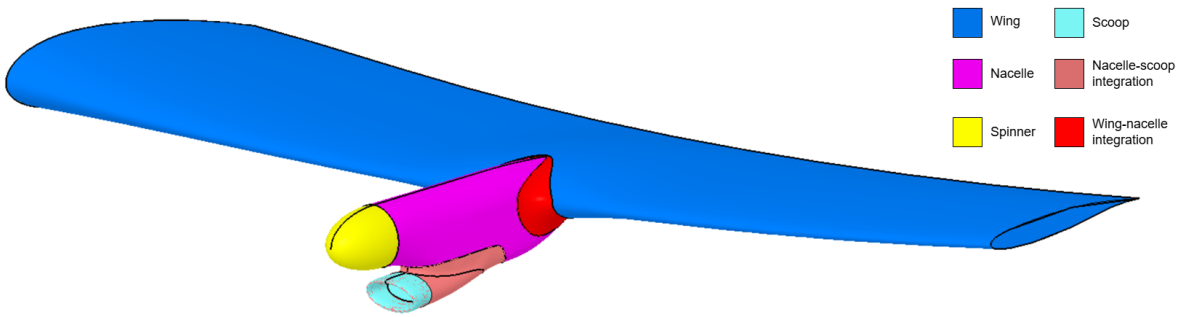


Figure 4.1: Generated OML of the`SystemAssembly` of *baseline design* with each component coloured.

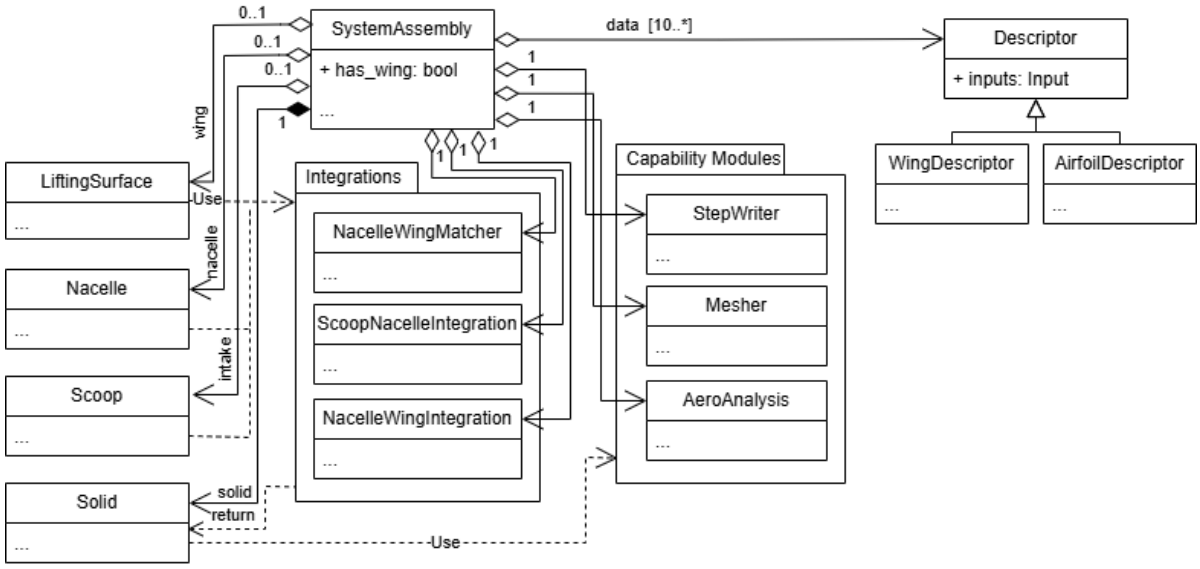


Figure 4.2: Schematic UML diagram of `SystemAssembly`

Modelling Flow

A typical `SystemAssembly` workflow is shown in figure 4.3, illustrating how the tool-integrated logic[1] interacts with a designer's process to progressively generate system geometry from individual components via the integration routines shown in figure 4.2. Rectangular data blocks represent `Descriptor` objects containing design parameters where only a subset is shown for clarity.

[1] 'Tool-integrated logic' refers to the automatic design-generation and export workflow embedded in the tool.

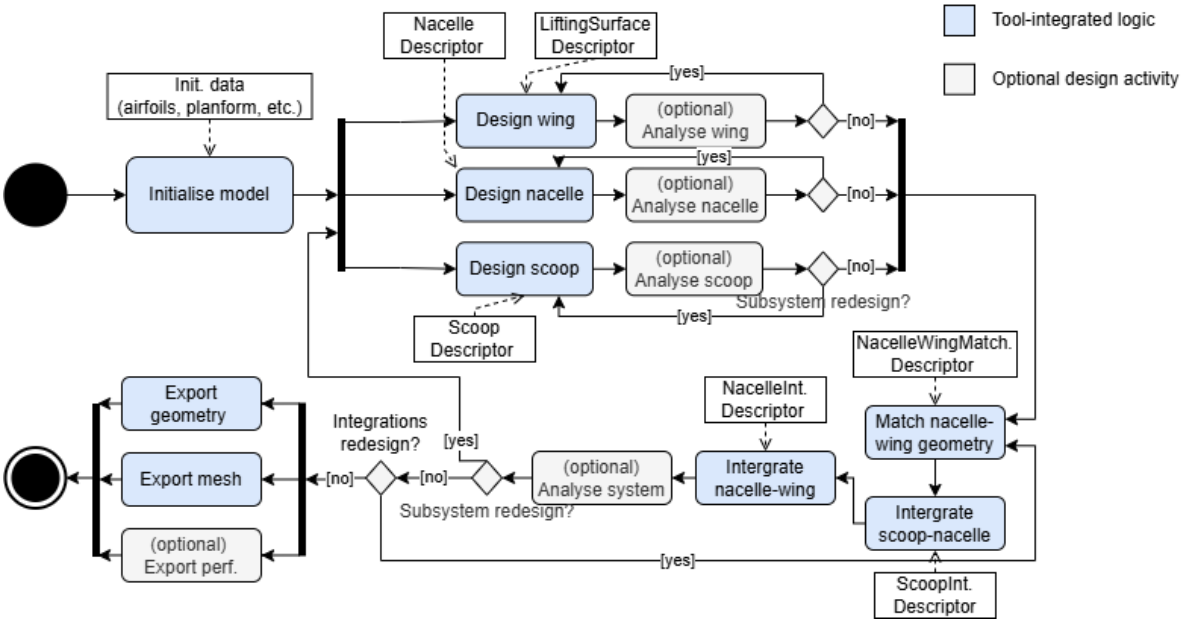


Figure 4.3: Flow diagram of a typical design flow of `SystemAssembly`, where the blue processes are tool-integrated processes and grey are optional processes for an example design workflow.

The workflow starts with model initialisation, where all `Descriptor` objects are instantiated and supplied to the `SystemAssembly`. A dedicated initialisation module has not been implemented. CPACS-based and reference-geometry initialisation were not implemented due to time and complexity constraints[2]. However, coordinate-based initialisation is used for airfoils (section 5.2.4) and nacelle cross-sections (section 6.4.2).

After initialisation, components are generated and analysed independently using their associated `Descriptor` objects, with aerodynamic evaluation available through the integrated Capability Modules (section 4.3). Iterative redesign based on aerodynamic feedback can be performed directly within the tool. Clean integration between components begins with *geometrical matching*, which enforces a clean intersection required for subsequent blending. For that reason, integration sequence starts by geometrically matching the wing and nacelle (section 6.7.2), integrating the scoop intake into the 'matched' nacelle (section 7.5), and finally combining this assembly with the wing to produce the complete WNI system geometry (`solid` in figure 4.2).

The final geometry can be analysed aerodynamically to obtain first-order performance estimates, refined as needed, and exported as STEP geometry (section 4.3.2), surface meshes in VTK format (section 4.3.3), and aerodynamic data to CSV or Excel (section 4.3.4).

4.2. Parametrisation as Composition

A key architectural principle introduced in this work is *Parametrisation as Composition*, which marks a shift from inheritance-based modelling towards a compositional paradigm. Rather than defining new subclasses for each component variant, parametrisation logic is encapsulated within dedicated classes that are composed into the parent component. This principle directly implements the *Strategy Pattern* [43]. In this behavioural design pattern, a family of algorithms (in this context the parametrisations), are defined and used interchangeably by a client (in this case, the component). The client and the strategies remain independent, allowing each to evolve without knowledge of the other's internal workings.

An example of *Parametrisation as Composition* applied to an airfoil is presented in figure 4.4. The airfoil defines two parametrisations: one governing shape definition through `curve_param`, and another managing placement and scaling through `trans_param`. Two curve parametrisations are implemented,

[2]Geometry-based initialisation would require optimisation-based matching procedures beyond the scope of this thesis and are recommendation for future works suggested in section 9.2.3.

the Class-Shape Transformation (CST; section 5.2.3) and NACA4-series, while two transformation parametrisations are provided: CPACS-based placement and the airfoil-on-rails method (section 5.2.5).

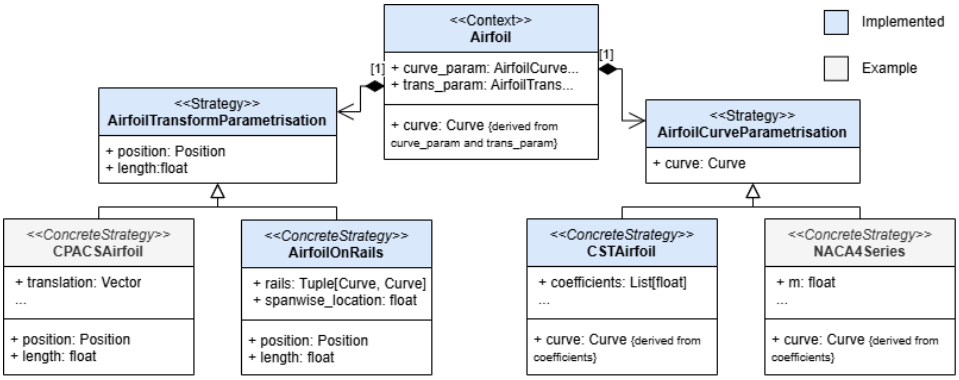


Figure 4.4: UML representation of the Parametrisation-as-Composition architecture applied to an airfoil. Implemented classes are shown in blue and not implemented classes in grey. The design realises the Strategy Pattern, with classes annotated by their respective pattern roles (Context, Strategy, ConcreteStrategy).

In contrast, a traditional inheritance-based parametrisation approach results in tight coupling between a component and its parametrisation, as illustrated in figure 4.5. Each new parametrisation must be implemented as a subclass of the base component, overriding attributes to define the new behaviour. Furthermore, when multiple parametrisations must be combined (for instance, shape and placement) double inheritance becomes necessary, leading to an explosion of concrete subclasses and reduced maintainability.

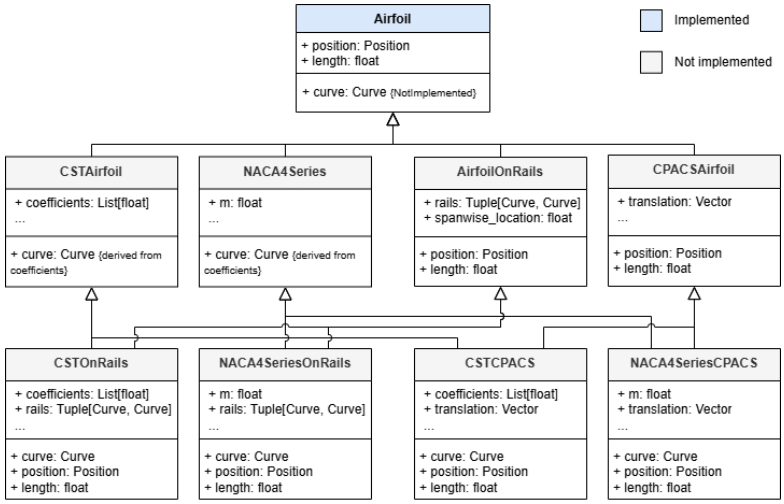


Figure 4.5: UML representation of the inheritance-based parametrisation approach for an airfoil. Implemented classes are shown in blue and example classes in grey.

This approach offers several advantages over traditional inheritance-based parametrisation:

- **Simplified class hierarchy:** The overall class structure is significantly simpler, as illustrated by comparing figure 4.4 and figure 4.5. Using inheritance to combine parametrisations would result in $n \times m$ concrete subclasses and $n \times m + n$ total classes,[3] whereas composition requires only m concrete classes and $m + n$ in total. The inheritance-based approach also introduces numerous between-class dependencies, visible in the dense inheritance network of figure 4.5. Given that

[3] n = number of parametrisation purposes (abstract strategies); m = number of parametrisations (concrete strategies), assuming an equal number of parametrisations for each purpose.

many parametrisations will likely be developed in this thesis and in subsequent work, maintaining a simple class hierarchy is highly desirable.

- **Enhanced modularity:** Each class in the compositional architecture is small and has a single responsibility, improving modularity and clarity. This design promotes flexibility and aligns with SR-9 - Component Flexibility, ensuring that components are not constrained by rigid internal logic. Conversely, large multi-purpose classes are more complex to maintain and harder to understand.
- **Ease of use:** Users can easily select desired parametrisations from available options and assign them to a component, rather than navigating an extensive list of combined subclasses.
- **Simplified extendability:** Extending functionality becomes straightforward, as the abstract parametrisation classes clearly define the rules for implementing new parametrisations. Developers can add or replace parametrisations without modifying the component or other strategies and parametrisations. By contrast, subclass-based extension often requires deep understanding of the parent class and risks breaking existing functionality by overriding incorrect attributes.
- **Improved knowledge reuse:** Because knowledge is encapsulated within modular parametrisation classes, design logic is clearly traceable and reusable. This supports the systematic knowledge management goals set out in Hypothesis 2.

Together, these advantages make *Parametrisation as Composition* a robust foundation for the future development of the tool. However, this architecture also introduces challenges, such as managing dependencies between different parametrisations and ensuring consistent access to parent attributes or external data, as fully instantiated parametrisations are fed into parent components. These limitations and their mitigations are discussed in section 4.2.1.

4.2.1. Descriptors and Construction

For *Parametrisation as Composition* to be generally applicable, parametrisations must accept both design variables (user inputs) and context variables (external data provided by a parent or environment). In practice, an airfoil placement model such as `AirfoilOnRails` depends on a spanwise location (design variable) and on leading/trailing rails (context), where the rails are computed upstream (e.g., by a wing). This motivates a separation between user-editable design data and external context in parametrisation inputs.

This separation is realised through the use of `Descriptor` objects. A descriptor encapsulates the input variables for a specific family of parametrisations and defines a method, `to_result(context)`, which instantiates the parametrisation using its internal input variables and injecting and the required external contextual data. Conceptually, a descriptor implements the Abstract Factory software design pattern [44] specialised in creating a parametrisation family within a particular context. Consequently, parametrisations remain fully interchangeable, following the Strategy pattern as explained in previous section, while the descriptors manage context binding and creation, consistent with the Abstract Factory pattern.

An implementation of this `Descriptor` concept is illustrated in figure 4.6. In this example, the `AirfoilTransformParametrisation` (the Abstract Product) is constructed by a `TransformDescriptor` (the Abstract Factory), using the wing as its context (the Client). For each parametrisation, representing a Concrete Strategy as introduced in section 4.2, a corresponding Concrete Descriptor (Concrete Factory) implements the `to_result(context)` method. These concrete descriptors are supplied to the wing, which invokes their creation methods to generate each parametrisation.

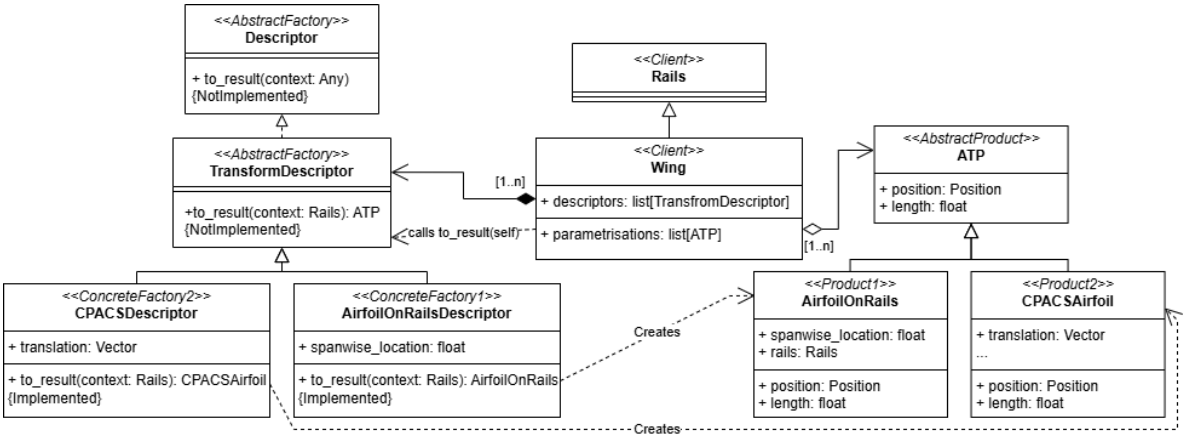


Figure 4.6: UML representation of `TransformDescriptor`, a descriptor capable of creating an `AirfoilTransformParametrisation` by injecting design data and context. Each class is annotated with its respective pattern role.

Figure 4.6 clearly demonstrates the separation between defining parametrisations and applying them within a contextual system. This separation provides a robust structure for extending parametrisations with additional logic and for managing dependencies of varying complexity. Furthermore, it makes the creation of new parametrisations highly intuitive: all dependencies, whether user-defined design variables or externally supplied context, can be declared directly in the parametrisation input definition. The corresponding descriptor then handles the separation between internal inputs and external context, ensuring clean and controlled instantiation.

It should be noted that the example shown in figure 4.6 is conceptual and not directly implemented in the current version of the model. In practice, the airfoil and wing implementation is more complex: `TransformDescriptor` objects are not passed directly to the wing, but rather to an intermediate airfoil descriptor that aggregates multiple parametrisations, such as shape definitions.[4]

4.3. Capability Modules

As noted in the section 4.1, the system assembly contains various capability modules (CMs). Similar as to how La Rocca [45] introduced a CM in his Multi-Model Generator (MMG) as ”the capability of the MMG to automate the generation of models to support multidisciplinary analysis of aircraft”, this tool includes similar CMs, capable of import of external data, full geometry export, automatic meshing and direct preliminary aerodynamic assessment.

4.3.1. Data Import

As explained in section 4.1, the only initialisation capability of the tool is the importing of coordinated data, specifically for curves, airfoils and (nacelle) cross-sections. Two-dimensional coordinate data is read from a simple coordinate data file and subsequently processed to fit point requirements for curves, airfoils and nacelle cross-sections. Through these processed points a curve can be interpolated or approximated to get the final geometrical curve.

As illustrated in figure 4.7, the data import workflow starts by loading a coordinate file (`.dat`) containing one or more groups of coordinates. These groups are chained by matching endpoints to form a continuous point sequence. Depending on the intended curve type, the points may be further processed to satisfy geometric conventions before being interpolated or approximated into the final curve geometry.

[4] Further details on the implementation of wings and airfoils are provided in chapter 5.

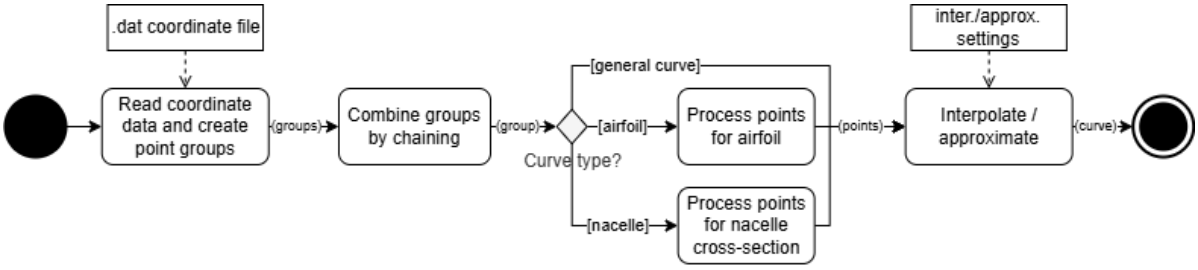


Figure 4.7: Coordinate Data Import Overview

For an airfoil, the points are processed as follows:

1. Close the trailing edge.
2. Reorder the points from the trailing edge (TE) to the leading edge (LE), then back to the TE, with the upper surface preceding the lower.
3. Normalise the airfoil such that the LE and TE correspond to coordinates $(0, 0)$ and $(1, 0)$ respectively.

For a nacelle cross-section, the data must be oriented consistently around a cross-section centre:

1. Identify the crown and keel points[5] (defined at $x = 0$).
2. Order the points counter-clockwise, starting from the crown point.

Following these steps, the processed point data are converted into a geometric curve through either interpolation or approximation, as specified by the user. The resulting curve serves as input to specific parametrisations of model components requiring a generic geometric curve, most notably, the airfoil and nacelle cross-section parametrisations.[6] This functionality enables the tool to construct airfoils and nacelle cross-sections directly from external coordinate data.

However, geometries created in this way are static, since no dynamic, user-modifiable parameters are defined within these parametrisations. To address this limitation, two optimisation algorithms were implemented to infer the CST class and shape coefficients from the imported point data. The details of these optimisation routines and their applications are discussed in chapter 5 and chapter 6.

4.3.2. Geometry Export Module

In accordance with SR-1.2-D2 - STEP Export, the tool exports the external geometry to STEP format. As the model is hierarchically structured, this hierarchy is preserved during export, capturing physical model structure and logical relationships thereby improving traceability of both geometric operations and design intent. The exported hierarchy includes a single *root geometry* and any associated *helper geometry*, with only the top-level object exporting the final external root shape and every sub-level object always exporting *helper geometry*.

Each geometry-exportable object in the model contains three key elements governing the export process:

- **Root geometry**: The external or physical geometry representing the actual component surface. This geometry is exported only for the root component of the model, ensuring that the exported STEP file correctly represents the final outer mold line of the system.
- **Helper geometry**: Supporting geometrical entities used in the construction of the root geometry, such as reference curves, section planes, or guiding curves. Including these helper geometries in the export allows for a deeper understanding of the generative logic of each component, aiding in downstream design modification and anaylses.

[5] Exact definitions of the crown and keel points are provided in chapter 6.

[6] A complete listing of all parametrisations associated with these components is provided in subsequent chapters.

- **Child export references**: A list of subcomponents references whose helper geometries must also be exported recursively. This ensures that the full model hierarchy, including all contributing geometric elements, is preserved within the exported STEP structure.

Figure 4.8 presents a UML representation of the geometry export architecture, implemented for two arbitrary components: a `LiftingSurface` containing multiple `Airfoil` objects. This architecture is reused throughout the model for all objects that are geometry -exportable. Both the `LiftingSurface` and `Airfoil` classes inherit from `GeometryExportBase`, implementing the three core attributes governing geometry export: root geometry, helper geometry, and child export references.

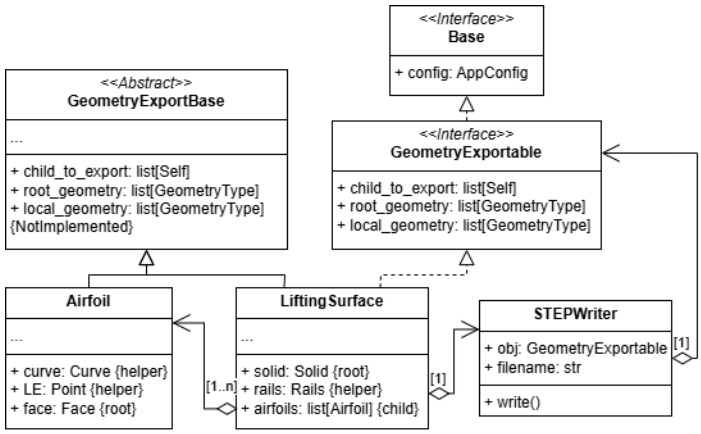


Figure 4.8: UML representation of the `GeometryExportBase` class implemented by both the `Airfoil` and `LiftingSurface`. The attributes are annotated as root, helper, or child. This diagram serves as a conceptual example; in practice, only references to geometrical attributes are implemented rather than direct instances.

The `LiftingSurface` additionally contains a `STEPWriter`, responsible for writing the complete model hierarchy to STEP format. When `write()` is invoked, the lifting surface is treated as the root object, resulting in the export of both its root and helper geometry. Since all `Airfoil` instances are annotated as geometrical children to export, each is recursively included in the STEP file as a child entity with its respective helper geometry, but without exporting its root geometry. For this recursive export to function correctly, all geometrical children must inherit from `GeometryExportBase` as well.

Finally, STEP export settings such as schema, unit system, and project metadata are defined globally in the `AppConfig` class. All root-exportable objects, i.e. those implementing `GeometryExportable`, must contain this configuration to ensure consistent and standardised output across the model. Subsequent chapters will discuss how this architecture is applied to specific components, detailing which geometries are exported and how the module behaves in practice.

STEP Export Algorithm

In order to export the full model hierarchy while distinguishing between the two geometry export purposes, this work introduces a new STEP export algorithm. The built-in ParaPy STEP export functionality cannot make this geometrical distinction and offers limited control over which objects are included in the exported model hierarchy.

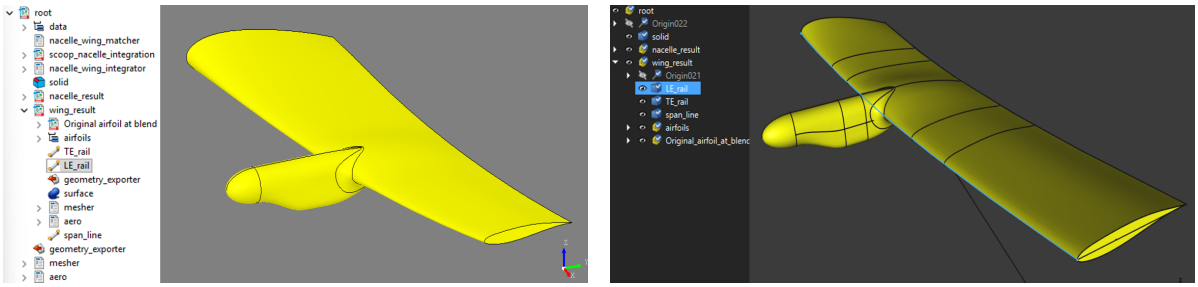
For each component in the model tree, an OCC *compound* is created. Both individual shapes and child components, represented as additional compounds, are added to this structure. When a component contains a sequence of shapes or subcomponents, these are likewise grouped into a single compound. This procedure of constructing compounds for every component (instances of `GeometryExportBase`) and populating them with their respective geometry and children forms the foundation of the STEP export algorithm detailed in chapter C.

Importing Results

After exporting a system or subsystem to STEP format, the resulting file contains the full model hierarchy and all geometric entities, allowing import into third-party CAD or analysis tools. Figure 4.9 shows this

using a *baseline design*[7] of a wing with nacelle assembly exported from ParaPy and re-imported into FreeCAD[8]. Figure 4.9a shows the assembly in ParaPy's viewer, where only the root geometry (external mold line) is visualised to distinguish it from helper geometry. The export follows the algorithm in figure C.1, executed via the `geometry_exporter`, an instance of `STEPWriter`.

Figure 4.9b shows the same model imported into FreeCAD from the generated `.stp` file. The hierarchy is preserved, including helper geometries such as the LE rail (`LE_rail`). Only the root geometry of the system assembly (`root`) is exported in full including root geometry, while children contribute only helper geometry.



(a) System assembly with nacelle and wing in ParaPy, including the model tree. Only the root geometry (outer mold line) is displayed to highlight the distinction between root and helper geometry.

(b) The same assembly imported into FreeCAD from the STEP file generated by the tool. Helper geometry of the leading-edge rail (`LE_rail`) is highlighted in both the model tree and viewer.

Figure 4.9: Comparison of the baseline design in ParaPy and the corresponding geometry imported into FreeCAD using the geometry export module.

4.3.3. Automatic Meshing

In accordance with SR-5-X-D6.1 - Automatic Meshing, the tool integrates an automatic meshing module to ensure that the generated geometries are compatible with the aerodynamic solver. Beyond enabling solver compatibility, the meshing module also serves as a key verification mechanism for assessing the computational quality of the geometry as further discussed in section 8.1.1.

This module, `ShapeMesher`, adopts an architecture similar to the geometry export system shown in figure 4.8. It can be attached to any meshable component, each of which provides a `shape_to_mesh` attribute representing its B-rep root geometry. The module includes parameters for mesh generation, grouping, local refinement, and produces a final `Mesh` object exportable in multiple formats[9].

A default configuration is provided using an unstructured surface mesh based on the NETGEN algorithm from the SALOME platform.[10] This generates triangular surface meshes with higher element density in regions of high curvature. An example for the baseline design is shown in figure 4.10. Mesh elements respect all topological edges, ensuring clean alignment between mesh boundaries and geometric boundaries, as illustrated in the zoomed wing-nacelle interface.

[7] The *baseline design* is a reference configuration used for development of the tool.

[8] `https://www.freecad.org/`

[9] The meshing algorithms themselves are not developed as part of this work. For implementation details, capabilities, and supported export formats, refer to the ParaPy documentation at `https://parapy.nl/docs/`.

[10] `https://www.salome-platform.org/`

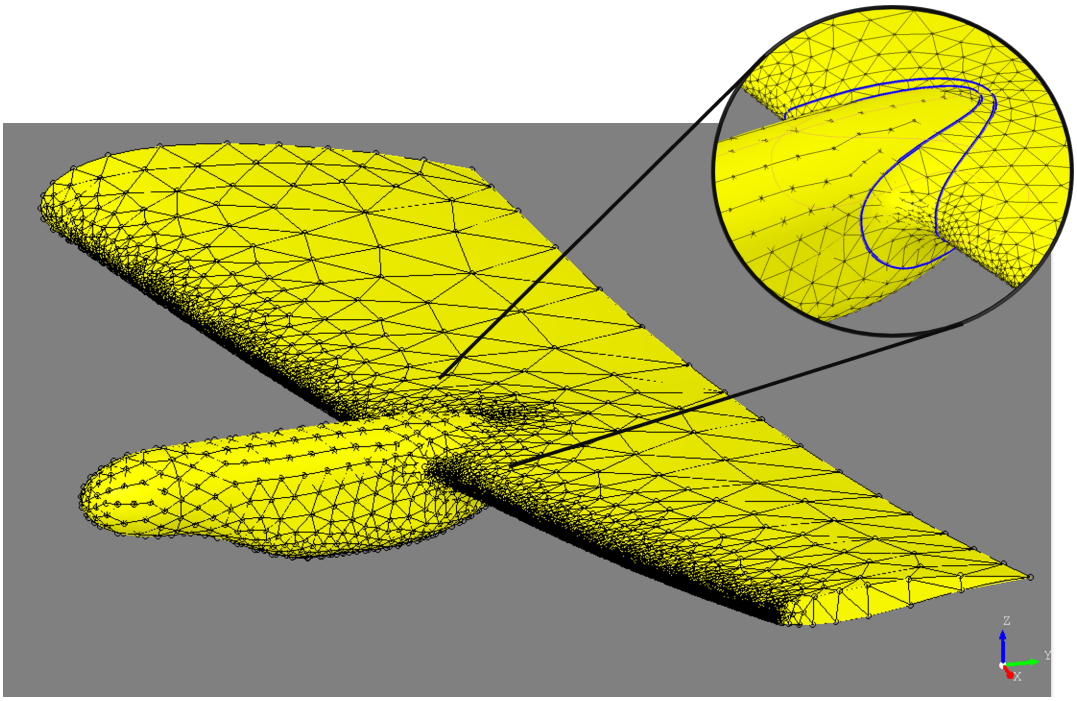


Figure 4.10: Unstructured surface mesh generated for the baseline design using 'Moderate' fineness. The zoomed-in region shows alignment between mesh edges and geometrical boundaries (in blue).

Developing subsystem-specific meshing strategies, refined controls, or assessing aerodynamic mesh sensitivity lies beyond the scope of this thesis. The generated meshes should therefore be regarded as preliminary geometric or visual verification tools rather than aerodynamically validated inputs. Extending this module to provide robust, high-quality meshes for all configurations is a key recommendation for future work.

4.3.4. Simulation Interface and Aerodynamics Module

As stated in SR-5-D5 - Aerodynamic Solver, the aerodynamic solver represents the first and currently only discipline integrated into the tool. The model incorporates a `Simulation` abstraction layer to be easily extendable to accommodate additional disciplines. This abstraction provides shared functionalities that can be readily reused by any future disciplinary modules. The aerodynamic module conforms to this standardised interface and serves as a reference implementation for integrating new disciplines into the tool.

Owing to this structured simulation interface and the tool's modular architecture, the framework can be seamlessly integrated into a wide range of research and design workflows, including MDO environments. This ensures that the tool is not confined to aerodynamic analysis alone, but instead establishes a robust foundation for comprehensive multidisciplinary integration, an essential capability for advancing research on WNI configurations for FC-powered aircraft presented in Hypothesis 2.

Simulation Interface

Identical to the previous capability modules, the `Simulation` interface can be contained by any component within the model. As an abstract interface, it does not impose specific constraints on the containing object (`obj` in figure 4.11). Its primary purpose is to define a `run()` method, which must be implemented by any new concrete discipline to execute a process such as a simulation or analysis. Following execution, the interface stores the resulting raw and unprocessed data either internally, as class attributes, or externally, in designated output files.

Each `Simulation` may include one or more `Parser` objects, as seen in figure 4.11, that translate raw simulation output into structured results, represented as mappings of `IndependentVariable` instances to `Result` values, where each independent variable represents a discrete data point within the simulation

output. Abstract base classes, `DataParser` and `FileParser`, define consistent rules for implementing concrete parsers, ensuring extensibility and interoperability across disciplines. The parsed results can subsequently be processed by dedicated `SimulationExporter` objects, which export the data to tabular formats such as CSV or Excel, with rows representing individual data points and columns corresponding to dependent and independent variables, enabling efficient downstream analysis and reporting.

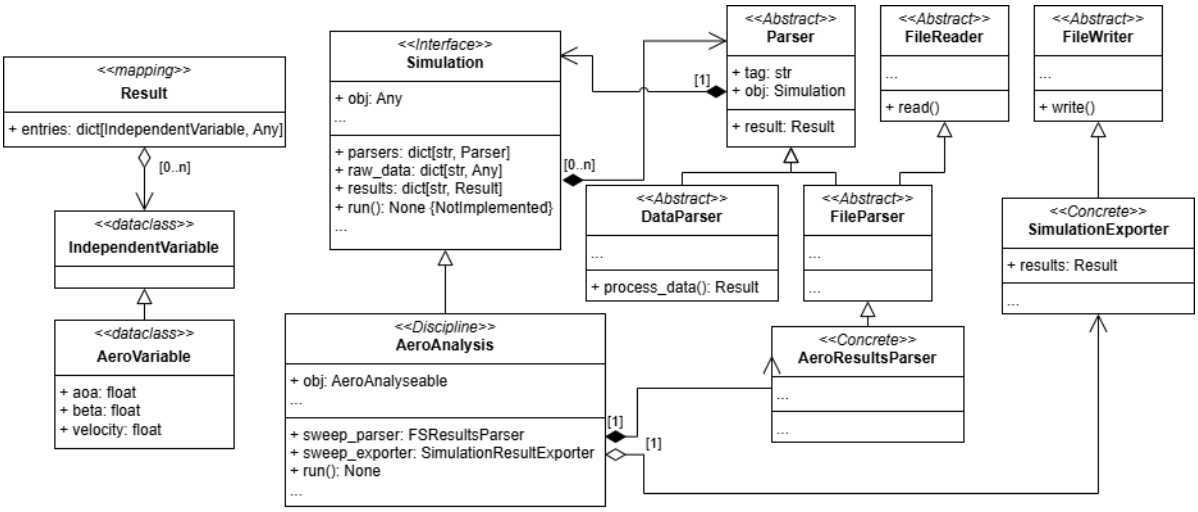


Figure 4.11: Schematic UML diagram of the abstract `Simulation` interface with related objects, implemented for the aerodynamics module.

The modular structure of the simulation framework makes the addition of new disciplines straightforward. The process for integrating a new discipline can be summarised as follows:

1. **Define a new simulation module:** Inherit from the abstract `Simulation` interface and implement the `run()` method to execute the simulation and store its raw output data, either internally or in an external file by using the various file writers.
2. **Define result types:** Extend `IndependentVariable` representing the independent variables and implement a mapping from this variable to results relevant to the new discipline, ensuring consistent data structure and traceability.
3. **Implement a parser(s):** Create a concrete subclasses of `DataParser` or `FileParser` that converts the raw output data into structured results, expressed as this mapping between the newly defined `IndependentVariable` instances and results.
4. **Export results:** Use (and optionally extend) the `SimulationResultExporter` class to export the processed data to standard formats such as Excel or CSV, enabling post-processing or integration into external workflows.

This workflow enables efficient integration of new simulation disciplines while preserving consistency, modularity, and compatibility with existing functionality across the tool.

Solver Choice

A wide range of aerodynamic analysis methods exists, ranging from analytical textbook-based approaches [46–49] to intake-specific formulations [50, 51]. While suitable for early conceptual studies, these methods lack the geometric fidelity required to analyse complex three-dimensional WNI configurations and are therefore unsuitable for this work. Low-fidelity computational solvers based on potential-flow or panel methods provide a better balance between accuracy and efficiency, with examples including AVL[11], VSAERO, and FlightStream. Among these, FlightStream (FS) [52] was selected for its computational efficiency, geometric robustness, and compatibility with the tool's meshing architecture.

[11] `https://web.mit.edu/drela/Public/web/avl/`

FlightStream is a surface-vorticity-based solver compatible with both structured and unstructured surface meshes, directly aligning with the meshing approach used in this tool (section 4.3.3). Its reduced sensitivity to local surface irregularities and geometric discontinuities makes it suitable for analysing complex WNI geometries using relatively coarse meshes [53]. FlightStream has been validated in academic and industrial studies [53–55], supporting its application in this thesis as a low- to mid-fidelity solver capable of supporting both verification and validation[12], and preliminary aerodynamic assessment.

Integration of FlightStream Interface

A specific implementation of the `Simulation` interface is the integrated aerodynamic module, defined as `AeroAnalysis` in figure 4.12. This module provides a direct interface to FS, using an automatically generated script that contains all instructions required for FS to execute a simulation. The script defines, among other aspects, the run case configuration, solver settings, and the paths to the imported mesh files. Once the script is written, FS executes it autonomously, performing the aerodynamic analysis and exporting the results.

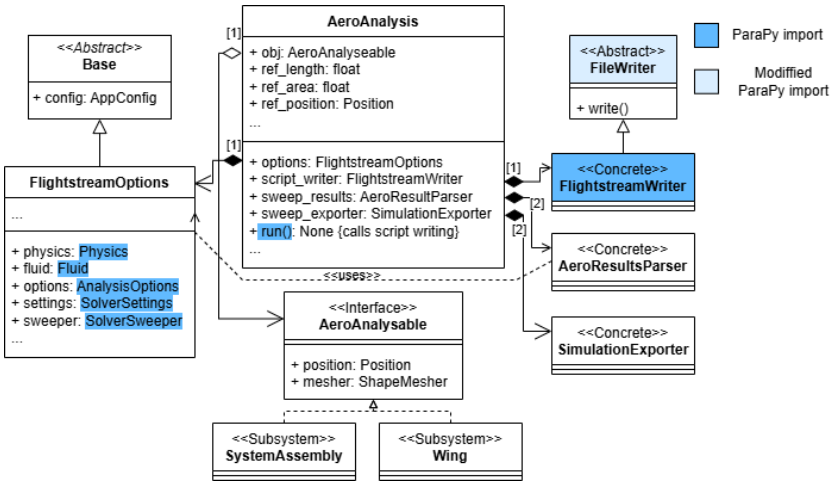


Figure 4.12: Schematic UML diagram of the `AeroAnalysis` class, illustrating its key components and their interaction with the FlightStream interface. The `SystemAssembly` and `Wing` classes are shown as representative analysable objects. Colour coding denotes implementation origin: **dark blue** indicates classes and functionalities directly reused from ParaPy, **light blue** represents elements imported with modifications, and **white** denotes newly created and implemented components.

The FS simulation process as integrated in the tool, is summarised below, with reference to figure 4.12:

1. The `AeroAnalysis` module receives the reference length, area, and position of the geometry to be analysed. The reference position is used to transform the mesh into the FlightStream coordinate system[13].
2. The user specifies solver parameters, including flow conditions, boundary-layer model, and experimental setup, within the `FlightstreamOptions` object.
3. The `run()` method is executed. It exports the surface mesh from the `ShapeMesher` associated with the analysable component, generates the FlightStream input script, and initiates the solver process.
4. FlightStream opens, runs the simulation on the supplied surface mesh, and writes the resulting aerodynamic data to an external file.
5. Dedicated single- and sweep-result parsers import this raw data and convert it into structured mappings of `AeroVariable` instances (see figure 4.11) to aerodynamic coefficients, including C_L, C_{D_i}, C_{D_0}, and relevant moment coefficients.

[12] See chapter 8.

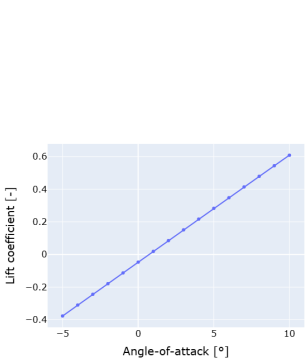
[13] In FlightStream, the *Reference* coordinate system is defined with the x-axis aligned with the freestream direction and the z-axis oriented upwards.

6. Optionally, the `SimulationExporter` may be used to export the processed aerodynamic data to tabular formats such as CSV or Excel for further analysis or documentation.

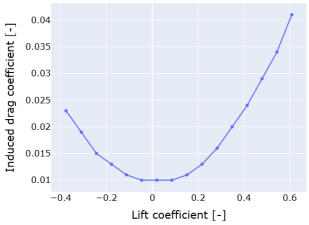
From this process, particular functionalities are already included in the FS interface of ParaPy which are: The FS script writing and all containing options and setting in the `FlightstreamOptions` class. Additionally, no detailed description of the full FS interface including solver settings, options and limitations are given due to the scope of aerodynamic integration in the tool. To elaborate further, no mesh convergence or quality study has been conducted, leading to questionable aerodynamic results even if the limitations and options of the solver are fully addressed. More on the current state of aerodynamic results in section 8.2.3 and recommendation for subsequent full aerodynamic integration study in section 9.2.2.

Illustrative Aerodynamic Results

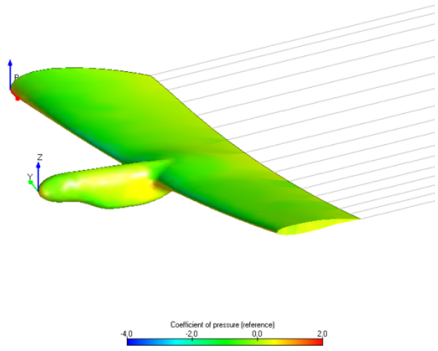
To illustrate the integrated aerodynamic interface, a representative FlightStream (FS) simulation was performed for the *baseline design*, purely to demonstrate the tool's automatic solver integration and post-processing capabilities. The results are not validated and should not be interpreted as physically accurate (see section 8.2.3). An overview of the automatically generated outputs is shown in figure 4.13, including a lift polar generated within the application (figure 4.13a), a drag bucket plotted from exported data to Excel (figure 4.13b), and a qualitative pressure distribution on the 3D model displayed in the FS interface (figure 4.13c). The latter also illustrates the coordinate transformation between the tool and FlightStream, with the *reference* system located at the spinner tip and the tool's global system at the wing root leading edge.



(a) Lift polar automatically generated by the tool.



(b) Drag bucket created from exported data in Excel.



(c) Pressure distribution visualisation in FlightStream, showing the FS *reference* (spinner tip) and tool global (wing root) coordinate systems.

Figure 4.13: Illustrative aerodynamic outputs generated through the integrated FlightStream workflow. The plots and visualisation serve solely as examples of the tool's automated analysis and export capabilities, not as validated aerodynamic results.

Limitations of Aerodynamics Module

As discussed in the preceding sections, the current aerodynamic implementation possesses several key limitations that must be explicitly highlighted:

- **No mesh quality assurance:** As stated in section 4.3.3, implementing meshing lies outside the scope of this work. Consequently, there is no verification or quality assessment of the automatically generated unstructured surface mesh. It is highly likely that aerodynamic results are strongly dependent on mesh quality, a relationship which has not yet been quantified or validated.
- **Lack of validation:** Comprehensive validation of the aerodynamic analyses for the generated WNI configurations is not included within the scope of this thesis. This limitation stems primarily from the absence of mesh quality control and time constraints that prevented the execution of a full validation process. Nevertheless, the tool successfully integrates an automatic aerodynamic solver capable of analysing designs within minutes, demonstrating strong potential for fulfilling IN-1.3.1 - Exploration Support and addressing Problem 2 identified in section 1.4.

- **Intake modelling not supported:** Although FS supports the simulation of intakes (referred to as *inlets* within FS), the current implementation of the FS interface in the tool does not expose functionality for importing or defining intake groups. While the meshing algorithm correctly generates the intake surface, it does not yet classify the mesh into appropriate inlet and outlet regions, nor does it assign the corresponding flow boundary conditions required for intake simulations.

These limitations define the directions for future development and validation efforts, particularly those addressed in chapter 8.

4.4. Gordon Surface

As illustrated by the preliminary parametrisation sketches in chapter 2, the model requires the representation of smooth yet highly curved curves, both for the outlines of the two-dimensional profiles and for the three-dimensional external geometry. These curves also define the cross-sectional shapes of various components[14]. Since the tool focuses on generating the outer mould line, it is evident that surface construction methods capable of handling varying profiles and guiding curves are essential. The algorithm presented in this section forms the fundamental geometrical operation used in this tool.

4.4.1. Previous Work and Rationale

In the *Clean Wing* parametrisation, introduced by Koning [56] and Van den Berg [57], the wing surface was defined by leading- and trailing-edge rails, with airfoils positioned between them. Both the rails and airfoils could be defined using arbitrary curve formulations, which necessitated a surface generation algorithm capable of constructing a surface from these guides and profiles. However, due to technological limitations[15], they were restricted to creating straight lofts between two profiles or general lofts through multiple arbitrary profiles. Consequently, the guiding rails were not incorporated into the final surface generation process. To mitigate this, Koning generated a single loft through a dense sequence of interpolated intermediate airfoils. Although this approach produced acceptable results, it introduced several drawbacks: smooth transitions were formed where geometric kinks were required, and surface accuracy was reduced because the resulting geometry did not follow the guiding rails. Hence, Koning and Van den Berg recommended that future work should focus on implementing a surface generation algorithm that explicitly incorporates both profiles and rails.

The importance of such an algorithm was further demonstrated in the work of Van Luijk [58], who reparametrised a previous outer-wing model of the Flying-V aircraft originally developed by Hillen [59]. Hillen identified that using linear lofts to model different wing sections led to geometric discontinuities, which in turn produced artificial pressure peaks in aerodynamic analyses. To address this, Van Luijk replaced these linear lofts with a more advanced surface generation technique, the Gordon surface algorithm, which allows for the simultaneous use of guiding and profile curves. However, at the time, ParaPy's implementation of the Gordon surface was limited to a maximum of two profiles and two guides, restricting its practical applicability.

Curve-network surface generation algorithms are available in advanced commercial CAD systems, such as the Loft functionality in CATIA V5[16] and Autodesk Fusion[17]. CATIA V5 supports the use of auxiliary (support) surfaces to align profiles and guide curves, enabling continuity control up to G^2, while Autodesk Fusion offers similar tangency and curvature constraints at the first and last loft profiles. However, these implementations are proprietary with their implementation not publicly documented. Within the open-source domain, the only fully available implementation of a complete curve-network interpolation algorithm is the Gordon surface formulation as implemented in the TiGL[18] geometry library, making it a unique reference for research-oriented projects.

[14] Figure 2.2 presents a representative example of such a shape, featuring a parametrised profile (side view) and cross-section (front view).

[15] Koning and Van den Berg employed the earlier KBE platform GDL (`https://genworks.com/`), which lacked several advanced capabilities now available in the more mature ParaPy platform used in this work.

[16] `https://dshelp-embed.3ds.com/2025/english/CATIA_P3/online/CATIAfr_C2/icmugCATIAfrs.htm`

[17] `https://help.autodesk.com/view/fusion360/ENU/?guid=SFC-LOFT`

[18] `https://www.dlr.de/en/sc/research-transfer/software-solutions/tigl`

4.4.2. The Algorithm

The curve-network interpolation algorithm employed in this work is the Gordon surface algorithm from TiGL [60] due to its open availability. The method addresses the problem of constructing a smooth surface that interpolates a network of intersecting profile and guide curves. Conceptually, the surface is generated by combining three intermediate interpolation surfaces: the skinning surface through the profiles $S_f(u, v)$, the skinning surface through the guides $S_g(u, v)$, and a corrective interpolated surface through the intersection points between profiles and guides. The final Gordon surface is obtained by adding the first two and subtracting the third, as schematically illustrated in figure 4.14. Due to use of this superposition principle, all three surfaces has to be *parametrically compatible*[19] with each-other, leading to the need of re-parametrisation of the input curves.

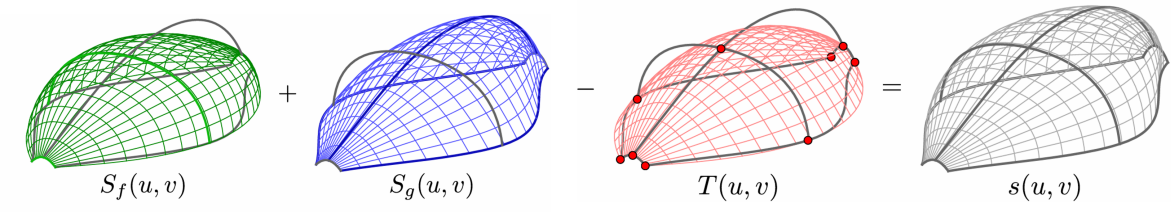


Figure 4.14: Construction of the Gordon surface using its three interpolation surfaces, adapted from [60].

The implementation used in this work is the open-source version of TiGL itself[20], which imposes several strict geometric requirements on the input curve network. In particular, each profile must intersect every guide exactly once, and no curve may extend beyond the intersection domain. When these geometric constraints are not perfectly satisfied, or when internal inconsistencies occur in the algorithm, the surface generation may fail. Such failures can occur silently or return ambiguous error messages, such as indicating that profiles and guides do not intersect when they do as verified using the underlying OCC kernel (section 8.1.3 addresses this topic in detail). To conclude, the robustness of this implementation is questionable and requires further research.

The exact causes of these failures have not been investigated in this study, as diagnosing and modifying the TiGL implementation lies outside the scope of this work. However, their frequency and location within the tool (e.g., at what point in the geometry generation flow) are quantified as part of the overall performance assessment in section 8.1.3. Leading to the final conclusion that 25% of geometric failure cases are due to Gordon surface internal failure of which visual example cases can be found in chapter D.

4.5. Blending of Solids Algorithm

As discussed in section 4.4, smooth geometric continuity is essential for accurate aerodynamic modelling as discontinuous (G^0) intersections would produce sharp edges and cusps. Therefore the integration of different subsystems within the tool, each defined by a distinct outer shape, must also ensure smooth transitions between adjacent surfaces requiring at least tangential (G^1) continuity. As outlined in section 4.1, these subsystems collectively form the complete aircraft configuration, and their mutual intersections or transitions have a significant effect on the overall aerodynamic performance, as highlighted in section 2.2.

To achieve this, a geometric operation is required that can blend two intersecting or overlapping surfaces into a single continuous solid while maintaining intuitive control over the intermediate transition. Due to this intermediate control requirement, the conventional blending technique of a simple fillet, does not suffice. Furthermore, the blending surface must be geometrically continuous with the adjacent faces of both solids, maintaining aerodynamic and geometrical smoothness.

To enforce consistent surface orientations, simplify inputs (e.g., blend directions), and leverage geometrical operations that automatically account for internal volume, the new algorithm is required to perform

[19] *Parametric compatibility* means that the surfaces are of same degree and share knot identical vectors [60].
[20] `https://github.com/DLR-SC/tigl`

blending at the solid level: it constructs an intermediate blending surface and returns a single unified body, hence the name `BlendSolid`.

Scope of Blending Algorithm

The implemented algorithm is limited to cases where the intersection between two solids occurs over exactly one face of each body. One of these faces must be a closed surface (for example, the curved side of a cylinder with a single seam edge), while the other is an non-closed (all edges are free) surface (such as a rectangular face of a cube). The seam edge of the closed surface must pass once through the opposing face, ensuring a single, well-defined overlap region. This restriction prevents ambiguous intersections (such as the cylindrical surface intersecting multiple faces of the cube) and ensures a robust and controllable blending operation.

Motivation for Development

Neither ParaPy nor OCC provides a native algorithm for creating smooth, parameter-controlled blends between intersecting solids, unlike commercial CAD tools such as CATIA[21], which offer dedicated blending and surface-filling features. This limitation of ParaPy and OCC (discussed in section 3.1.4) motivated the development of a custom blending approach in this work. The only related OCC functionality is the Coons patch[22], which interpolates boundary curves but offers limited internal shape control, making it unsuitable for aerodynamic applications where curvature quality and flow sensitivity are critical. Moreover, multiple Coons patches generally yield inferior continuity compared to a single Gordon surface [61].

Consequently, a new blending algorithm was developed and implemented within ParaPy, designed to achieve controlled and continuous surface transitions between solids. This algorithm, termed `BlendSolid`, provides the level of flexibility and smoothness required for aerodynamic integration of the subsystems in this work.

4.5.1. The Algorithm

The `BlendSolid` procedure proceeds in four phases: (1) prepare intersecting solids, (2) trim the overlapping faces, (3) construct blend guides, and (4) generate and assemble the blend surface into a watertight solid. A geometric step-by-step illustration on a cylinder-cube example is given in Figure 4.15. Because the guide construction controls the local blending shape, the “Blend guides” step is detailed separately in Section 4.5.2.

4.5.2. Blending Curve

A `BlendCurve` is a Bézier curve constructed between two points lying on the boundaries of two faces adjacent to the intended blending surface. An outline of the algorithm is shown in figure 4.16. The curve’s global shape is governed primarily by the local geometry of the supporting faces, ensuring the required geometric continuity between the curve and the adjacent surfaces. This continuity follows from the properties of Bézier curves: G^0-continuity is enforced by the endpoint control points, G^1-continuity by the alignment of the first interior control points with the endpoint tangent directions, and higher-order continuity through analogous alignment of subsequent control points. By positioning these control points along geometrically meaningful directions derived from the faces, the resulting `BlendCurve` smoothly transitions between the surface boundaries.

In addition to this geometry-driven behaviour, a small number of user-defined parameters allow controlled adjustment of the curve’s shape which are highlighted in section 4.5.3. These parameters influence, for example, the weighting between competing tangent directions or the distance of interior control points from the endpoints, thereby providing designers with intuitive flexibility while preserving the required continuity constraints.

[21] `https://www.3ds.com/products/catia`
[22] `GeomFill_ConstrainedFilling` in OCC

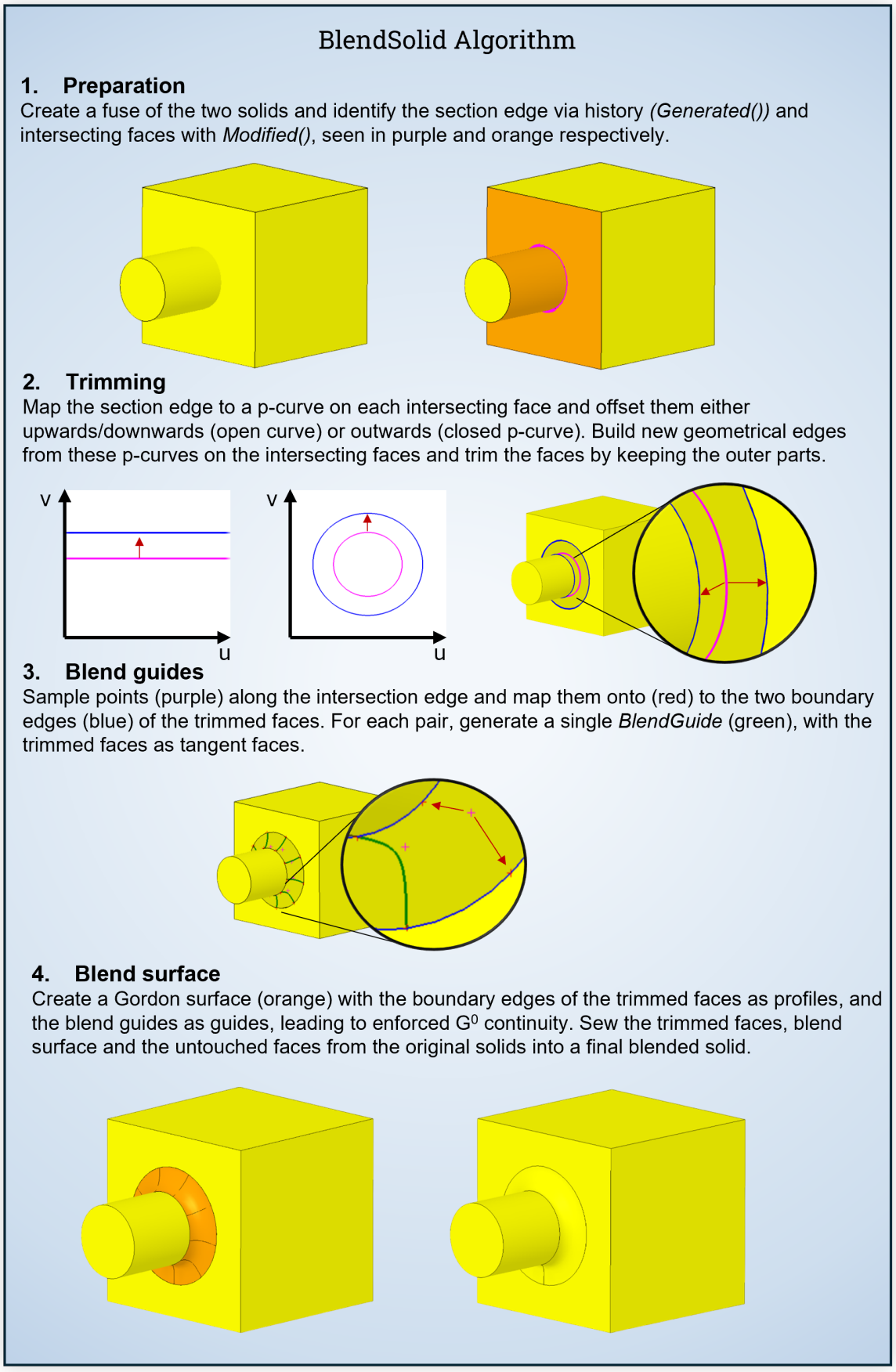
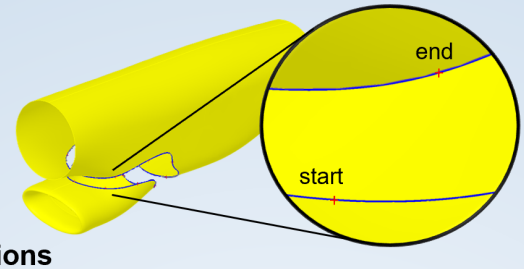


Figure 4.15: Illustrative workflow of the `BlendSolid` algorithm showing the merging of cube and a cylinder into a single continuous blended geometry.

BlendCurve Algorithm

1. Boundary Edges

For each end points (start and end), identify the boundary edge (blue) on the supporting face, by checking which edge of the face, contains the end point.

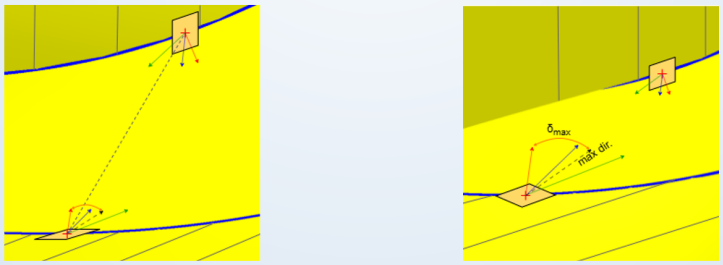


2. Tangent directions

For both end points, evaluate the supporting face normal (orange) at that location. Form the *blend direction* (red), which is the cross-product between the face normal and the tangent of the boundary edge (blue). Compute the *projected direction* (green), i.e. the chord (gray) projected onto the normal plane.

3. Final Tangents

Final tangent (blue) is a weighted combination of the *blend direction* and *projected direction*. A maximum angular deviation from the *blend direction*, is provided by a single *maximum angle* (δ_{max}), increasing the sewing robustness in the *BlendSolid* algorithm.



4. Control Points

Depending on desired contintuity (G^1 or G^2), place 1 or 2 control points along the final tangents. The distance from the end points is governed by a single σ, with $\sigma = 1$ corresponds with placement at one-third of the chord length away from the end point.

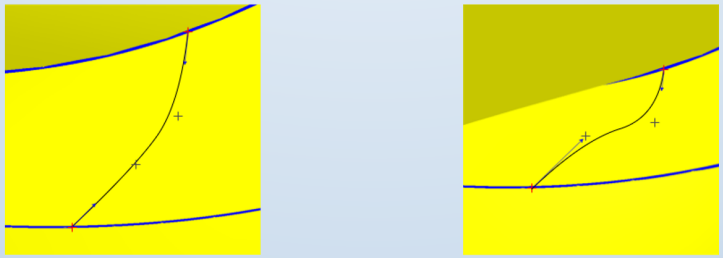


Figure 4.16: Illustrative workflow of the `BlendCurve` algorithm showing the construction of a Bézier curve from two points on the nacelle and scoop boundary for the baseline design. With two views of the same situations, an iso-view and the same view rotated upwards with a significant amount.

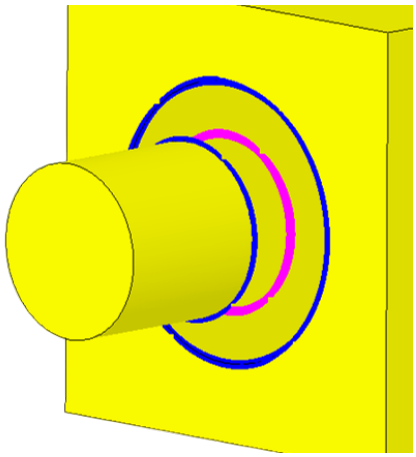
4.5.3. Shape Control

A key advantage of `BlendSolid` is the intuitive, parametric control it provides over the intermediate blending surface. In contrast to conventional blending approaches like constant-radius fillets, which offer limited influence over the blend geometry, `BlendSolid` exposes parameters that allow direct control over blend surface size, sharpness, and directional behaviour.

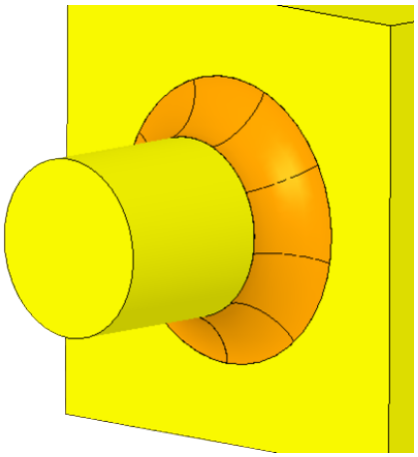
Trimming distance

The first shape control parameter is the *trimming distance*, which defines how far the intersecting faces are retracted prior to constructing the blending surface (step 2 in figure 4.15). A single trimming value applies an equal setback to both faces, as illustrated in figure 4.17a and figure 4.17b. Alternatively, two independent values may be specified: the first for the trimming distance measured from the first solid, and the second from the other, as demonstrated in figure 4.17c and figure 4.17d.

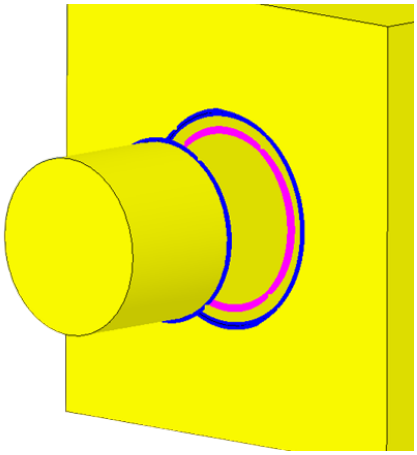
The trimming distance in `BlendSolid` is analogous to a rolling-ball fillet with constant radius: equal distances yield a fillet-like intermediate surface (figure 4.17b), while allowing different distances per face provides greater flexibility and local shape preservation. Because trimming distances are defined in absolute units, they are independent of face orientation, unlike fillet radii where trimming of the surfaces depend on normal angles between them. A current limitation is that trimming distances cannot vary along the intersection curve without significantly complicating the p-curve shifting in step 2 of figure 4.15.



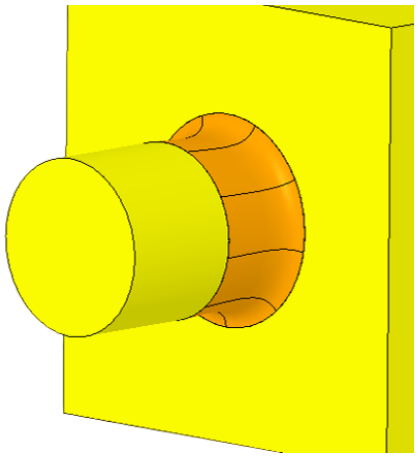
(a) Section edge (purple) and trimmed face boundaries (blue).



(b) Resulting blend surface (orange) and blending guides.



(c) Section edge (purple) and trimmed face boundaries (blue).



(d) Resulting blend surface (orange) and blending guides.

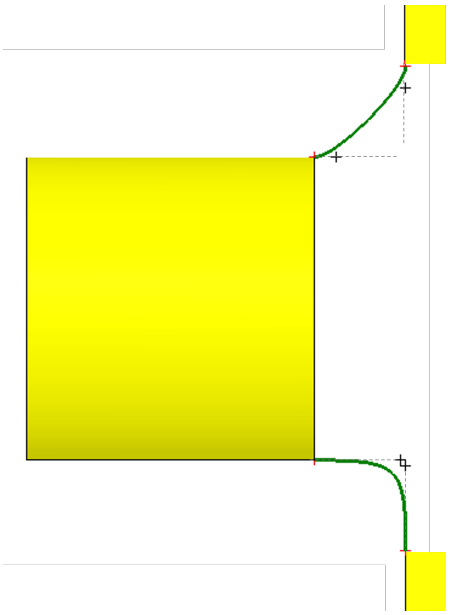
Figure 4.17: Cylinder-cube blending using `BlendSolid`. Top: uniform trimming distance of 0.15 m in both directions. Bottom: asymmetric trimming with 0.25 m toward the cylinder and 0.05 m toward the rectangular face.

Sharpness Control

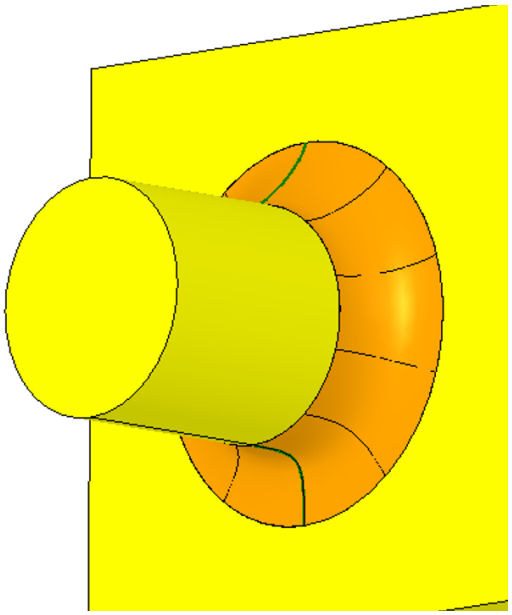
The sharpness of the blend is governed by a parameter σ, which is passed to all blending curves composing the surface. The parameter controls the distance of the second row of Bézier control points from the curve endpoints (step 4 in figure 4.16), with the points constrained to lie on a direction tangent (more on control of this direction in section 4.5.3 under Blend Guides Direction) to both adjacent surfaces.

For $\sigma = 1.0$, the control points lie at one-third of the chord length, producing a moderately rounded blend. Higher values produce a sharper curvature, while lower values yield smoother, flatter transitions. Two representative cases are illustrated in figure 4.18a.

Instead of a single σ value, a σ-*law* may be supplied, allowing the sharpness to vary along the intersection edge between the solids. In figure 4.18, an interpolated law is used with three points (u, σ): $(0.0,\ 2.0)$, $(0.5,\ 0.5)$, and $(1.0,\ 2.0)$. The resulting surface in figure 4.18b demonstrates smooth variation of sharpness across the blend.



(a) Two blend curves (green) with $\sigma = 2.0$ (bottom) and $\sigma = 0.5$ (top).



(b) Resulting blend surface (orange) with varying sharpness.

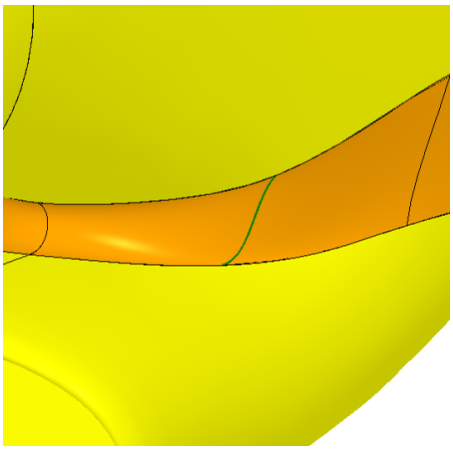
Figure 4.18: Cylinder-cube blend using an interpolated σ-law starting at 2.0, to 0.5 halfway and back to 2.0 along the intersection curve.

Blend Guides Direction

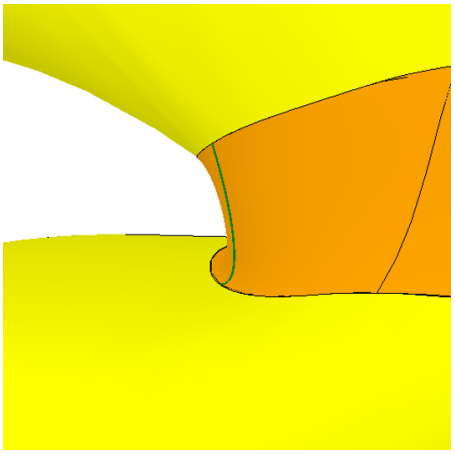
To satisfy tangency, the `BlendCurve` Bézier control points lie in the plane normal to the adjacent faces (orange plane in figure 4.16), where the G^1-continuous tangent (blue) is obtained by interpolating between the *direct* direction toward the opposite endpoint (green) and the *boundary-normal* direction of the adjacent face (red), with interpolation factors ranging from 0.0 to 1.0.

Intuitively, the direction factors control how directly the blend curve connects its endpoints. A value of 0.0 aligns the curve fully toward the opposite endpoint, producing a flatter and less sheared surface, while a value of 1.0 aligns it with the boundary-normal direction, yielding a more parametrically aligned and sheared surface, both seen in figure 4.19. Intermediate values provide smooth control between these behaviours.

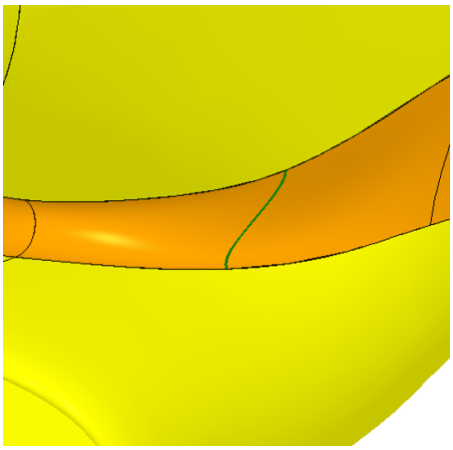
To ensure robust sewing in step 4 of figure 4.15, the parameter `max_angle` limits deviation from the boundary-normal direction. Setting `max_angle` to $0°$ enforces full boundary-normal alignment (equivalent to clamping the direction factors to 1.0), while larger values allow increasing freedom toward more direct blending.



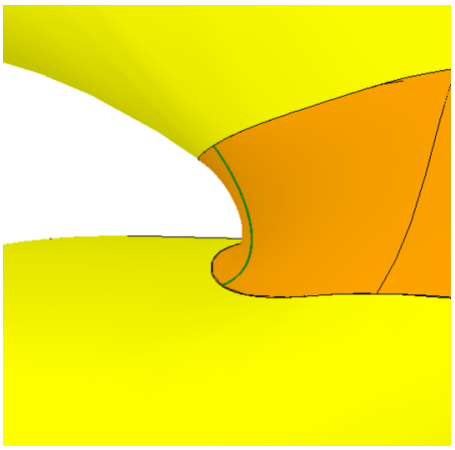
(a) Frontal view highlighting the modified blend curve (green) with direction factors set to 0.0.



(b) Rear view showing the same modified blend curve with direction factors set to 0.0.



(c) Frontal view with stronger parametric alignment with the boundary edge or direction factors set to 1.0.



(d) Rear view with stronger parametric alignment with the boundary edge or direction factors set to 1.0.

Figure 4.19: Effect of blend-curve direction control in nacelle-scoop blending. Top row: fully *direct* tangent direction. Bottom row: aligned with the adjacent-face boundary normal.

4.5.4. Continuity Assessment

As introduced in figure 4.15, the blending surface generated by the `BlendSolid` algorithm is a Gordon surface that interpolates a set of tangent `BlendCurve` guide curves. Since this surface enforces tangency only at the guide locations and not in between, the resulting blending surface is *quasi-tangent*[23] to the adjacent faces. Increasing the number of guides is therefore expected to improve overall tangential continuity along the boundary.

Continuity Metrics

Tangency (G^1 continuity) between two surfaces is quantified by the angular deviation of their surface normals along the shared boundary. For N evaluation points, three scalar indicators are defined, as maximum deviation, mean deviation and RMS deviation, seen in equation (4.1), equation (4.2) and equation (4.3) respectively. Additionally, a sampling convergence study (figure 4.20) demonstrated that $N = 100$ points is sufficient for stable continuity metrics. This value is used for all subsequent evaluations.

[23] 'Quasi-tangent' is in this work an arbitrary term and defined as almost tangent where two geometrical entities tangency deviates with less then 1° along the full boundary.

$$\Delta\theta_{\text{max}} = \max_i \Delta\theta_i \tag{4.1}$$

$$\Delta\theta_{\text{mean}} = \frac{1}{N}\sum_i \Delta\theta_i \tag{4.2}$$

$$\Delta\theta_{\text{RMS}} = \sqrt{\frac{1}{N}\sum_i \Delta\theta_i^2} \tag{4.3}$$

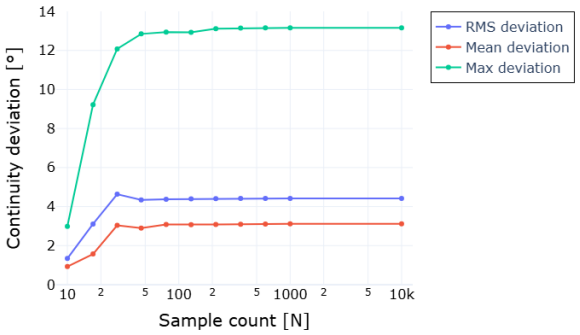


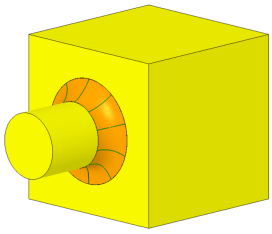
Figure 4.20: Sampling convergence of continuity deviations for the baseline nacelle-scoop blend with 10 guides.

Blending Cases

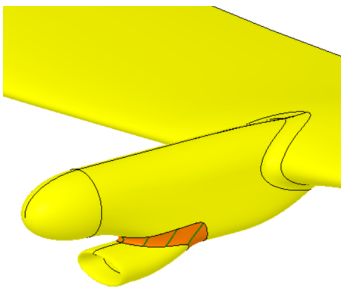
Three representative blending configurations were analysed:

1. **Cylinder-cube blend**, seen in section 4.5.4 (low geometric complexity)
2. **Nacelle-scoop blend** of baseline design, seen in section 4.5.4 (moderate complexity)
3. **Wing-nacelle blend** of baseline design, seen in section 4.5.4 (high complexity)

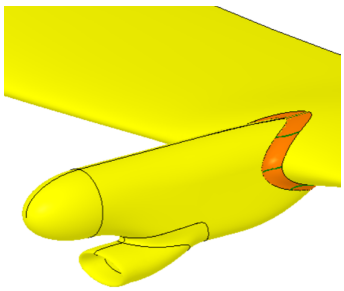
Geometric complexity in this context is characterized by the curvature of the intersection curve between the solids and the angular deviation of the surface normals. According to this definition, the wing-nacelle blend exhibits the highest complexity. Its leading-edge curvature induces a strongly double curved intersection, and the associated surface normals are nearly orthogonal at the high curvature leading-edge, producing a substantially more demanding blending configuration than the cylinder-cube or nacelle-scoop cases.



(a) Simple **cylinder-cube blend** geometry.



(b) Nacelle-scoop blend of the baseline design.



(c) Wing-nacelle blend of baseline design

Figure 4.21: `BlendSolid` (orange blending surface) used in different blending cases with identical blending parameters, σ = 1.0, offset = 0.1 m and 10 blending guides (green) with default direction factors = 0.0 and max angle to 45°.

Continuity Results

Across all three use cases, tangential continuity improves as the number of blending guides increases, with the largest gains occurring for the first few guides. For the simple cylinder-cube blend, the continuity metrics approach zero, consistent with the theoretical limit of infinite guides approaching full continuity. In contrast, the scoop-nacelle and wing-nacelle baseline integrations exhibit instability beyond approximately 70 guides, where continuity deviations increase and the Gordon surface solver ultimately diverges.

Although full divergence occurs near 70 guides for both configurations, instabilities already appear at lower counts. Across all cases, a practically stable region is observed up to approximately 16 guides,

beyond which solver failures become increasingly likely. Therefore, the `BlendSolid` algorithm is considered fully stable up to 16 blending guides for the use cases examined in this work.

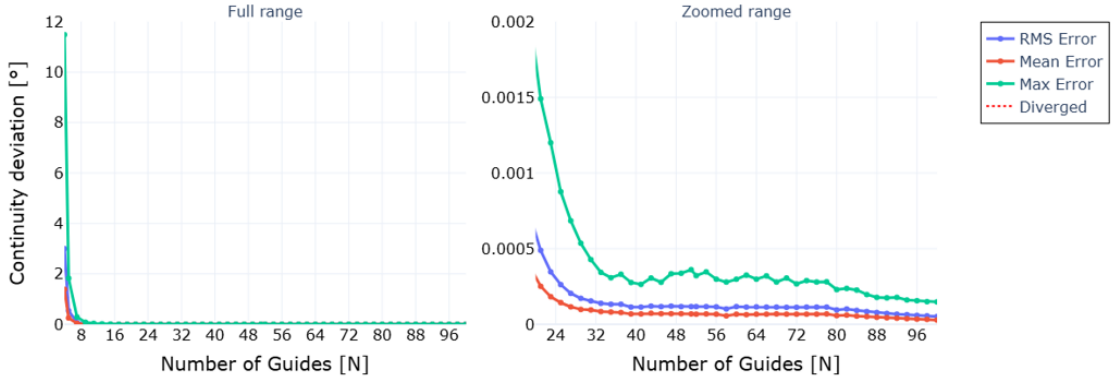


Figure 4.22: Convergence of continuity deviations of the `BlendSolid` for the simple geometry, with 100 sample points.

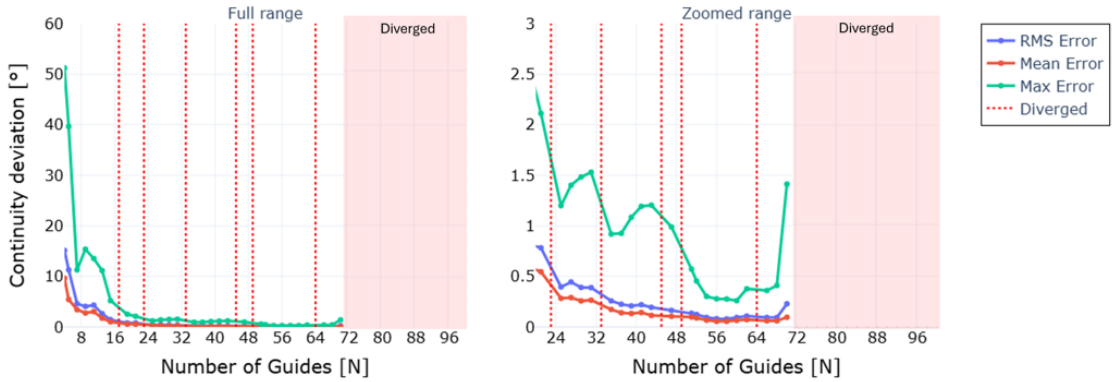


Figure 4.23: Convergence of continuity deviations of the `BlendSolid` for the baseline design at nacelle-scoop integration, with 100 sample points.

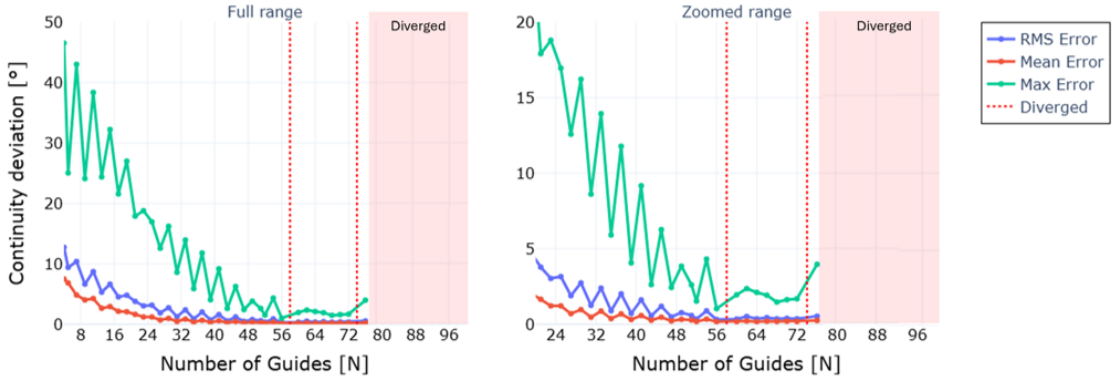


Figure 4.24: Convergence of continuity deviations of the `BlendSolid` for the baseline design at wing-nacelle integration, with 100 sample points.

For the simple geometry, continuity deviations are on the order of 10^{-2} degrees in the stable region below 16 guides (RMS $= 0.00207°$, mean $= 0.00099°$, max $= 0.00863°$), and reduce further to the order of 10^{-4} degrees for higher guide counts (RMS $= 52.2\,\mu°$, mean $= 27.9\,\mu°$, max $= 148.6\,\mu°$). Since numerical tolerances in the geometry kernel are typically of the order 10^{-7}, these results show that the `BlendSolid` does not reach full numerical tangency, but becomes significantly close for simple geometry.

For the nacelle-scoop case, RMS and mean deviations reach approximately $1.44°$ and $1.02°$ at the end of the stable region, which may be considered *quasi-tangent* for conceptual aerodynamic use. However, the maximum deviation remains high ($5.20°$), indicating non-uniform tangency along the boundary. In unstable regions, deviations drop to about $0.3°$, implying near-uniform quasi-tangency at the expense of geometric robustness.

For the wing-nacelle case, continuity is lower: in the stable region, RMS and mean deviations are $6.63°$ and $2.92°$, exceeding typical quasi-tangent limits. In unstable regions, the RMS drops below $0.5°$, but maximum deviations remain above $1.4°$, indicating non-uniform tangency. Nevertheless, algorithm robustness increases, resulting in a larger region where quasi-tangency can be enforced.

Conclusion

The continuity of `BlendSolid` strongly depends on the geometric complexity of the intersecting solids. Near-tangent surfaces are obtained for simple intersections (table 4.1), while quasi-tangent behaviour is achievable for moderately complex cases such as the nacelle-scoop blend, though not uniformly. For highly curved intersections, such as wing-nacelle, quasi-tangency is not reached in the stable region, and `BlendSolid` is therefore recommended for simple to moderately complex blending operations.

Use Case	Stable RMS Deviation [°]	Tangency Assessment
Cylinder-cube blend	$< 10^{-4}$	Near numerically tangent
Scoop-nacelle blend	~ 1.0	Quasi-tangent
Wing-nacelle blend	> 2.5	Not quasi-tangent in stable region

Table 4.1: Continuity performance summary of `BlendSolid` for three geometric configurations.

4.5.5. Conclusion and Limitations

The `BlendSolid` algorithm represents one of the first fully parametric blending strategies, offering an intuitive method for creating smooth transition geometry between overlapping solids. It enables quasi-tangent blends for low- to moderately complex intersections with stable, well-bounded user control. The unique `BlendSolid` algorithm is designed specifically for knowledge-based engineering workflows and highly parametric, generative modelling.

Despite its potential, the current implementation of `BlendSolid` exhibits several limitations:

- **Only quasi-tangent surfaces are achieved**, and only for low- to mid-complexity geometries. Higher-fidelity aerodynamic studies typically require strict G^1 or G^2 continuity, which is not guaranteed. The limitation originates from the TiGL Gordon surface implementation, which does not enforce continuity with adjacent faces.
- **No single optimal blend solution is generated.** The user must manually choose all shape-control parameters. No optimisation or objective-driven blending strategy, such as minimising curvature variation or bending energy, is included. As a result, either many design variables must be introduced in MDO workflows, or sub-optimal default blends are used, potentially degrading aerodynamic performance, geometry integration quality and design time.
- **Independence of blending guides can compromise geometric fidelity.** Because guides are constructed without mutual coupling, the resulting surface may become locally compressed or stretched, degrading the preservation of boundary-edge shape. This is particularly problematic in regions requiring local aerodynamic precision (e.g. wing leading edges). This loss of blending guide spacing consistency is also suspected to contribute to divergence of the Gordon solver when rails become too close or overlapping.
- **Gordon surface performance limits the algorithm.** Increasing the number of guides improves global tangency, but also increases the likelihood of solver divergence. For complex intersections, divergence was observed even with a moderate number of guides, constraining practical use to a reduced guide count and thereby limiting achievable continuity by order of 10 when comparing 16 and 70 blend guides.

- **Restricted geometric applicability.** The present implementation supports only the specific case of *one closed face intersecting one open face* with a single, continuous intersection curve. More general blending scenarios, like multiple intersection loops, non-manifold contact, partial overlaps, etc., are not supported. While sufficient for the configurations in this thesis, it limits broader applicability.

Overall, the `BlendSolid` algorithm demonstrates strong potential as a robust parametric solution for blending complex shapes. To fully realise this potential, the following improvements are recommended, listed in order from highest to lowest priority:

1. **(Re-)implementation of the Gordon surface algorithm.** A re-implementation of the Gordon surface method is strongly recommended within ParaPy or directly in OCC. The current TiGL-based implementation lacks robustness, offers limited diagnostic feedback when failures occur, and-critically-does not enforce tangential continuity with adjacent faces. A more suitable approach would mirror the behaviour of the *Loft Surface* operator in CATIA[24], which supports profile-guide driven surface construction that explicitly respects G^1 continuity. An improved algorithm of this nature would directly address two principal limitations identified in this work: the quasi-tangent continuity of the current blending surface and the growing instability of the TiGL implementation when the number of guides increases. With such an enhancement, the resulting blend surface would be numerically tangent to neighbouring faces by design, substantially improving suitability for high-fidelity aerodynamic applications.
2. **Improved blending guide construction.** In its present form, each blend guide is generated independently, which means no constraint exists to maintain a consistent geometric progression between neighbouring guides. As a result, the blending surface may exhibit excessive local stretching or compression, and the intermediate cross-section shape is not preserved except at the boundary curves. It is recommended that future work incorporates coupling between guides during construction, for example by correlating guide direction factors or enforcing shape similarity constraints along the blending region. An alternative or complementary approach is to introduce intermediate interpolation profiles between the two trimmed boundary edges, ensuring that the blending surface reflects the evolution of shape between endpoints more faithfully. Such modifications would significantly increase the effectiveness of `BlendSolid` for aerodynamically sensitive regions such as wing leading edges or nacelle-pylon junctions.
3. **Automated initial parameter estimation / optimisation.** At present, obtaining a desirable blending surface requires manual tuning of the control parameters described in section 4.5.3. For MDO or large-scale design-space exploration, this manual procedure is impractical and does not guarantee a high-performing initial blend. It is therefore recommended that a method be implemented to compute an initial solution automatically based on a chosen objective, e.g. minimising a bending energy or minimising curvature variation. Such a feature would reduce user workload and enable more meaningful integration of `BlendSolid` into automated optimisation workflows.
4. **Extension of geometrical scope and applicability.** The current algorithm is restricted to cases where two solids intersect over exactly one closed and one open face, and where the resulting intersection curve lies entirely within a single face on each solid. Broadening the applicability to more general intersection configurations would increase the practical value of the algorithm substantially. Recommended extensions include: support for multiple intersection patches, blending between arbitrarily shaped overlaps, and the ability to blend not only solids but also surface patches. This will require more advanced tracking of face orientations and local parameterisation, but would remove one of the major constraints on usability and allow `BlendSolid` to serve as a generic blending operator rather than a special-case tool.

[24] `https://dshelp-embed.3ds.com/2025/english/CATIA_P3/online/CATIAfr_C2/iaeugCATIAfrs.htm`

5

WIng Package Modelling

Building on the parametric modelling principles established earlier, this chapter introduces the wing as the primary lift-generating component of the WNI system assembly. A generic lifting-surface formulation is adopted to ensure reusability and extensibility through shared, modular profile and airfoil parametrisations.

Section 5.1 presents the wing package architecture , emphasising class relationships and responsibilities. This is followed by the definition of profiles and airfoils in section 5.2. These elements are combined into a 3D lifting surface in section 5.3, including a *clean wing* parametrisation [56, 57]. A matched lifting-surface primitive for nacelle integration is introduced in section 5.5, and the associated capability modules are discussed in section 5.6.

5.1. Architectural Overview

In aircraft wing design, the wing's aerodynamic characteristics are commonly governed by its 2D planform representation or its projection onto a reference plane [62]. From a geometric standpoint, a planform may be fully described by two spatial curves: the LE and TE rails. Accordingly, the core representation of any lifting surface in this work begins with a general rail-based definition referred to as the `WingSkeleton`[1].

The complete 3D wing shape, however, is not defined by planform geometry alone. Cross-sectional airfoils placed between the rails determine the aerodynamic shape and thickness. The simplest unrestricted 2D sectional shape in this work is the `Profile`. Together, the `WingSkeleton` and sectional `Profile` objects form the structural foundation for more concrete lifting-surface implementations, as illustrated in figure 5.1.

To clarify class responsibilities, the classes in figure 5.1 are grouped into three categories, used consistently throughout this thesis:

- **Abstract** classes define fundamental modelling concepts and constraints without prescribing an implementation, intentionally left to a minimal. All subclasses must respect this core logic.
- **Configurable parametric** classes derive their behaviour from composed parametrisations (section 4.2) and can be extended by supplying new parametrisation objects.
- **Specialisation** classes implement fixed modelling strategies on top of abstract bases, encoding rigid logic that typically requires class modification to change.

[1] The `WingSkeleton` is not restricted to planar geometry as its LE and TE rails may be any spatial curves, with additional constraints introduced later in this chapter.

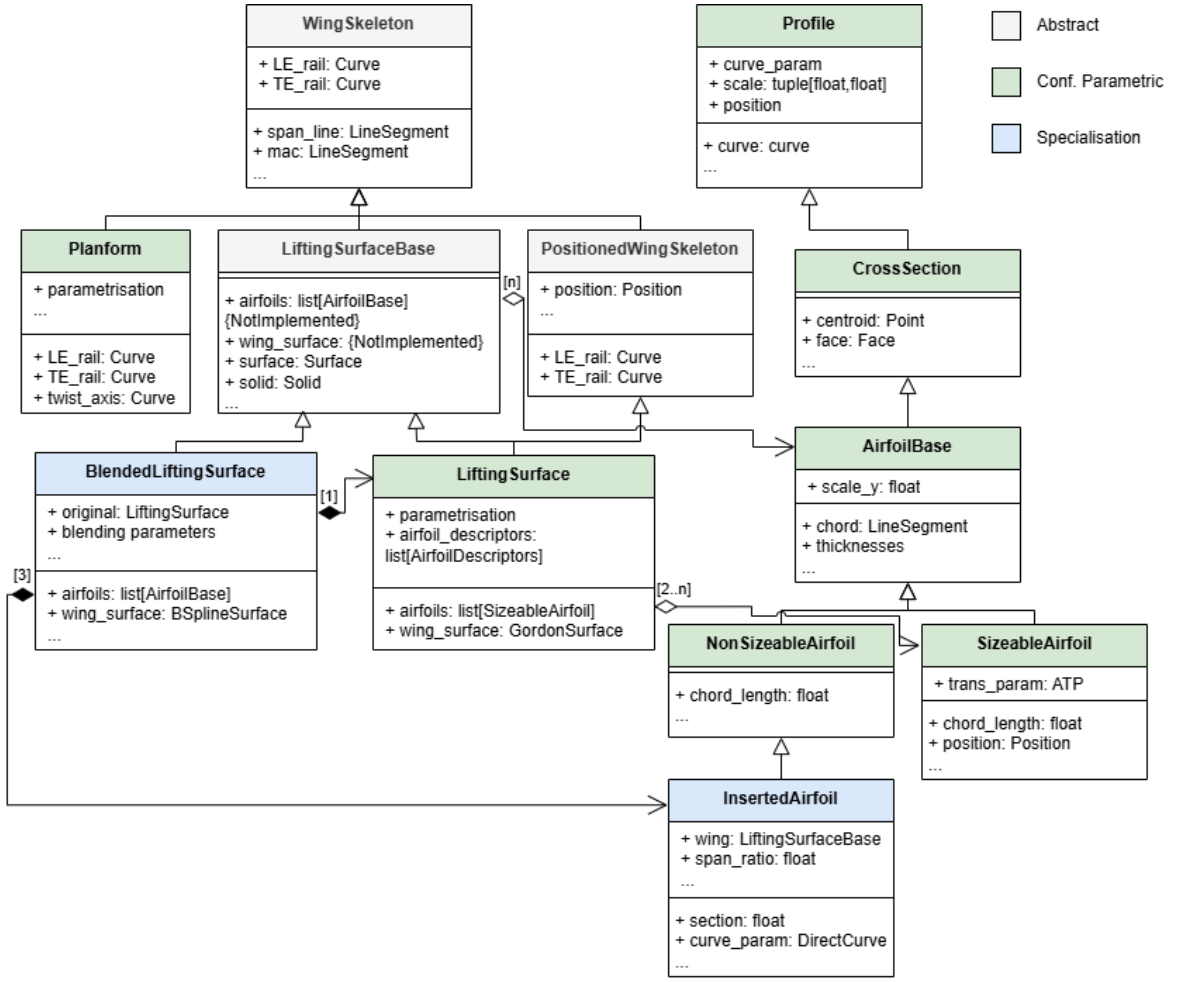


Figure 5.1: UML representation of all wing-related classes. Abstract classes (gray), configurable parametric classes (green), and specialisation classes (blue) illustrate the structural and functional relationships between lifting-surface and profile entities. Not all attributes are shown as the diagram serves primarily as a structural overview.

At the top level, the `WingSkeleton` defines the LE and TE rails and computes high-level wing-related geometric properties such as span, spanline, and mean aerodynamic chord (MAC). The `Planform` is a parametrisation-driven (parametrisation configurable) subclass that constrains the rails to the XY-plane, while the `PositionedWingSkeleton` repositions any rail definition using a given `Position` transform.

For cross-sectional geometry, `Profile` provides a general 2D shape with a shape parametrisation, positioning and scaling options. `CrossSection` enforces that the curve is closed and counter-clockwise oriented. `AirfoilBase` builds airfoil-specific properties (LE, TE, chord, t/c, etc.), enabling two concrete subclasses: `NonSizeableAirfoil`, where chord length follows directly from the parametrisation curve, and `SizeableAirfoil`, which computes size from a specific `AirfoilTransformParametrisation` (ATP). The specialised `InsertedAirfoil` extracts airfoils directly from a lifting surface at a specific span location, and is used by the `MatchedLiftingSurface`.

The `LiftingSurfaceBase` class defines a generic lifting surface by combining `WingSkeleton` rails with airfoils to form a 3D geometry. Its *configurable parametric* subclass, `LiftingSurface`, extends this by using rail parametrisations and airfoil descriptors (section 4.2.1 and section 5.3.2), allowing the surface to be fully defined through interchangeable parametrisation components. This makes it the primary and most reusable lifting-surface implementation, applicable to wings, and potentially propeller blades, empennage components, and high-lift devices.

Lastly, the specialised `MatchedLiftingSurface` locally increases the thickness in a selected region of

an existing lifting surface. This behaviour is similar to Free-Form Deformation (FFD) techniques [63] as it deforms an existing surface and is primarily used to generate smooth geometric transitions (blends) between lifting surfaces and adjacent bodies, such as nacelles or fuselage structures, while preserving the underlying parametrised definition of the original lifting surface.

5.2. Profiles and Airfoils

Profiles, cross-sections and airfoils[2] constitute the first half of the wing package introduced in the previous section. As Sóbester and Forrester [62] note, "the external geometry of a topologically simple[3] three-dimensional body can be viewed as a surface lofted over a series of cross-sections," highlighting that cross-sectional profiles are fundamental building blocks of any 3D shape in this work. Consequently, profiles and cross-sections form the basis of subsequent geometric components, as for example, a wing is primarily defined by its airfoil sections, with additional rails providing enhanced lofting control.

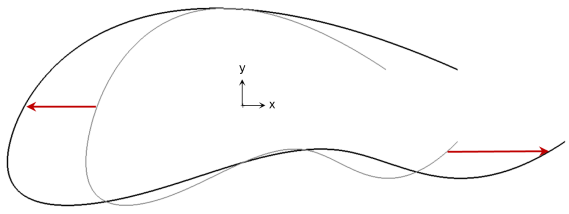
5.2.1. Profiles and Cross-sections

A (open) `Profile` represents the most fundamental 2D cross-sectional shape used throughout this work. Conceptually, it is defined as any planar curve[4] that may be positioned arbitrarily in 3D space. In addition, the `Profile` supports non-uniform scaling in the local x- and y-directions of the curve plane, allowing controlled variations in sectional shape.

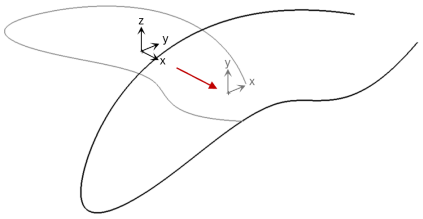
To ensure geometric consistency, two requirements are imposed on the underlying, *unpositioned* base curve:

- The curve must be a valid `Curve` object.
- The curve must lie entirely in the local XY-plane.

This base curve is produced by a subclass of the abstract `CurveParametrisation`, which is responsible for generating the planar curve definition in its local coordinate system and will be discussed more elaborately in section 5.2.3. After generation, the curve is scaled in x and y according to the prescribed scaling factors, as illustrated in figure 5.2a, and subsequently transformed into the global coordinate system using the supplied `Position`, as shown in figure 5.2b.



(a) Scaling of a base profile (grey) by a factor of 1.5 in its local x-direction, resulting in the scaled profile (black).



(b) Transformation of the scaled profile into the global coordinate system (black). The original, unscaled profile and its local frame are shown in grey.

Figure 5.2: Scaling and transformation of an arbitrary `Profile`.

The `CrossSection` subclass imposes additional geometric constraints on the unpositioned base curve:

- The curve must be closed.

[2] In this thesis a specific terminology is defined: *profiles* are two-dimensional open curves, *cross-sections* are closed profiles, and *airfoils* are cross-sections with uniquely defined LE and TE points.

[3] *Topologically simple* refers to bodies whose surface is a single, closed, genus-0 (i.e., without holes) manifold that can be continuously deformed into a sphere without tearing or removing holes. The wing, nacelle, and scoop intake considered in this thesis satisfy this property and are therefore representable as lofted surfaces over sequential cross-sections. Note that this mathematical definition differs from the CAD notion of topology where topology refers to how different geometrical entities are related.

[4] Throughout this thesis, the term *curve* refers to any subclass of ParaPy's `Curve`, i.e. an OpenCascade `Geom_Curve`, including but not limited to lines, circular arcs, conics, Bézier and B-spline curves.

- The curve must be orientated *counter-clockwise* with respect to its local coordinate system, as depicted in figure 5.3.

Closure enables construction of a geometric face, allowing computation of a Centre-of-Gravity (CoG). From this CoG, characteristic points such as the horizontal and vertical points can be determined, which are of particular importance in nacelle modelling (see chapter 6). The enforced orientation ensures consistent face normals and curve directions, which is essential for robust downstream geometric operations.

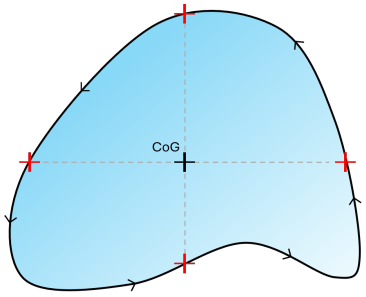


Figure 5.3: Example closed `CrossSection` with counter-clockwise orientation. The computed Centre-of-Gravity (CoG) is annotated and horizontal/vertical characteristic points are shown in red.

5.2.2. Airfoils

Airfoils form a specialisation of cross-sections that incorporate additional geometric analyses to compute a range of aerodynamic shape characteristics. These computations are performed directly on the airfoil curve itself. Consequently, any curve may serve as an airfoil input, provided it satisfies the previously mentioned general cross-section requirements as well as additional constraints imposed by `AirfoilBase`:

- The start and end of the curve must coincide with the x-extrema and lie on the local x-axis.
- The leading-edge (LE) point must be located at $(0, 0)$.

The `AirfoilBase` class computes a number of geometric properties of the scaled and located airfoil as attributes, including: LE and TE points, LE radius, chord line, camber line, maximum thickness-to-chord ratio (t/c), and the location of the maximum t/c. An example of these computed characteristics for a Clark-Y airfoil is shown in figure 5.4.

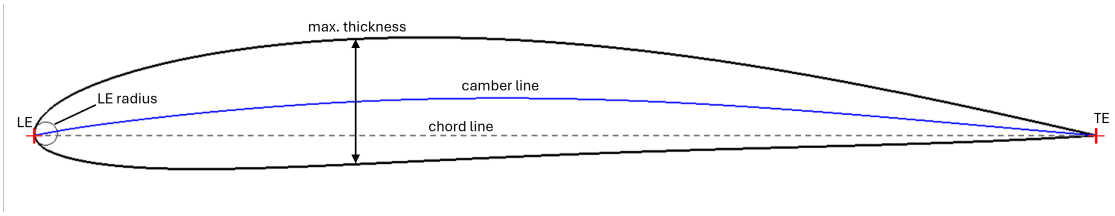


Figure 5.4: Clark-Y airfoil with computed characteristics: $t/c_{\max} = 0.118$, LE radius $= 0.011$ m, and $x_{t/c_{\max}} = 0.303\,c$. visualised in the ParaPy viewer

Two concrete subclasses extend `AirfoilBase`: `NonSizeableAirfoil` and `SizeableAirfoil`. The distinction between them lies in how the chord length is determined. A `NonSizeableAirfoil` derives its chord directly from the geometry of the base curve defined in the curve parametrisation and thus loses its airfoil scaling capability. In contrast, a `SizeableAirfoil` allows the chord length to be prescribed explicitly by means of a transformation parametrisation.

This functionality introduces an additional constraint for the `SizeableAirfoil`: the underlying airfoil curve from the curve parametrisation must be non-dimensional, spanning from LE at $(0, 0)$ to TE at $(1, 0)$. As illustrated in figure 5.1, the sizeable airfoil adds, in addition to the curve parametrisation,

a new *transform* parametrisation with an `AirfoilTransformParametrisation` (ATP)[5]. The ATP computes both the chord value and the spatial placement of the airfoil. Further implementation details of these transformation parametrisations are provided in section 5.2.5.

5.2.3. Curve for Profiles and Airfoils

The most primitive option is the *direct* curve parametrisation, which accepts any curve object as input, i.e. `Curve` subclasses (such as the `BlendCurve` section 4.5.2). This offers maximum flexibility, but at the expense of robustness and design intuition, as no aerodynamic or profile-specific constraints are enforced and the curve definition is decoupled from airfoil design philosophies. Its use is therefore limited to cases requiring explicit geometric control, where the designer has detailed knowledge of curve conformity. Nonetheless, it enables airfoil construction from generic primitives such as Bézier curves, as demonstrated by Van den Berg [57].

A more structured, aerodynamically oriented alternative is the Class-Shape Transformation (CST) method [64]. CST defines shape families through a class function (e.g. airfoils and nacelle sections[6]), while geometric variation is introduced via a shape function expressed using Bernstein polynomials B_i^n (equation (5.2)). The complete formulation, including an optional trailing-edge thickness term, is given in equation (5.3). Applied to upper and lower surfaces, CST yields a closed airfoil curve that satisfies the prescribed parametrisation requirements.

$$C_{N2}^{N1}(\psi) = \psi^{N1}[1-\psi]^{N2} \quad (5.1) \qquad S(\psi) = \sum_{i=0}^{n} A_i B_i^n(\psi) \quad (5.2) \qquad \zeta(\psi) = C(\psi)\, S(\psi) + \psi\, t_{TE} \quad (5.3)$$

For reusability, a single `CST` object is implemented to generate CST curves from shape coefficients, class exponents, trailing edge gap, and total length. The algorithm evaluates the CST formulation by sampling the non-dimensional domain $[0, 1]$ (with ψ and ζ as defined previously), computing the corresponding coordinates, and interpolating the resulting 2D point set using an `InterpolatedCurve`. A `CSTAirfoilCurve` then composes two such `CST` objects: one for the upper and one for the lower surface. After the points are combined and interpolated to form a closed airfoil curve.

For airfoil applications, the class exponents are typically fixed at $N1 = 0.5$ and $N2 = 1.0$, yielding rounded LE and sharp TE airfoil shapes. Shape coefficients provide localised control[7], with early coefficients affecting the leading edge and later ones the trailing edge, as illustrated in figure 5.5. CST offers several advantages [64]: a small number of parameters can represent a wide range of airfoils, key aerodynamic features are directly controllable such as LE radius and boat-tail angle, and smooth airfoil shapes can be accurately captured.

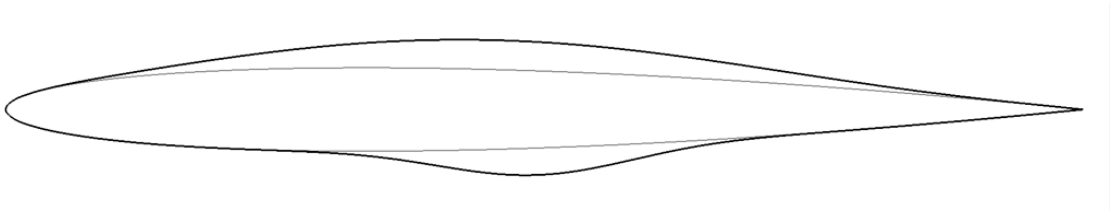


Figure 5.5: Airfoil constructed using the CST parametrisation. The baseline airfoil (grey) uses shape coefficients of 0.1 and -0.1. The perturbed airfoil (black) modifies the mid-surface coefficient on the upper surface (5 coefficients) to 0.3, and the corresponding coefficient on the lower surface (33 coefficients) to -0.6, demonstrating local controllability.

New airfoil families can be introduced by subclassing `AirfoilCurveParametrisation` and implementing the `curve` property to generate any closed, counter-clockwise curve in the domain $[0, 1]$ for `SizeableAirfoil`, or any curve in $[0, \text{chord}]$ for `NonSizeableAirfoil`. Potential extensions include

[5] The term *transform* is used here because geometric modelling operations typically combine translation, rotation, and scaling of an object. Since the ATP determines position, orientation, and chord length, it effectively applies a geometric transform to the airfoil.

[6] See section 6.4.1.

[7] Not fully local, as each Bernstein polynomial influences the full domain.

NACA-series parametrisations, Joukowski airfoils, airfoils from thickness distribution and variable camber, or more general polynomial or spline-based parameterisations [62].

5.2.4. Imported Airfoils

To satisfy SR-6.1-D7.3 - Coordinate Data Input, the tool provides a dedicated curve parametrisation for sizeable airfoils that enables direct import of airfoil geometry from 2D coordinate files. As described in section 4.3.1, the data import module supports reading `.dat` files containing 2D or 3D point sets, which are processed as a single closed group of airfoil coordinates. The corresponding parametrisation, `DirectImportedAirfoil`, includes an airfoil-data reader that processes these coordinates following the rules defined in section 4.3.1. The processed points are subsequently interpolated to construct a non-dimensional airfoil curve, completing the procedure outlined in figure 4.7. This preprocessing ensures that the imported data consistently produces valid airfoil curves that meet the requirements of section 5.2.2.

Although this parametrisation enables importing arbitrary external airfoils, the resulting curve is static and non-parametric, since it does not expose intuitive shape or design parameters. To introduce parametric control, a lightweight optimisation procedure is implemented to fit a CST-based curve to a provided set of airfoil points. This optimisation estimates the CST parameters[8] from non-dimensionalised 2D points in the interval $[0, 1]$, given a user-specified number of shape coefficients and upper and lower bounds for each parameter.

The optimisation proceeds as follows. Let the discrete set of airfoil points for one surface (upper or lower) be given by equation (5.4). A reduced CST formulation using only the shape coefficients is then defined as in equation (5.5), where the design vector $\overline{\mathbf{x}} = (A_1, \ldots, A_n)^\mathsf{T}$ contains the n shape coefficients. The fitting problem is stated as the bound-constrained least-squares objective in equation (5.6), solved using the Sequential Least-Squares Programming (SLSQP) method from SciPy[9].

$$P_i = \big(\psi_i,\, \zeta_i\big), \quad i = 1, \ldots, N, \qquad \psi_i \in [0, 1], \tag{5.4}$$

$$S(\psi; \overline{\mathbf{x}}) = \sum_{i=0}^{n} A_i \, B_i^n(\psi), \qquad \zeta(\psi; \overline{\mathbf{x}}) = C(\psi)\, S(\psi; \overline{\mathbf{x}}), \tag{5.5}$$

$$\begin{aligned} \min_{\overline{\mathbf{x}} \in \mathbb{R}^n} \quad & f(\overline{\mathbf{x}}) = \frac{1}{N} \sum_{i=1}^{N} \big[y_i - \zeta(\psi_i; \overline{\mathbf{x}})\big]^2 \\ \text{subject to} \quad & x_k^{\mathrm{L}} \le x_k \le x_k^{\mathrm{U}}, \qquad k = 0, \ldots, n. \end{aligned} \tag{5.6}$$

An example fit is presented in figure 5.6, where coordinate data from the Grumman K-1 transonic airfoil[10] are imported using the `CSTImportedAirfoilCurve` parametrisation. The optimisation is performed with $n = 5$, bounds $x_k^{\mathrm{L}} = -1$ and $x_k^{\mathrm{U}} = 1$, and a convergence tolerance of 10^{-12}. The resulting CST curve matches the direct interpolated curve with high fidelity: among 1000 comparison points, the RMS deviation is 0.000323 (0.032%), the mean deviation is 0.000256 (0.025%), and the maximum deviation is 0.000947 (0.095%), demonstrating that the CST representation accurately reproduces the imported geometry enabling the transform from static data points into parametric airfoils.

[8] For airfoils, only the shape coefficients are optimised because the class coefficients are fixed at $N_1 = 0.5$ and $N_2 = 1.0$. For nacelle cross-sections, the class coefficients are also included as design variables. See chapter 6 for the more involved CST optimisation strategy used there.

[9] `https://docs.scipy.org/doc/scipy/tutorial/optimize.html`

[10] `http://airfoiltools.com/airfoil/details?airfoil=k1-il`

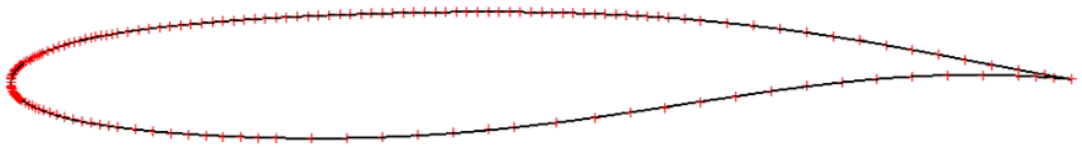


Figure 5.6: Grumman K-1 transonic airfoil: CST-fitted curve (black) and imported data (red). The optimisation yields upper-surface coefficients [0.170, 0.140, 0.160, 0.263, 0.170] and lower-surface coefficients [-0.167, -0.096, -0.223, -0.013, 0.095].

5.2.5. Airfoil on Rails and Transform Parametrisations

Analogous to the curve parametrisations in section 5.2.3, the *direct* transform parametrisation positions an airfoil using a user-defined global location and chord length. While suitable for isolated airfoil studies, it is inadequate for wings or lifting surfaces, where placement likely depend on the host or reference geometry.

Airfoils are typically used within a wing or lifting-surface context and therefore require a consistent placement relative to the wing coordinate system. In this work, the airfoil placement method of Van den Berg [57] is adopted, positioning airfoils between the leading- and trailing-edge rails using intuitive parameters such as span ratio, chord ratio, cant angle, and dihedral angle, as shown in figure 5.7. As such, four independent parameters span the full set of degrees of freedom required to position, scale, and orientate the airfoil, while the rail constraints guarantee that it remains geometrically bounded between the LE and TE rails. This approach is implemented as the *airfoil-on-rails* parametrisation via the `AirfoilOnRails` object, which internally separates chord and dihedral parametrisations.

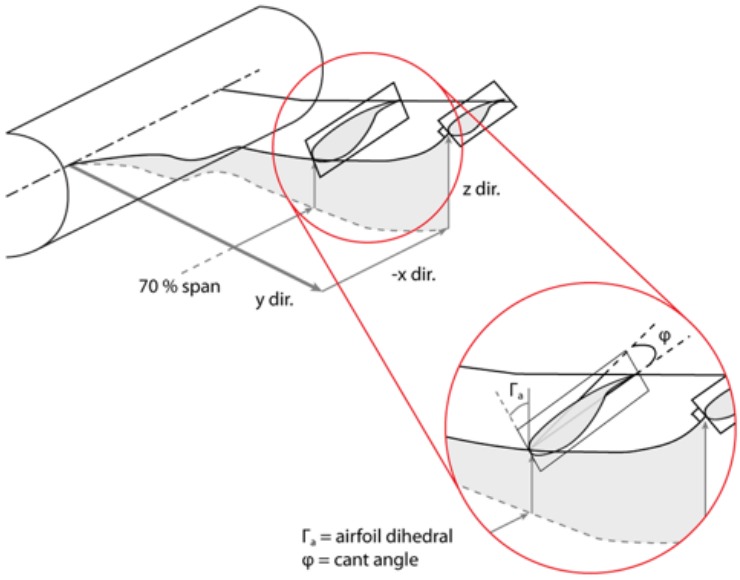


Figure 5.7: Definitions for the placement of *airfoil-on-rails* [57].

The airfoil-on-rails placement begins by computing the *reference position*: the local coordinate frame of the airfoil when both cant and dihedral angles are set to $0°$. This reference frame depends solely on the span and chord ratios and the shapes of the LE and TE rails. Using these quantities, the workflow in figure 5.8 produces intermediate LE and TE points from which the reference coordinate system is constructed. These points do not yet correspond to the final LE and TE of the placed airfoil as they serve as the geometric basis for subsequent sizing and rotations.

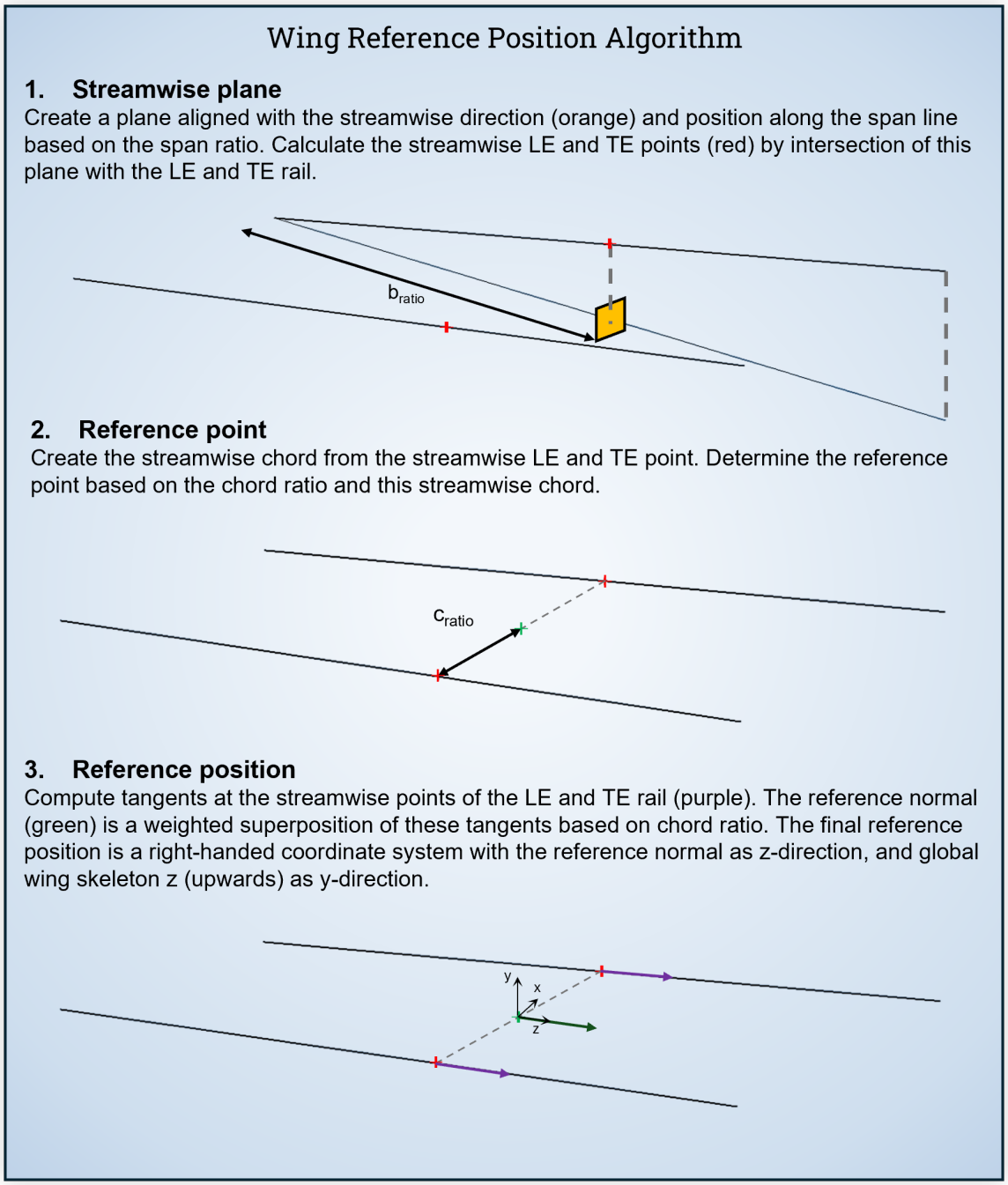


Figure 5.8: Illustrative workflow for computing the reference airfoil position for *airfoil-on-rails*, showing the determination of the coordinate system from the LE and TE rails with span line for a span and chord ratio of 0.5.

The chord parametrisation defines the LE and TE points and thus the chord line. The `DirectChord` variant computes these using the reference airfoil position, LE and TE rails, and a user-defined cant angle, defined as the rotation between the airfoil chord and the reference x-axis (see figure 5.8). Alternatively, the *streamwise* chord parametrisation aligns the chord with the wing's local streamwise direction and derives the cant angle accordingly, useful for aligning airfoils of swept wings with the flow.

The dihedral parametrisation applies the final out-of-plane rotation to the chorded airfoil. Two variants are provided: a *direct* dihedral parametrisation using a prescribed rotation, and a *z-aligned* parametrisation that aligns the airfoil's local z-axis with the wing's global z-axis.

Descriptors

The airfoil-on-rails parametrisations introduced previously depend explicitly on the geometric context defined by the leading- and trailing-edge rails. As discussed in section 4.2.1, *Descriptors* provide a structured mechanism for supplying such context while grouping all inputs like positional and shape-related parameters into a single data object.

The `AirfoilDescriptor` serves as the top-level descriptor for airfoils in a lifting-surface context, implementing `to_result()` (figure 4.6) to produce a concrete `SizeableAirfoil` from a wing skeleton. It combines an airfoil curve parametrisation with an airfoil-positioning input descriptor and forwards the required contextual information to the underlying airfoil-on-rails parametrisations, ensuring consistent evaluation.

5.3. Lifting Surface

The lifting-surface objects form the second part of the wing package and are based on the *wing* High-Level Primitive of the DARwing Multi-Model Generator developed by Van den Berg [57] and Koning [56]. The implementation in this work follows the *parametrisation-as-composition* principle, separating core logic from design-specific parametrisations and thereby supporting SR-9 - Component Flexibility and SR-9.1 - Replaceable Logic, resulting in a modular and reusable wing component.

5.3.1. Wings Skeleton and Planform

The `WingSkeleton` is the fundamental object of the lifting-surface package, defined by leading- and trailing-edge rails that may be any `Curve`. The only requirement is that both originate at local $x = 0$. A consistent local coordinate system is used, with x spanwise, y streamwise, and z upward, forming a right-handed system seen in figure 5.9.

The `WingSkeleton` defines the *spanline* as a spanwise line from the wing root with length equal to the wing span, positioned at the y-coordinate of the longer rail endpoint to ensure robustness. By projecting the rails onto the XY-plane, the skeleton fully defines the wing planform and enables estimation of key aerodynamic quantities, including the Mean Aerodynamic Chord (MAC). The chord distribution is obtained by sampling the projected rails, from which the planform area, second moment (equation (5.7) and equation (5.8)), MAC (equation (5.9)), and its spanwise location are computed. Planform rails that are shorter by span are linearly extended as needed to the end of the longer rail, and the resulting MAC is projected back onto the 3D geometry for visualisation.

$$S = \sum_{i=0}^{n-1} \frac{(c_i + c_{i+2})}{2}(x_{i+1} - x_i) \tag{5.7}$$

$$I_2 = \sum_{i=0}^{n-1} \frac{(c_i^2 + c_{i+2}^2)}{2}(x_{i+1} - x_i) \tag{5.8}$$

$$MAC = \frac{2}{S}\int_0^{b/2} c(x)^2 dx = I_2/S \tag{5.9}$$

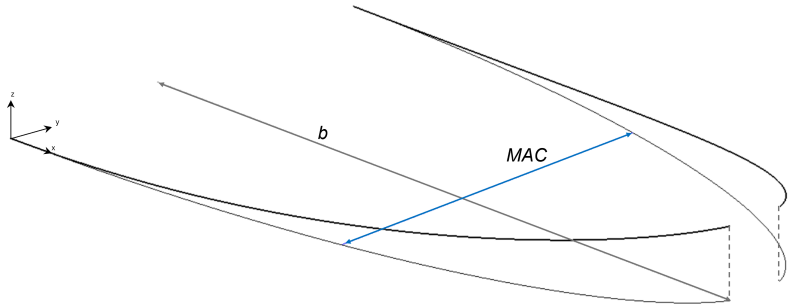


Figure 5.9: Wing skeleton for two 3D elliptical rails, with computed MAC $= 5.10\,\mathrm{m}$ and spanline length $= 9.89\,\mathrm{m}$.

A specialised form of the wing skeleton is the `WingPlanform`, which imposes additional geometric constraints:

- The local coordinate system is fixed to the global XOY system ($(0, 0, 0)$ with default orientation).
- Both rails must lie completely within the XY-plane.
- The start points of both rails must lie on the positive y-axis.
- The LE rail must begin at the XOY origin, implying the TE rail begins at a larger x-coordinate.

Because planforms are fundamental aerodynamic entities intended for use and extension (section 5.1), the `WingPlanform` incorporates two parametrisable components: a planform-rails parametrisation and a twist-axis parametrisation. A representative implementation is the `RectangularPlanform`, which constructs rails from user-specified half-span b, sweep Λ, and taper ratio λ. Two variants (implemented as sub-parametrisations) exist, generated either from aspect ratio AR or from root chord c_r, both following classical trapezoidal-wing relations. These relationships allow direct computation of rail endpoints, yielding straight-line segments for both LE and TE rails.

The twist-axis parametrisation is available in two forms: a *direct* parametrisation, and the `TwistAxisAtChord`, which defines the twist axis as a weighted average of the LE and TE rails. For example, specifying a chordwise ratio varying from 0.4 at the root to 0.6 at the tip yields a twist axis that smoothly transitions between these relative weights. As usual, since this parametrisation is dependent on context, namely the rails, a descriptor is introduced for this parametrisation to be used in a planform. An example of an instantiated planform with its computed MAC is shown in figure 5.10.

Finally, there is the positioned wing skeleton, which positions its rails based on the provided `position`. This way, the wing skeleton and thus derived lifting surfaces, can be repositioned anywhere in the model.

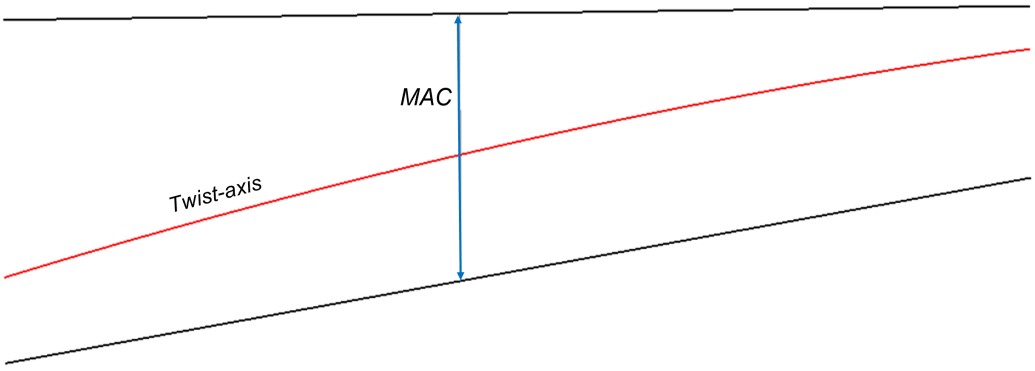


Figure 5.10: Rectangular wing planform with $b = 10\,\mathrm{m}$, $\Lambda = 10°$, $\lambda = 0.4$, $AR = 8$, and a twist axis varying from $0.25c$ at the root to $0.75c$ at the tip, resulting in a computed MAC of $2.59\,\mathrm{m}$.

5.3.2. Lifting Surface

The lifting surface is the central object of the package. The abstract `LiftingSurfaceBase` combines the `WingSkeleton` rails with supplied airfoils to construct the 3D `wing_surface`, while leaving airfoil placement method unspecified. It defines the interface for wing surface generation, identifies root and tip airfoils, closes the geometry into a solid, and integrates the capability modules (section 4.3) for export, meshing, and aerodynamic analysis.

The first concrete implementation is the `LiftingSurface` class, which extends the abstract base by adding a rail parametrisation and airfoil descriptors. As discussed in section 5.2.5 under Descriptors, airfoil parametrisations require descriptors because some, particularly airfoil-on-rails parametrisations, depend on wing-skeleton context, whereas the current rail parametrisations (section 5.3.3) do not[11].

[11] For future consistency, the author intends to wrap all parametrisations in descriptors. This will standardise the extension protocol and simplify tool usage. For context-free parametrisations the argument `to_result(context)` can simply receive a null context.

The `LiftingSurface` instantiates both rails and airfoils and constructs the `wing_surface` using a Gordon surface from section 4.4, with rails as guides and airfoils as cross-sections seen in figure 5.11.

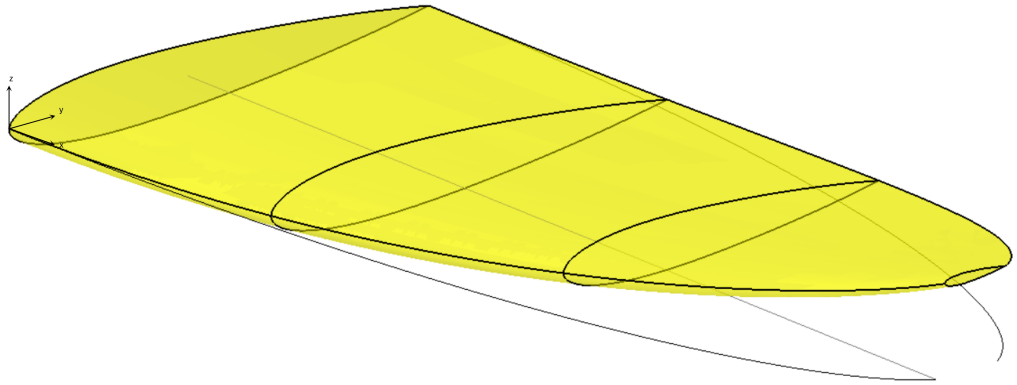


Figure 5.11: Wing skeleton from figure 5.9 instantiated as a concrete lifting surface with four NACA0012 airfoils placed along the span using the airfoil-on-rails transform parametrisation.

5.3.3. Clean Wing

The primary rail parametrisation used in this work is the *clean wing* from the DARwing tool by Koning [56] and Van den Berg [57]. In addition to planform parameters (section 5.1), it incorporates dihedral (Γ) and twist (ϵ), identified as essential for 3D wing definition and aerodynamic behaviour by Sóbester and Forrester [62].

The clean wing parametrisation begins with a 2D planform, including its twist axis, to which a dihedral distribution is applied. Subsequently, a twist distribution is defined and applied by rotating the rails about this twist axis. The full procedure is depicted in figure 5.12.

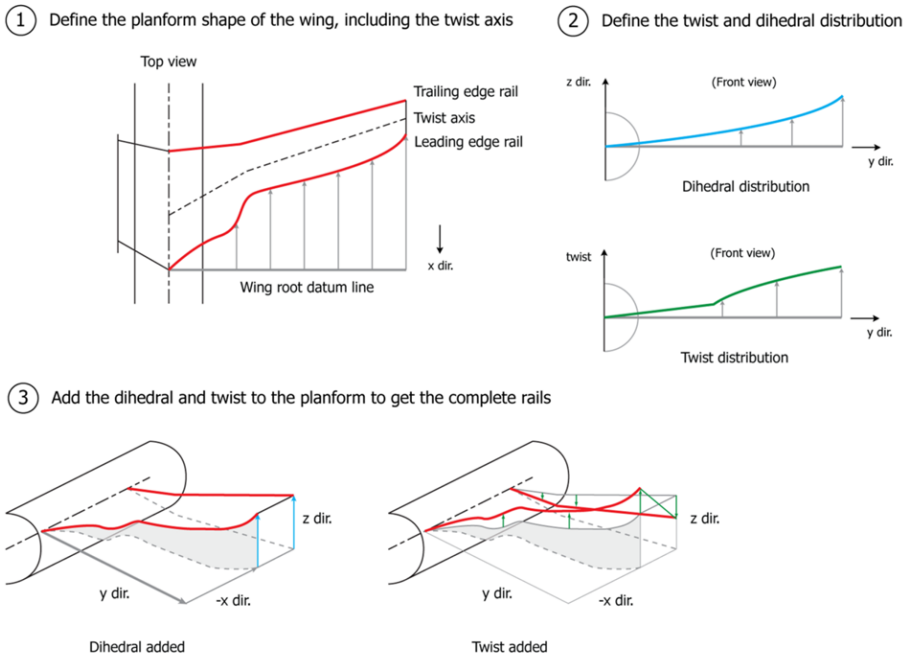


Figure 5.12: Procedure for constructing the leading- and trailing-edge rails from a planar planform [57].

Geometrically, the dihedral is applied by first scaling the dihedral distribution to the physical span of

the wing. A dihedral surface is then constructed by lofting this distribution in the streamwise direction such that it exceeds the extent of the LE, TE, and twist axis. The rails and twist axis are projected onto this surface, enabling the application of any dihedral function to a rail or twist axis of any curve type, as shown in figure 5.13.

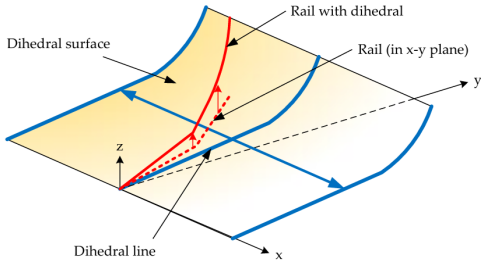


Figure 5.13: Procedure for applying dihedral to a rail [57].

The final step is the application of twist to the dihedral-modified rails. Each rail is sampled, after which each point is rotated around the twist axis according to the prescribed twist distribution, as illustrated in figure 5.14. A new interpolated curve is then created through all twisted points, resulting in rails to which both dihedral and twist have been applied.

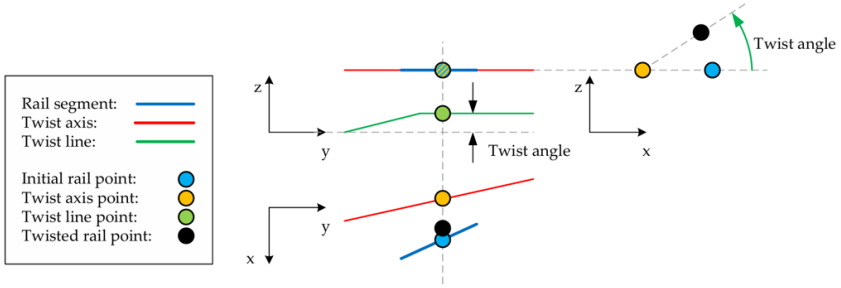


Figure 5.14: Procedure for applying twist to sampled points along a rail [57].

An example `LiftingSurface` using the clean wing parametrisation is shown in figure 5.15, where the continuous dihedral and twist distributions illustrate their strong design impact. The use of distribution functions and planform-based definition enables high-fidelity design changes with intuitive parameters.

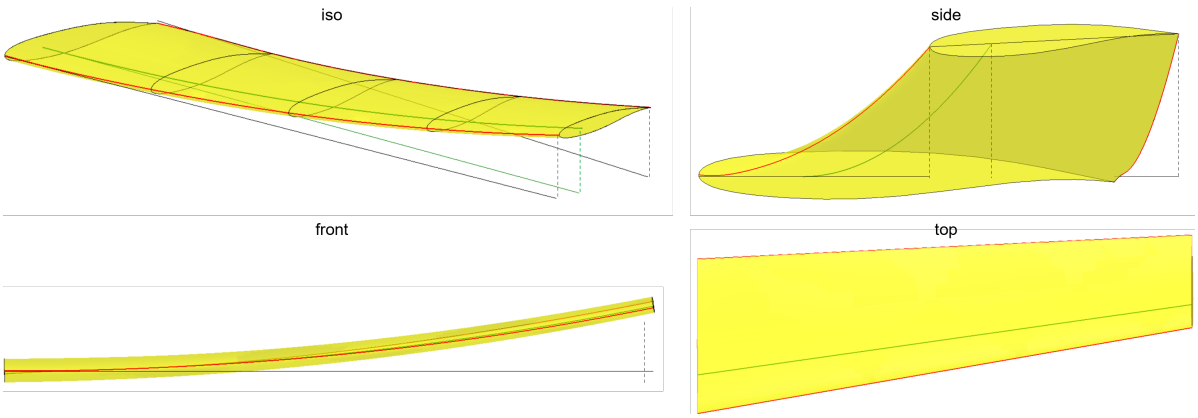


Figure 5.15: Example of a lifting surface constructed with the clean wing parametrisation, showing the rails (red), twist axis (green), airfoils (black), and 2D planform (grey).

Polynomial Functions

Because the clean wing parametrisation requires explicit dihedral and twist distributions, this work introduces a structured method for creating such functions. The `EvaluatedInterpolatedCurve` class is an interpolated curve constructed from three 1D scalar functions, e.g. $f(x) = y$, which determine the x-, y-, and z-coordinates over a specified interval. This enables the generation of geometrical curves from arbitrary analytical expressions and is later used for implementing the new *Euler*-spiral in chapter 7.

A specialisation of this class, `PolynomialCurve`, accepts parametric polynomial definitions for x, y, and z. Each is a parametrised object producing a polynomial function from a set of coefficients, following the standard definition of a polynomial:

$$p(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n.$$

5.4. Notable Improvements from Previous Work

This work transfers the clean wing parametrisation from the DARwing tool, originally implemented in GDL, into the modern ParaPy KBE framework, previously unsuccessful attempted by Heimans [65]. In doing so, it follows a new parametrisation methodology in which parametrisations are no longer intrinsically tied to their target objects. This separation removes the dependency between a component and its underlying parametrisation, and enables parametrisations to be combined, exchanged, or extended without compromising the structure or usability of the tool.

A key benefit of this approach is improved extensibility: parametrisations can be added or replaced without modifying core functionality. By keeping lifting-surface objects lightweight and assumption-free, the same architecture can support diverse geometries and use-cases beyond conventional wings, such as propeller blades, turbine vanes, and empennage surfaces, highlighting the value of decoupling components from their parametrisations as such components often require a distinct design perspective and, consequently, different parametrisations.

The lifting surfaces in this work use the Gordon surface algorithm, improving upon the loft-based approaches in DARwing, ParaWing [65], and work of Hillen [59]. While Van Luijk also employed Gordon surfaces, his implementation was limited to two profiles and guides (section 4.4.1). The present implementation supports an arbitrary number of airfoil profiles and leading- and trailing-edge guides, significantly enhancing geometric fidelity and configurability.

5.5. Matched Lifting Surface

Because nacelles for hydrogen or hybrid-electric propulsion are typically smaller than those of conventional ICE aircraft, they enable tighter and potentially more aerodynamically efficient wing-nacelle integration. As outlined in section 4.1, this first requires geometric matching of the wing and nacelle surfaces, leading to clean overlapping geometry which can then be merged using blending techniques such as filleting or the `BlendSolid` method (section 4.5). This integration demands local modification of the wing surface as blending between two shapes requires a clean intersection, a capability provided by the `MatchedLiftingSurface` through controlled surface morphing.

The deformation must satisfy two critical requirements. First, the aerodynamic shape of the underlying lifting surface must remain intact. Accordingly, the deformation is restricted to stretching or scaling the existing surface, in analogy to the scaling behaviour of profiles discussed in section 5.2.1. Second, the modification must be spatially localised: the wing geometry should remain unchanged outside a prescribed *transition region*. This ensures that the local integration with the nacelle does not propagate unintended geometric changes across the entire lifting surface.

5.5.1. Algorithm

The matched lifting-surface algorithm defines a spanwise transition region using a central *location* and *width*, within which the surface may deform while the remainder of the wing remains unchanged. Deformation is applied by scaling the surface with a *deformation factor*, locally increasing thickness through displacement of the B-spline control points (poles) away from the reference surface (figure 5.16). As B-spline surfaces depend linearly on their control points, this results in smooth deformation while preserving continuity and knot structure and enabling reshaping.

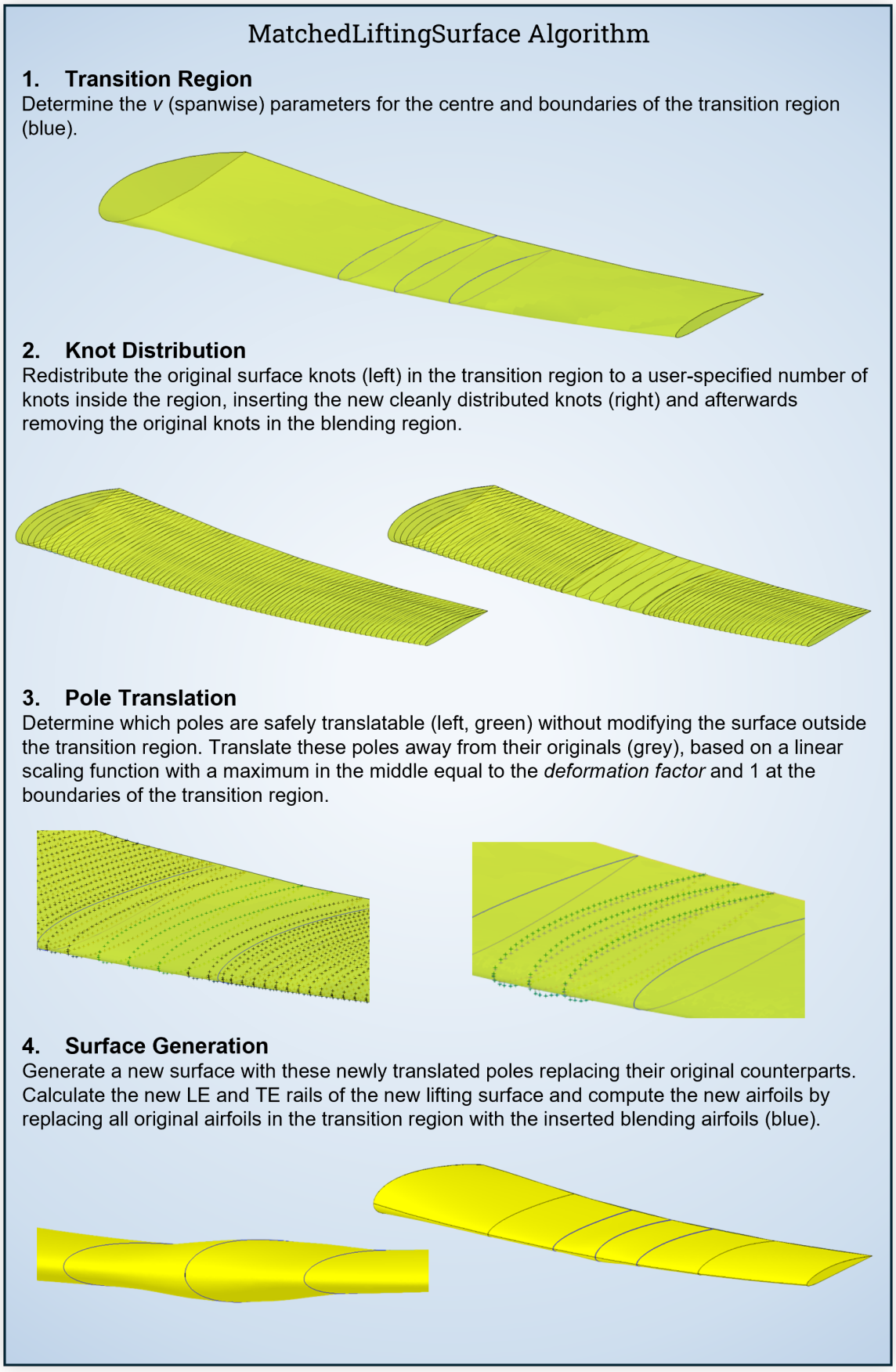


Figure 5.16: Illustrative workflow for modifying an original wing into a matched wing by providing a transition region and deformation factor.

The maximum deformation is applied at the centre of the blending region. To enable this, the knot vector within the region is redistributed to obtain an evenly spaced set of knots, ensuring that a row of control points lies exactly at the centre. This motivates step 2 of the algorithm.

A key requirement is that the wing surface must remain unchanged outside the transition region. This is enforced by exploiting a fundamental property of B-spline curves and surfaces: a control point P_i of degree p affects the curve only on the interval $[u_i, u_{i+p+1}]$, known as the support of the basis function $N_{i,p}(u)$. For surfaces, the same principle applies separately in the u and v directions. Because the transition region boundaries are placed exactly at knot locations (due to the knot redistribution), and because the degree of the surface is known, the algorithm can identify precisely which poles influence the geometry inside the transition region. Only these poles are modified; all others remain fixed, guaranteeing that the external surface geometry is preserved.

5.5.2. Inserted Airfoil

The matched lifting surface uses inserted airfoils at key locations to provide designers with a clear reference of the transition region. These inserted airfoils also serve as diagnostic tools: comparing their shapes before and after blending allows verification of shape preservation within the transition region and gives insight into the effective magnitude of the deformation. Because a B-spline surface does not pass through its control points, moving poles by the *deformation factor* does not directly translate into a proportional geometric displacement, making these airfoil samples particularly useful as seen in the figure 5.17 where the inserted airfoils at the middle of the transition in figure 5.16 are compared, highlighting the maintaining of aerodynamic shape in the transition region.

An `InsertedAirfoil` is generated by first computing a reference position on the wing, identical to the reference position construction in figure 5.8, at a user-specified span and chord ratio. Two insertion strategies are available. The *parameter-aligned* method sections the wing surface with a plane aligned with the v-parameter direction of the reference position. The *reference-positional* method sections the wing in the local XY-plane of the reference position. The resulting section curve is then used as a *direct* curve parametrisation for the inserted airfoil, making it a non-sizeable airfoil (thus subclass of `NonSizeableAirfoil`) with its chord length determined directly from the intersection curve.

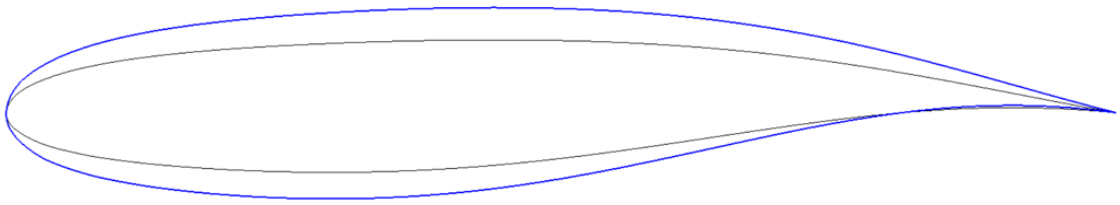


Figure 5.17: Grumman K-1 transonic airfoil as inserted airfoil into a wing with *deformation factor* of 1.5 was used (blue) and compared to an inserted wing of the unscaled wing (black), leading to an actual scale increase of 1.446.

5.5.3. Limitations and Recommendations

In summary, the `MatchedLiftingSurface` provides a functional algorithm for locally scaling a lifting surface to facilitate geometric matching with other components prior to integration or blending. However, in its current form, several limitations remain:

- The *deformation factor* specifies only the displacement applied to the control points, not the actual geometric scaling experienced by the lifting surface. As a result, the true deformation is difficult to control precisely, limiting applicability for use-cases requiring high geometric accuracy.
- Only a single deformation factor is supported for both upper and lower surfaces. Consequently, the deformation of the upper and lower sides cannot be prescribed independently. In addition, no notion of chordwise differential scaling is included.
- The algorithm does not explicitly enforce that the control points closest to the centre of the transition region coincide exactly with its parametric midpoint, causing a slight off-centring of the applied deformation.

- The method assumes that the underlying lifting surface has a clean and well-behaved parametrisation, with the v-direction aligned spanwise and the u-direction aligned chordwise. In practice, this restricts applicability to lifting surfaces with evenly distributed and geometrically meaningful parameterisations.
- No quantitative assessment, analogous to the continuity study performed for `BlendSolid` in section 4.5.4, has yet been carried out to evaluate the magnitude, accuracy, or smoothness of the resulting deformation.

These limitations indicate that the `MatchedLiftingSurface` should still be regarded as a prototype rather than a production-ready modelling component. A more rigorous performance study is strongly recommended, assessing both the accuracy of the resulting surface scaling and the fidelity with which the original shape is preserved.

Finally, it is worth noting the conceptual similarity between the current algorithm and the classical Free-Form Deformation (FFD) technique introduced by Sederberg and Parry [63]. FFD alters geometry by manipulating the control points of an enclosing deformation lattice, enabling highly accurate, smooth, and locally confined modifications where any geometry outside the lattice remains unaffected [62]. This technique has proven practical and robust in aerodynamic design workflows, for example in the optimisation of the TU Delft Flying-V by Van Luijk [58]. For these reasons, implementing a full FFD module in ParaPy and applying it to the transition region is strongly recommended as a future improvement to the `MatchedLiftingSurface`.

5.6. Capability Module Implementations

The `LiftingSurface` and `MatchedLiftingSurface` constitute the two concrete classes in this package that generate full 3D geometry and play a central aerodynamic and geometric role both within the system assembly and as standalone components. Consequently, both classes incorporate the aerodynamics capability module (section 4.3.4) and the automatic meshing module (section 4.3.3), ensuring that any instantiated lifting surface, independent of its parametrisation or intended use-case, is directly suitable for aerodynamic assessment.[12]

Furthermore, all components introduced in this chapter implement the `GeometryExportBase` interface from section 4.3.2, enabling export to STEP format conforming to SR-1.2-D2 - STEP Export. This ensures that each object, regardless of its complexity or parametrisation, integrates cleanly into the wider system model and downstream workflows.

[12] Although this thesis considers the lifting surface primarily as a wing within the system assembly, the underlying design deliberately supports broader application domains such as propeller blades, turbine vanes, or tail surfaces, as discussed in earlier sections.

6

Nacelle

This chapter introduces the nacelle as the primary housing for the fuel-cell subsystems and propeller, and as the second major component of the system assembly. Section 6.1 presents an overview of the nacelle package, including the spinner (section 6.2), internal subsystems (section 6.3), and the nacelle geometry itself, comprising cross-sectional definitions (section 6.4), the main nacelle body (section 6.5), and the matched nacelle used for geometric matching (section 6.6). Wing–nacelle integration is discussed in section 6.7, followed by the implementation of the associated capability modules in section 6.8.

6.1. Architectural Overview

Previous nacelle parametrisation work has largely focused on ICE-powered turbofan configurations, typically using podded layouts, as exemplified by the models of Proesmans [66] and Otting [67]. These parametrisations are tailored to conventional cowl shaping and are therefore poorly suited to propeller- or fuel-cell nacelles, which differ fundamentally in topology, aerodynamics, and integration requirements. In contrast, propeller and fuel-cell nacelles allow greater conceptual freedom, with lower inflow demands and no established external design standard (chapters 1 and 2). Consequently, this work adopts a flexible modelling framework, representing nacelles as lofted bodies defined by cross-sections and guides following Sóbester and Forrester [62]. This approach forms the geometrical basis in `AbstractNacelle` and uses the Gordon surface algorithm (section 4.4) with an overview of the resulting architecture is given in figure 6.1.

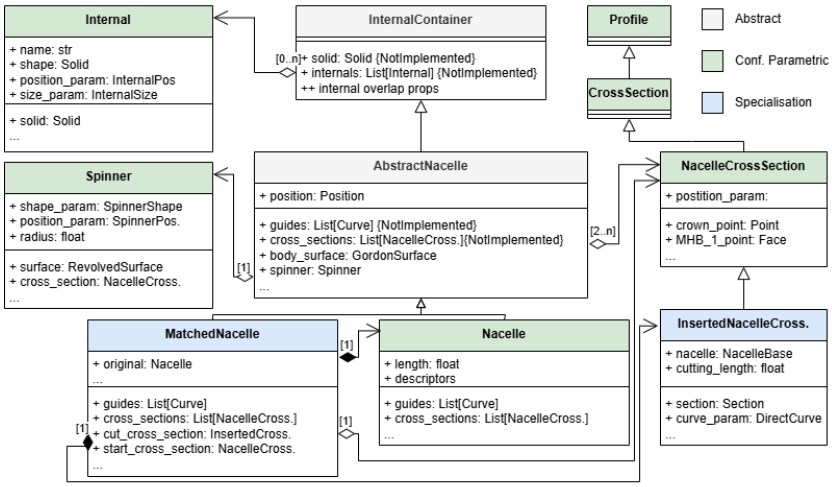


Figure 6.1: UML representation of all abstract (gray), configurable parametric (green) and specialised (blue) nacelle-related classes, illustrating the structural and functional relationships.

In addition to its own geometry, a nacelle contains two additional subsystems: a spinner and internal components. The `Spinner` defines the aerodynamic hub around the propeller roots (section 2.3), while internal systems are represented by `Internal` objects within an `InternalContainer`. Both are implemented as *configurable parametric* objects, allowing the containing components and thus the full nacelle model to adapt to new design philosophies and future aircraft architectures.

Several specialised cross-section types are introduced to support nacelle modelling. These extend the generic `CrossSection` class from section 5.2.1 by enforcing additional geometric constraints specific to nacelles. The most notable specialisation is the `InsertedNacelleCrossSection`, which mirrors the behaviour of the `InsertedAirfoil` from chapter 5 but adapts it for nacelle geometry, including independent scaling in both the local x- and y-directions.

The primary concrete nacelle class is `Nacelle`, a configurable parametric object whose guides, cross-sections, and spinner are all generated from parametrisations wrapped in descriptors.[1] This class produces the full nacelle geometry and is intended as the main workhorse for nacelle modelling within the overall system architecture.

The primary concrete implementation of the abstract nacelle is the configurable parametric `Nacelle`, which uses a guide parametrisation (wrapped in a descriptor) together with cross-section and spinner descriptors to construct the full geometry. A second implementation, `MatchedNacelle`, enables local surface modification analogous to the `MatchedLiftingSurface` (section 5.5). However it does not employ a surface deformation-like strategy but works by replacing a portion of the nacelle surface with a smoothly transitioning patch toward a target cross-section, defined using a *specialised* `InsertedNacelleCrossSection`.

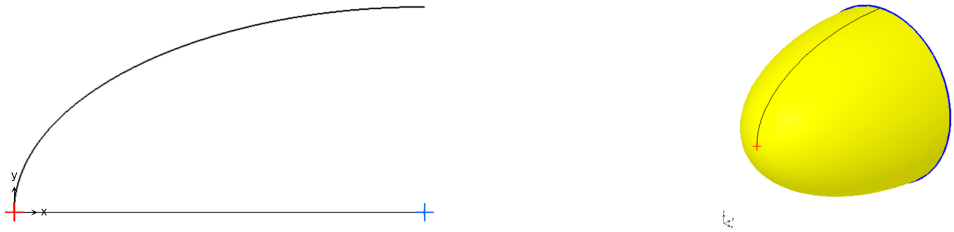
6.2. Spinner

As introduced in section 2.3, the spinner is the aerodynamic fairing surrounding the propeller hub and has a substantial influence on both propeller aerodynamics and the interaction between the propeller and the nacelle. From the preliminary parametrisation shown in figure 2.4, it follows that the complete spinner geometry can be defined from its 2D side-view profile.

The *configurable parametric* `Spinner` is defined by a single aerodynamic profile curve, `outer_curve` (κ_s in figure 2.4), generated by a shape parametrisation, scaled by the spinner radius, and positioned via a *position parametrisation*. Revolving this curve about the `center_line` produces the 3D spinner outer-mold surface (figure 6.2b). The swept edge of this surface is then converted into a `NacelleCrossSection` to provide the aft interface for nacelle construction (section 6.5).

To ensure that any chosen shape parametrisation produces a valid spinner geometry, the base curve must satisfy a small set of geometric prerequisites:

- The curve starts at the local origin $(0, 0, 0)$.
- The curve lies entirely within the positive XY-quadrant.
- The curve is non-self-intersecting.



(a) Spinner profile sketch showing the outer curve (thick black), the centreline (black), the spinner start (blue), and the top of the outer profile curve (red).

(b) Iso view of the spinner surface, highlighting the outer profile curve (black), the revolved cross-section (blue), and the spinner tip (red).

Figure 6.2: Spinner with elliptical shape parametrisation, medium AR (=2) and radius of 1 m.

[1] See section 4.2.1 for a detailed explanation of `Descriptor` objects and their interaction with parametrisations.

Area-ruling dimples (see section 2.3 under Spinner Design Considerations) are not included in the present implementation, as the tool targets FC-powered aircraft operating at relatively low Mach numbers, for which such features are likely aerodynamically sub-optimal. Nevertheless, the `Spinner` object is intentionally general-purpose and not restricted to tractor FC configurations, making dimple modelling a worthwhile extension for future applications requiring transonic optimisation.

For potential future implementations, the following modelling strategy is recommended. Since the dimples do not correspond to a 2D profile deformation but are inherently three-dimensional, they should not be built into the base profile. Instead, parametrised spheroids can be placed to intersect the spinner surface slightly, after which Boolean subtraction yields the sharp cavities, or dimples. These cavities can then be smoothed using fillets. The spheroids should be positioned along outward surface normals to maintain geometric consistency.

6.2.1. Shape Parametrisations

Two shape parametrisation strategies are currently provided. The first is the *direct* parametrisation, which gives the designer full freedom to prescribe any valid spinner profile curve. This comes with reduced robustness and limited intuitive control but remains essential for unconventional designs.

The primary parametrisation implemented is the *elliptical spinner*, in which an `EllipticalArc` is defined from a single aspect ratio (AR = l_s/d_s), while adhering to the geometric requirements listed above. Example of ellipical spinner can be seen in figure 6.2. Elliptical spinners are widely used in the literature [29, 30, 68].[2]

Although not implemented in this version, several extensions are recommended for future work, including conic spinners [30, 68], ellipsoids with boattails [29], blunt spinner tips [69], and flat spinners [29]. These types offer more flexibility for either tractor or pusher configurations and can be incorporated easily due to the modular nature of the shape-based parametrisation.

6.2.2. Positioning

At present, spinners are only used in the context of tractor-configured nacelles. Accordingly, a single positioning parametrisation is implemented: `TractorSpinnerPosition`. This parametrisation takes the nacelle centreline as input and positions the spinner at its forward end, aligned with the direction of the centreline. As this parametrisation requires contextual information from its parent nacelle, *descriptors* are employed to supply this context appropriately.

In the current implementation, the spinner orientation always coincides with the nacelle centreline, and no parameters are included to introduce an offset angle. However, such extensions can be added straightforwardly by subclassing or extending this position parametrisation.

For alternative configurations, such as pusher arrangements discussed by Mota Mera [29], or standalone spinners used outside a nacelle context, the general `SpinnerPosition` interface can be extended to produce new parametrisations with suitable orientation and placement parameters.

6.3. Internal Subsystems

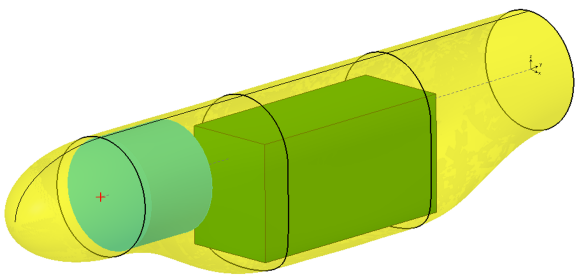
As identified in section 1.4 and section 1.4.1, no existing parametric framework captures the coupling between external WNI geometry and internal fuel-cell subsystems. This work addresses that gap by enabling internal subsystems to be embedded within higher-level components. Within the current scope, these subsystems are modelled as generic volumes inside a parent component, without assumptions on function or architecture, allowing representation of any system discussed in section 2.1. Each `Internal` object contains basic metadata (name, description and category) and a non-dimensional solid defining its topology (figure 6.3b), which is scaled to a target volume via the `InternalSize` parametrisation, preserving flexibility across the diverse sizing approaches reported in the literature report [24].

`Internal` objects may be placed in any `InternalContainer`. This abstract class provides two core capabilities: it defines a single `solid` representing the geometry of the parent container, and it performs preliminary geometric consistency checks, such as detecting subsystem overlap or identifying portions of a subsystem that extend beyond the container volume. In the current version of the tool, only

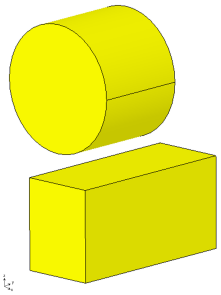
[2] See [24] for an overview of spinner parametrisation trends in the literature.

`AbstractNacelle` inherits from this class, making the nacelle the sole component capable of hosting internal subsystems.

An illustrative example is shown in figure 6.3, where the *baseline* nacelle contains an electric motor and fuel-cell stack sized for an approximate 4 MW power requirement using the specific-power values from chapter A[3]. This demonstrates that the nacelle volume is of a realistic order of magnitude for MW-class FC aircraft and highlights the value of including internal subsystems, which can be readily extended to other components such as lifting surfaces (chapter 5), integration geometries (section 6.7 and section 7.5), or combined system assemblies, in future work.



(a) *Baseline* nacelle containing two internal subsystems: an electric motor (cyan; 400 kg, 0.5 m^3) and a fuel cell system (green; 1333.33 kg, 1.6 m^3).



(b) Non-dimensional shapes (1 m^3) used for the electric motor and FC subsystem.

Figure 6.3: Internal subsystems embedded in the *baseline* nacelle design for a total power requirement of 4 MW, using estimates of 10 kW/kg for the electrical motor and 2.5 MW/m^3 for FC system.

6.3.1. Sizing

As concluded in section 2.1.1, this thesis does not attempt to introduce new subsystem-specific sizing methodologies, nor does it attempt to model the aerodynamic or thermodynamic coupling between FC subsystems and external geometry. For this reason, the implemented sizing parametrisations are intentionally simple while remaining sufficiently flexible and allowing for extendability of designing knowledge.

Two subsystem sizing approaches are supported: direct volume-based sizing via a `VolumeSize` object, useful when volumes are known or for assessing maximum allowable space, and mass-based sizing using density scaling, which reflects the preliminary subsystem data typically available in early-stage FC design (chapter A). Both approaches are illustrated in figure 6.3a, with the electric motor sized by mass and the fuel-cell stack by volume. Extending these sizing methodologies in line with the literature [24] is left to future work as new implementations of subsystem sizing parametrisations.

6.3.2. Positioning of Internals

Internal subsystems can be positioned either by direct specification of a global location or relative to another component using the `InternalRelativeTo` parametrisation. The latter enables placement with respect to a parent or other subsystems, preserving spatial relationships within an assembly (figure 6.3a). As this depends on reference geometry, contextual information is provided through `Descriptors` (section 4.2.1), which are therefore inputs to the `InternalContainer`.

The split of internal objects, sizing and positing parametrisations enables flexibility for future users to inject such workflow and knowledge into the tool without making any architectural changes and thus any prior knowledge about the tool's internals. Dependencies and complex interaction in these methodologies are expected, further promoting the use of descriptors for internals. To conclude, the generic parametrisations and descriptors promotes scalability and utility of the tool in its current version as well as future versions.

[3] No dedicated FC sizing methodology is implemented and thus the assumptions are illustrative only.

6.4. Nacelle Cross-sections

As stated in section 6.1, the nacelle geometry consists of 2D planar cross-sections connected with guides. For that reason, a large part of the nacelle package consists of these cross-sections specifically for nacelles, namely the `NacelleCrossSection` object extending the generic cross-section introduced in section 5.2.1. This extension comes with additional requirements for the unpositioned and scaled base curve:

- The start of the curve is exactly on the local positive y-axis (x=0).

This requirement enables a consistent definition of nacelle cross-section reference points. Following conventional practice, each cross-section is characterised by the *crown*, *keel*, and two *Maximum Half Breadth (MHB)* points (figure 6.4). The crown is defined as the first point of the curve, ensuring a consistent origin and orientation, while the keel is the point with minimum local y-coordinate relative to the crown. The MHB points are defined at the centroid y-location of the enclosed face, with left and right assignment following the counter-clockwise orientation requirement (section 5.2.1). These reference points are then used to compute the cross-section height, width, and resulting aspect ratio AR.

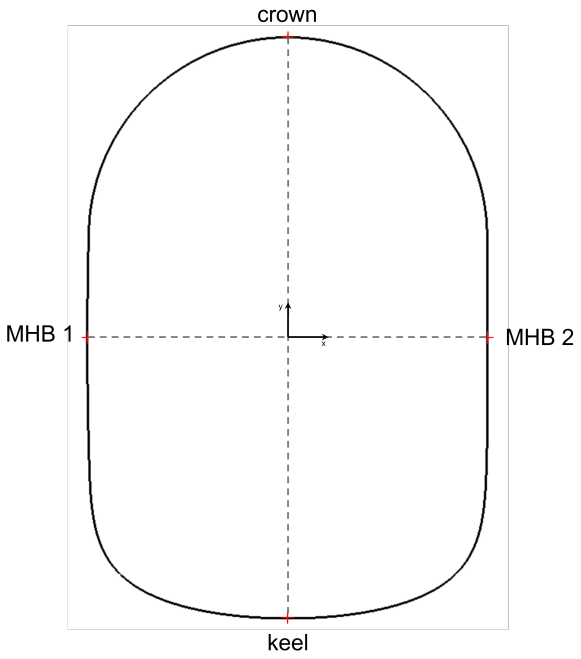


Figure 6.4: Computed nacelle cross-section points for a CST-based cross-section ($N_u = 0.5$ and $N_l = 0.1$ with uniform shape parameters) generated within the tool, showing the crown, MHB, and keel points (red).

6.4.1. Curves for Nacelle Cross-sections

Identical to the airfoil and profile definitions, the most basic curve parametrisation for nacelle cross-sections is the *direct* parametrisation, which constructs the cross-sectional curve from any given geometric curve. Although flexible, this approach is generally unsuitable in practice: because nacelle cross-sections impose additional requirements, most notably the crown-point condition, the imported or manually provided curve often requires reparametrisation (e.g. start point shifting) before becoming valid.

The primary and recommended parametrisation for nacelle cross-sections is the CST method introduced in section 5.2.3, which provides a flexible and expressive representation for aerodynamic bodies using the identical class, shape, and combined formulations (equation (5.1), equation (5.2), equation (5.3)). Unlike airfoils, nacelle geometries span a wider range of shapes, and the class exponents N_1 and N_2 are therefore treated as user-defined parameters (figure 6.5), enabling representation of cross-sections from circular to highly non-circular forms. Upper and lower CST curves are generated

by sampling $\psi \in [0, 1]$, after which the resulting non-dimensional cross-section is scaled using the `width` property of the `NacelleCrossSection`.

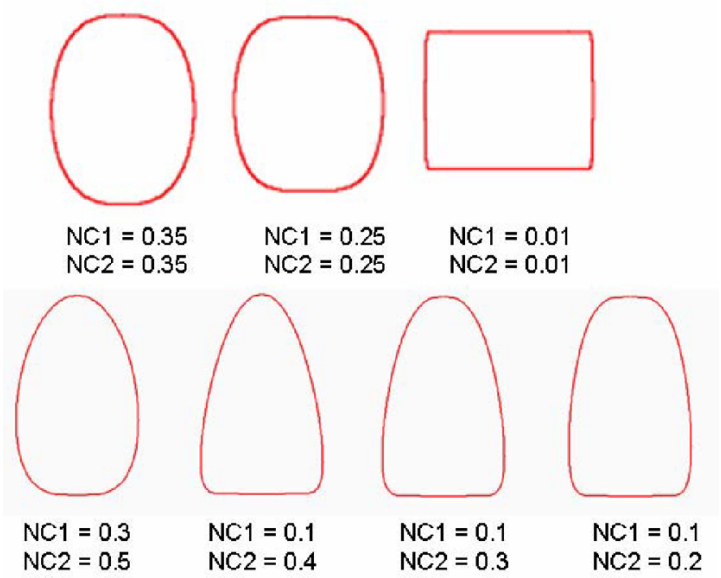


Figure 6.5: Examples of nacelle cross-sectional families defined using CST, where $NC1$ and $NC2$ denote the upper and lower class factors, respectively [64].

6.4.2. Imported Cross-sections

To satisfy SR-6.1-D7.3 - Coordinate Data Input, the tool includes curve parametrisations capable of importing nacelle cross-sections directly from 2D coordinate data stored in `.dat` files. This process mirrors the imported-airfoil workflow of section 5.2.4: coordinates are read and pre-processed according to the procedure in section 4.3.1 and figure 4.7, after which `DirectImportedNacelleCrossSection` constructs an interpolated geometric curve.

While sufficient for fixed, non-parametric cross-sections, these directly imported curves lack the flexibility required for parametric design. To introduce parametric control, the tool employs a CST-based curve-fitting optimisation similar to the approach described for airfoils in section 5.2.4, but extended for nacelle-specific considerations. The optimisation estimates CST parameters from non-dimensionalised data points in $[0, 1]$, using a user-specified number of shape coefficients and parameter bounds.

For nacelles, however, the class of the cross-section is often unknown at initialisation. The class factors N_1 and N_2 are therefore included as design variables leading to a design vector of $\overline{\mathbf{x}} = (A_1, \ldots, A_n, N_1, N_2)^{\mathsf{T}}$ containing the n shape coefficients plus the two class coefficients. The discrete input points for the upper and lower surfaces are defined as equation (6.1), leading to the new reduced CST form for nacelle cross-section optimisation in equation (6.2).

$$P_i = \big(\psi_i,\, \zeta_i\big), \quad i = 1, \ldots, N, \qquad \psi_i \in [0, 1], \tag{6.1}$$

$$S(\psi; \overline{\mathbf{x}}) = \sum_{i=0}^{n} A_i \, B_i^n(\psi), \qquad C_{N2}^{N1}(\psi; \overline{\mathbf{x}}) = \psi^{N1}[1 - \psi]^{N2}, \qquad \zeta(\psi; \overline{\mathbf{x}}) = C(\psi)\, S(\psi; \overline{\mathbf{x}}), \tag{6.2}$$

Because nacelle cross-sections may be initialised from sparse data (N small), designers may choose a comparatively large number of shape coefficients (n large) to retain modelling flexibility. However, if $n > N$, unconstrained optimisation can result in highly oscillatory (“wobbly”) curves due to excessive degrees of freedom. To mitigate this, a *wobbliness* measure is introduced in equation (6.3).

$$W(\overline{\mathbf{x}}) = \int_0^1 \left(\frac{\mathrm{d}^2 S(\psi; \overline{\mathbf{x}})}{\mathrm{d}\psi^2} \right)^2 \mathrm{d}\psi. \tag{6.3}$$

This wobbliness measure penalises large curvature fluctuations in the shape function. The class function is excluded from this measure, as its derivatives diverge[4] at $\psi = 0$ and $\psi = 1$. Introducing this new constraint consequently leads to the full optimisation defined in equation (6.4), where w_c is the designer-selected wobble limit.

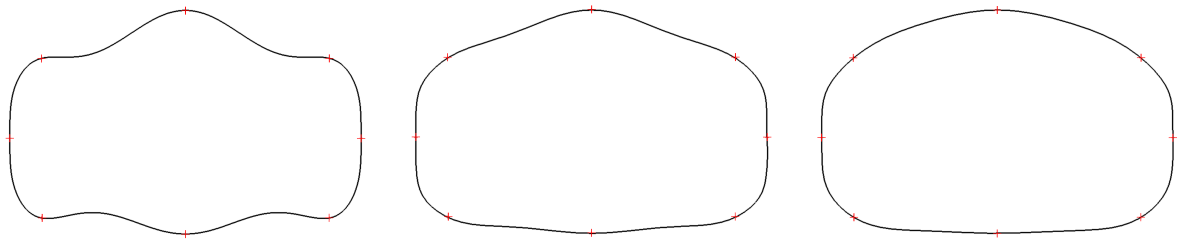
$$\begin{aligned} \min_{\overline{\mathbf{x}} \in \mathbb{R}^{n+2}} \quad & f(\overline{\mathbf{x}}) = \frac{1}{N} \sum_{i=1}^{N} \left[y_i - \zeta(\psi_i; \overline{\mathbf{x}}) \right]^2 \\ \text{subject to} \quad & g(\overline{\mathbf{x}}) = w_c - W(\overline{\mathbf{x}}) \ \geq \ 0, \\ & x_k^{\mathrm{L}} \leq x_k \leq x_k^{\mathrm{U}}, \qquad k = 0, \ldots, n+2. \end{aligned} \tag{6.4}$$

The resulting optimised nacelle cross-sections for three different wobble-constraint levels are shown in figure 6.6. All runs use a total of $n = 16$ CST shape coefficients (upper and lower each) and $N = 5$ data points per side, corresponding to the following non-dimensionalised point set:

$$(0.00,\, 0.40),\ (-0.45,\, 0.25),\ (-0.55,\, 0.00),\ (0.00,\, -0.30),\ (0.45,\, -0.25),\ (0.55,\, 0.00),\ (0.45,\, 0.25).$$

These points represent a generic nacelle-like profile and allow the optimisation to reconstruct both sharp and smooth shapes depending on the chosen wobble constraint w_c.

The unconstrained optimisation ($w_c = 10000$) produces a highly oscillatory solution, demonstrating the tendency of CST to overfit when $n > N$ seen in figure 6.6a. Imposing moderate or strict wobble limits leads to progressively smoother and more physically meaningful results, while still matching the intended shape as seen from figure 6.6b and figure 6.6c. The assessment of accuracy of this CST optimisation can be found in section 5.2.4, where this optimisation process is assessed and verified for CST airfoils.



(a) Unconstrained optimisation ($w_c = 10000$), shape parameters $N_u = 0.253$, $N_l = 0.267$.

(b) Moderate wobble constraint ($w_c = 20$), shape parameters $N_u = 0.207$, $N_l = 0.225$.

(c) Strict wobble constraint ($w_c = 5$), shape parameters $N_u = 0.205$, $N_l = 0.229$.

Figure 6.6: Optimised CST nacelle cross-sections for increasing wobble constraint levels ($n = 16$, $N = 5$ data points per side seen in red).

6.4.3. Nacelle Cross-sections Positioning

Analogous to airfoils, the most basic positioning approach for nacelle cross-sections is the *direct* positioning parametrisation. While this method offers full freedom to place and orient a cross-section anywhere in the model, it comes at the cost of losing geometric coupling with the parent nacelle, making it less suitable for consistent parametric modelling.

The primary and most robust positioning strategy is the *nacelle middle-line* parametrisation which shows very close resemblance with the *airfoil-on-rails* from section 5.2.5. The `NacelleMiddleLinePositioning` places each cross-section along the nacelle's middle line and then applies optional relative translations in the local x- and y-directions, as illustrated in figure 6.7. In addition, the parametrisation introduces

[4] The class-function derivatives contain terms proportional to $\psi^{N_1 - 1}$ or $(1-\psi)^{N_2 - 1}$, which approach infinity at the endpoints for typical values of $N_1, N_2 \in [0, 1]$.

a *cant* angle (rotation about the local y-axis) and a *dihedral* angle (rotation about the local x-axis), allowing controlled orientation of the cross-section along the nacelle body. Because the middle line is defined as a straight axis from nacelle start to spinner tip, the required context for this parametrisation reduce to the middle line itself and the nacelle's upward (local z) direction, both supplied via descriptors.

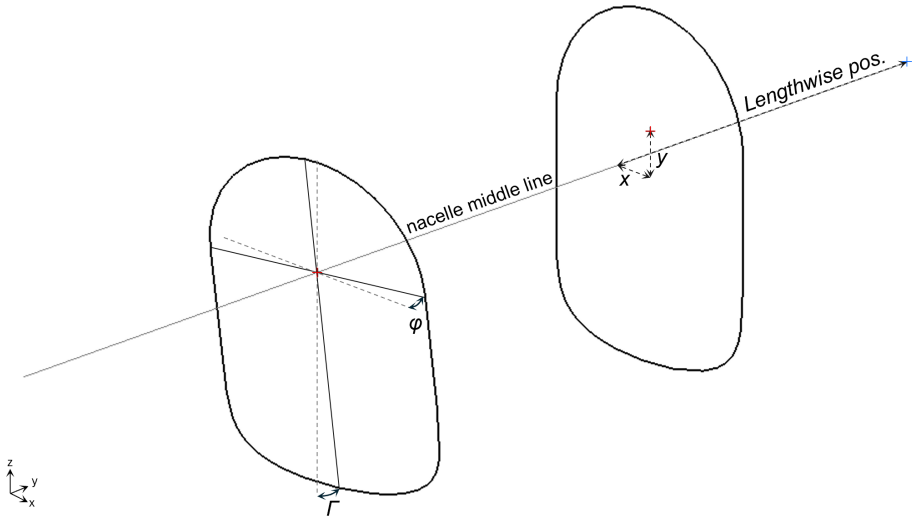


Figure 6.7: Positioning of two nacelle cross-sections along the nacelle middle line, including relative x- and y-offsets, cant angle (φ), and dihedral angle (Γ). The local cross-section origin is shown in red, the middle line in grey, and the nacelle start in blue.

6.5. Nacelle

As outlined in section 6.1, the nacelle geometry is constructed from a series of planar cross-sections connected by lateral guides. The `AbstractNacelle` defines the interface for these entities, enabling multiple concrete implementations to share a consistent structure. This mirrors the philosophy used for lifting surfaces in figure 5.1: both a flexible, *configurable parametric* implementation and a more constrained, *specialised* variant are implemented. The body surface is generated as a Gordon surface (section 4.4) interpolating between the guides and cross-sections, after which a closed shell and solid are formed using the spinner surface and the aft termination face.

The preliminary nacelle layout introduced in figure 2.3 highlights two additional considerations for the parametrisation: a nacelle inclination angle and a longitudinal centreline, here referred to as the `middle_line`. This line spans the length of the nacelle, from the wing attachment to the spinner tip, and provides an intuitive spatial reference for positioning cross-sections, greatly simplifying orientation control and overall geometric consistency as shown in section 6.4.3. Moreover, the nacelle's position and orientation are defined by positioning this middle line (see figure 6.8), allowing axial and vertical offsets from figure 2.3 to be encoded directly. Consequently, the nacelle's local coordinate system is identical to its position frame, which establishes the *upwards* direction used for all contained components, including cross-sections, guides, spinner, and internal subsystems.

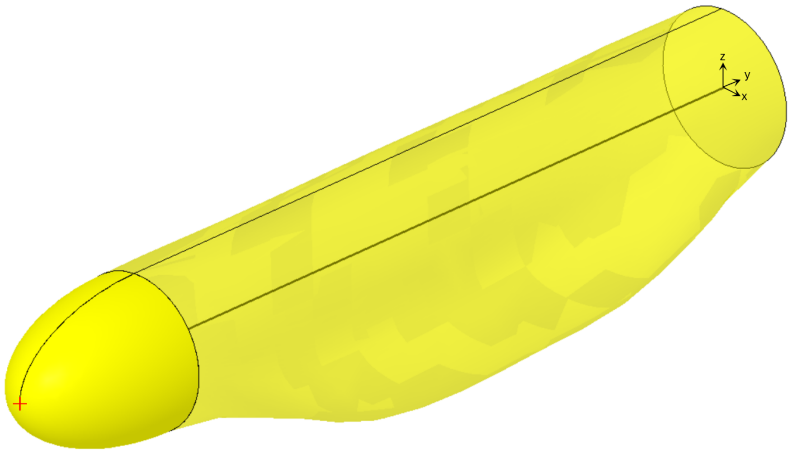


Figure 6.8: Concrete nacelle from the *baseline* design showing the nacelle end (red), the middle line (bold black), the nacelle's local coordinate system (and thus start), and the spinner (solid yellow).

The spinner and internals are left as abstract in `AbstractNacelle` because they are instantiated concretely in `Nacelle` and passed through unchanged in the specialised blended nacelle (see section 6.6). The base class also inherits from `InternalContainer`, allowing automatic geometric consistency checks including overlap detection and quantifying subsystem volume lying outside the nacelle geometry.

6.5.1. Concrete Nacelle

The concrete `Nacelle` class realises the abstract spinner, internals, cross-sections, and guides through input descriptors. Descriptors are essential here because each component's parametrisation depends directly on the nacelle context:

- Cross-sections depend on the nacelle middle line (section 6.4.3).
- Guides require the cross-sections as their construction context (section 6.5.2).
- The spinner position is entirely defined by the middle line (section 6.2.2).
- Internals may be positioned relative to any nacelle-based reference (section 6.3.2) and do not have any coupling with nacelle geometry.

The placement of the `Nacelle` itself follows the general position parametrisation described in section 6.7.1.

6.5.2. Nacelle from Cross-sections

The primary guide parametrisation is the *nacelle-from-cross-sections* method, in which guides are derived directly from the positioned cross-sections. As defined in section 6.4, each cross-section provides four characteristic points: the crown, keel, and two MHB points. The parametrisation constructs four corresponding guides by smoothly interpolating each set of points. Although this resembles a conventional smooth loft through the sections, it offers finer geometric control by allowing tangency specifications at each intersection point.

A guide may be constrained in four distinct ways:

1. Prescribing an explicit direction.
2. Enforcing streamwise (local y-axis) alignment.
3. Enforcing spinner tangency at the forward end of guide.
4. Leaving the tangent free for the interpolation algorithm to handle smoothly.

These options provide designers with localised and intuitive control over nacelle shaping without the necessity to explicitly formulate exact guide directions. An illustration of the keel guide under different tangency specifications is shown in figure 6.9.

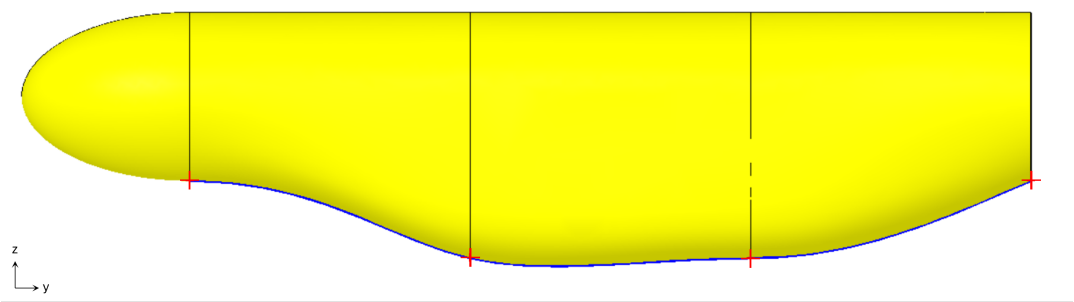


Figure 6.9: Side view of the nacelle-from-cross-sections parametrisation of *baseline* design, showing keel guide behaviour (blue) under different tangency options (from left to right): spinner tangent, smooth, streamwise (local y), and smooth.

6.6. Matched Nacelle

As discussed throughout this thesis, FC-powered and novel propulsion concepts enable improved wing–nacelle integration. The first step is geometric matching of the nacelle to the wing (section 4.1), after which the integration surface can be formed using blending techniques such as classical filleting or the `BlendSolid` method (section 4.5). Following the same philosophy as section 5.5, the `MatchedNacelle` enables targeted modification of a conventional nacelle to conform to the wing geometry.

Because the nacelle its starting point resides inside the wing (section 6.7.1), only the starting portion (from nacelle's local coordinate system in figure 6.8) must be altered. Therefore, the `MatchedNacelle` replaces a user-specified portion of the forward segment of the original nacelle while ensuring that the geometry outside of this region remains exactly unchanged. This forms the first key requirement of the integration namely that the unmodified nacelle geometry must be preserved perfectly from the cut location onwards.

A second requirement follows from the physical relationship between wing and nacelle at the interface. After blending, the modified starting section is expected to reside fully inside the wing. Consequently, its precise local shape does not need to match the original nacelle cross-section identically as it must merely capture the same overall scale and form. Additionally, because the original nacelle start may lie outside or offset from the wing contour, the new blended cross-section must be repositioned so that it lies completely within the wing volume at a user-defined location in all three dimensions. This defines the second requirement namely the new starting cross-section must represent a scaled approximation of the original nacelle shape, shifted to any new user-defined position position.

6.6.1. Algorithm

An overview of the `MatchedNacelle` workflow is presented in figure 6.10. The procedure begins with the construction of a new nacelle starting cross-section at a user-specified location. This section is represented by an ellipse with user-defined width and height. An elliptical form is selected because nacelle cross-sections are characterised by four critical circumferential reference points (crown, MHB and keel points), as described in section 6.4, which directly correspond to the widths and heights of an ellipse. This makes the ellipse a robust generic approximation for a wide class of nacelle shapes.

Next, the original nacelle is cut at a specified transition location using an `InsertedNacelleCrossSection` (section 6.6.2). The original nacelle guides are then trimmed up to this inserted section starting at the nacelle end and extended to intersect with the new starting cross-section. Because nacelle geometry is defined using four consistent guide curves aligned to the nacelle's circumferential reference points, the correspondence between original and new guide endpoints is inherently preserved.

With these geometric relationships established, a new Gordon surface (section 4.4) is constructed using the kept original cross-sections, the inserted cut section, and the new starting section, all connected by the updated guides. A new middle line and start location are simultaneously defined, while preserving all internal content such as the spinner and internal subsystems. This results in a fully updated `MatchedNacelle` that remains consistent with the original downstream geometry.

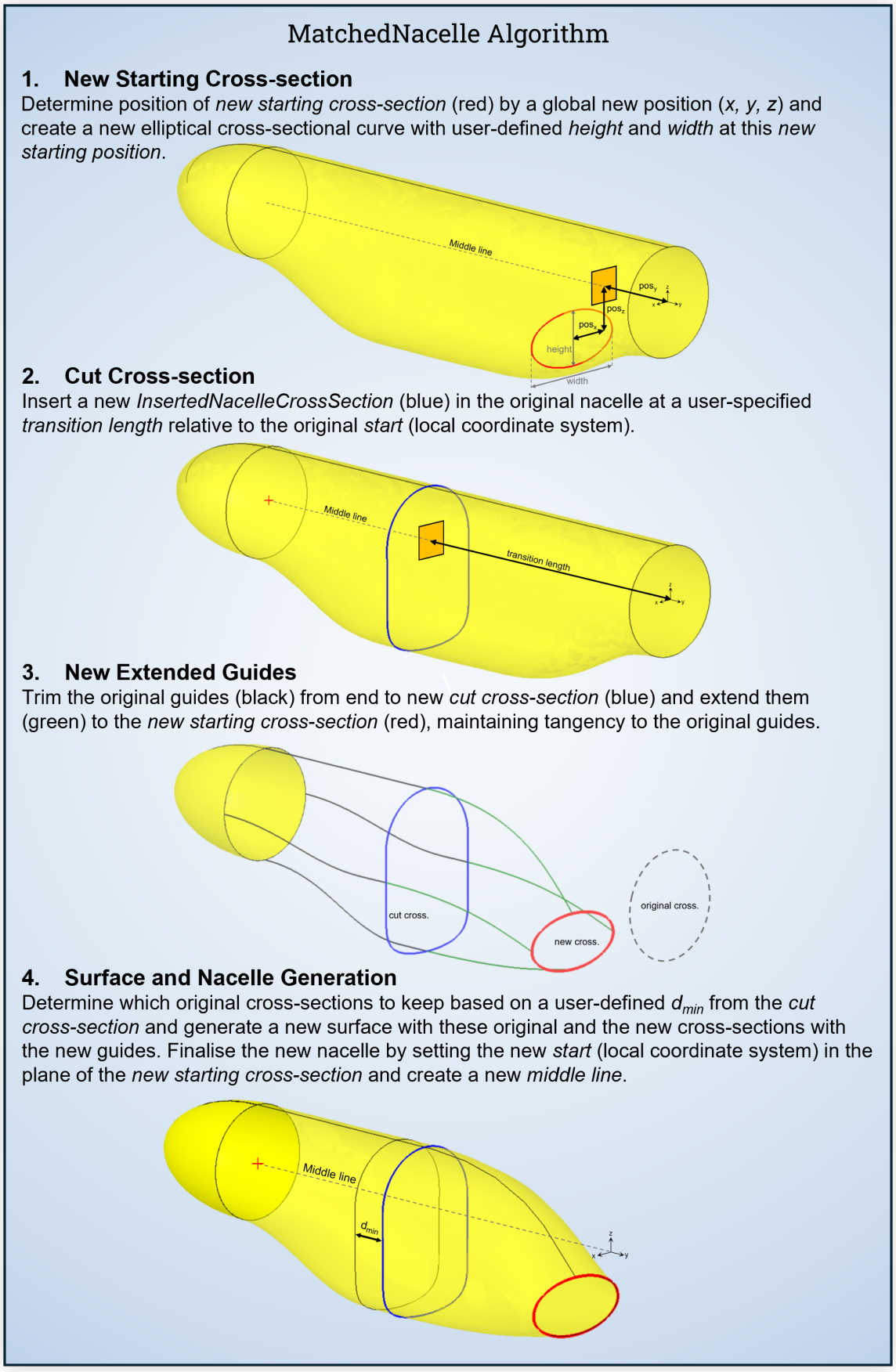


Figure 6.10: Illustrative workflow for constructing a `MatchedNacelle` by defining a new nacelle start location and transition cut.

6.6.2. Inserted Cross-section

The matched nacelle uses an inserted nacelle cross-section to define which regions of the original nacelle geometry are preserved and which are modified during geometry matching. Similar to the `InsertedAirfoil` introduced in section 5.5.2, the `InsertedNacelleCrossSection` serves both as a diagnostic reference, identifying the boundary between unchanged and redesigned geometry, and as a critical geometric entity for constructing the matched nacelle surface. In addition, this inserted cross-section is particularly important for nacelle–scoop integration, as it defines the precise location and target geometry where the scoop duct intersects the nacelle, as discussed in section 7.5.

The process starts by defining a reference position along the nacelle centreline using a user-defined relative length ratio (*Cut Cross-section* in figure 6.10). A local reference frame aligned with the nacelle coordinate system is established, with a rotation applied to match the nacelle cross-section convention[5]. The nacelle is then intersected with the local XY-plane (figure 6.10) to obtain the cutting curve used for a *direct* cross-section parametrisation.

In contrast to `InsertedAirfoil`, the `InsertedNacelleCrossSection` preserves scalable dimensionality in both width and height. This enables robust geometric matching in nacelle-scoop configurations and justifies its direct inheritance from `NacelleCrossSection`, as shown in figure 6.1. An example of a uniformly scaled inserted nacelle cross-section, positioned at 50% nacelle length in the *baseline* design, is shown in figure 6.11.

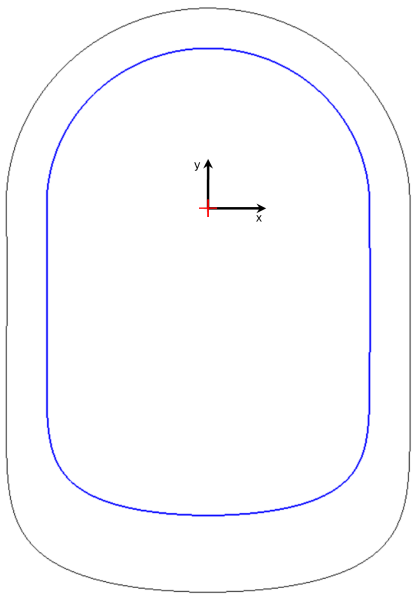


Figure 6.11: Inserted nacelle cross-section (blue) at 50% nacelle length in the baseline design, uniformly scaled by a factor of 0.8 and compared to the original nacelle contour (black).

6.6.3. Limitations and Recommendations

In summary, the `MatchedNacelle` provides a practical and robust mechanism for modifying the forward portion of an existing nacelle. It enables intuitive parametric control for modifying nacelle geometry specifically designed for wing–nacelle matching and preserves the original nacelle geometry exactly beyond the transition region. Nevertheless, several important limitations remain:

- **Limited control over guide extensions** restricts the geometric design freedom of the modified nacelle section. At present, the extension of guides is automatic and cannot be parametrically tailored.

[5] For zero cant and dihedral, the cross-section axes align with the nacelle x- and z-axes (figure 6.7).

- **Internal subsystems are not considered during shape modification**, which may result in post-modification violations of internal volume constraints. Any infeasibilities, such as internals protruding outside the updated nacelle, are only detected afterwards by the `InternalContainer`.
- **A fixed starting cross-section shape** (elliptical) narrows applicability to scenarios focused purely on wing–nacelle matching. Other applications may require that the blended region retains the original cross-sectional topology rather than adopting a generic elliptical form.

Given these limitations, future extensions of the `MatchedNacelle` should target broader integration contexts beyond wing–nacelle sizing, motivating added capabilities such as guide control, shape-preserving transitions, and subsystem-aware constraints. This would enable applications including automated internal layout optimisation and higher-fidelity nacelle–wing aerodynamic studies.

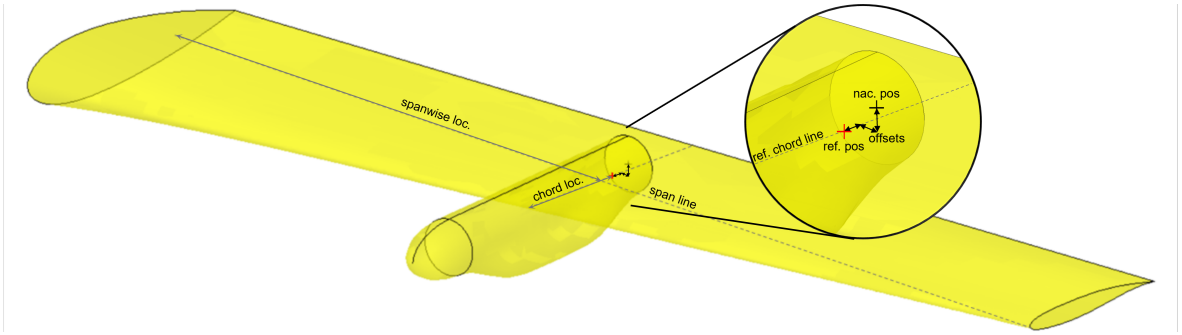
6.7. Wing-Nacelle Integration

Wing–nacelle integration is achieved by first geometrically matching the wing and nacelle to create a smooth overlap (section 4.1), followed by blending using either conventional filleting or the `BlendSolid` method. This process relies on the `MatchedLiftingSurface` (section 5.5) and `MatchedNacelle` (section 6.6), both developed specifically for wing–nacelle geometry matching.

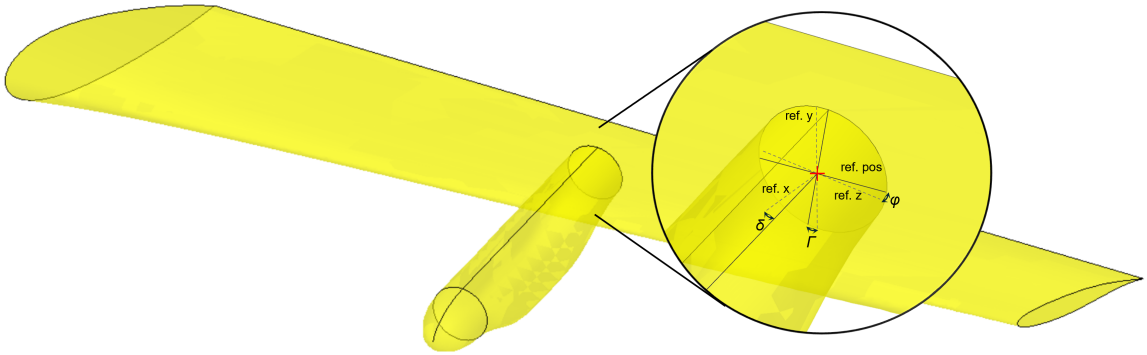
6.7.1. Nacelle Positioning

The nacelle positioning procedure is defined through a sequence of translation and orientation steps:

- **Translation:** The nacelle is positioned at the desired attachment location by prescribing its spanwise and chordwise position relative to a wing reference location (section 5.2.5). Optional local offsets in the x, y, and z directions may be applied to fine-tune the placement (figure 6.12a).
- **Orientation:** The nacelle is oriented about this reference position using a droop angle δ_{n}, a cant angle φ_{n}, and an axial rotation Γ_{n}. Together, these rotations provide intuitive and fully parametric control over the nacelle orientation relative to the wing (figure 6.12b).



(a) Initial nacelle placement: the nacelle start point is positioned using a spanwise and chordwise location relative to the wing reference position (red), followed by local x, y, and z offsets for fine adjustment.



(b) Orientation of the nacelle relative to the wing reference frame (red) using three angles: droop δ_n (about x), rotation Γ_n (about y), and cant φ_n (about z). The reference axes differ from the geometric span and chord directions, as noted in section 5.2.5.

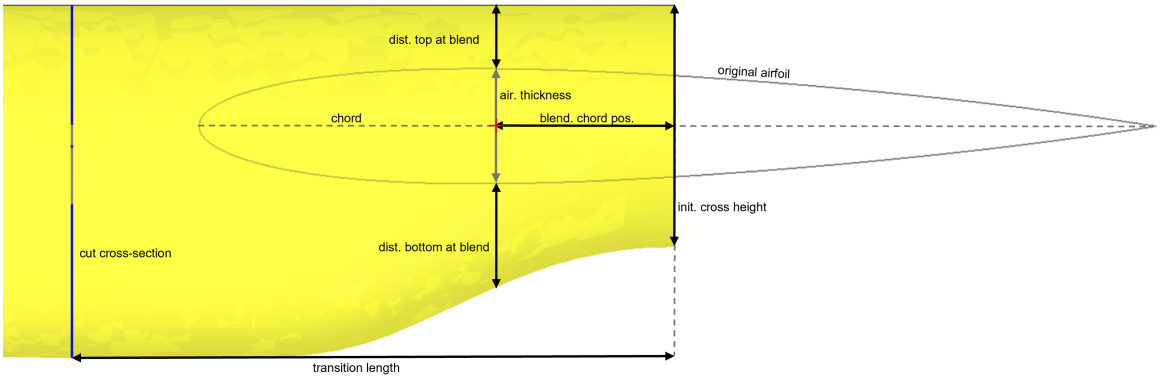
Figure 6.12: Nacelle positioning on a straight wing for the *baseline* design: placement via reference location and offsets, followed by 3D orientation control.

6.7.2. Wing-Nacelle Geometry Matching

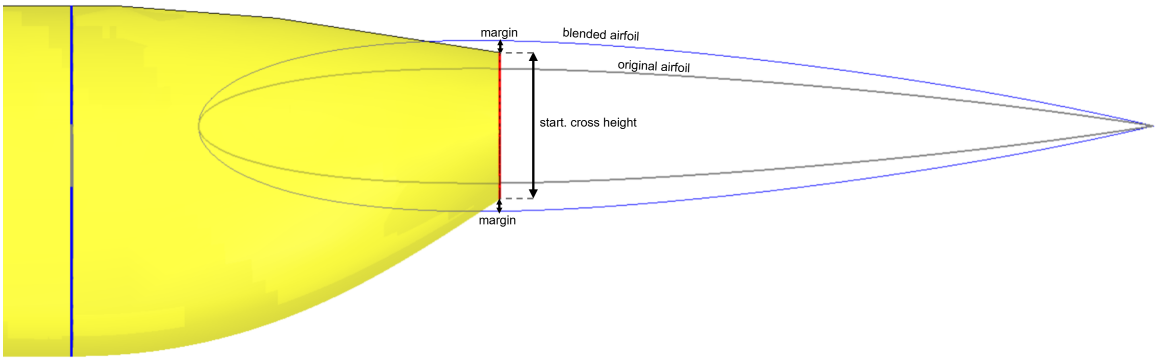
Wing–nacelle matching is performed by the `WingNacelleMatcher`, which operates on a positioned nacelle and a `LiftingSurface`. It coordinates the `MatchedLiftingSurface` and `MatchedNacelle` algorithms to reshape both components for compatible blending. Based on user-defined settings, the matcher computes the required wing *deformation factor* and nacelle start cross-section size and position, ensuring the matched nacelle lies fully within the matched wing surface, a prerequisite for a successful blend.

The matching logic is visualised in figure 6.13. First, figure 6.13a shows the original configuration, including key geometric distances between the nacelle and wing. The user specifies the desired chordwise blend position, after which the matcher evaluates the local relative geometry. Additionally, a *scaling distribution* parameter defines how the matching is achieved: where a value of 0.0 means the nacelle retains its original size, and the wing scales fully toward it and a value of 1.0 means the wing is unchanged, and the nacelle is scaled to fit inside. Intermediate values smoothly interpolate between both extremes.

Additionally, the user controls how much of each component's geometry is modified or left unchanged: the `MatchedLiftingSurface` transition *width* determines the spanwise portion of the wing affected and the `MatchedNacelle` *transition length* defines how far the nacelle is reshaped toward the new start section. These two parameters operate independently of the scaling distribution, enabling fine control over geometrical impact. Since figure 6.13 shows only a side view, the wing blending width is not visible.



(a) Initial wing–nacelle configuration showing the geometric distances used to compute the required wing scaling and new nacelle start cross-section height at the user-selected blend location (red).



(b) Matched wing (thin blue) and matched nacelle after applying the scale distribution. The new nacelle start cross-section (red) and cut cross-section (blue) are highlighted, together with the enforced robustness margin.

Figure 6.13: Wing–nacelle geometry matching in the *baseline* configuration, performed by scaling of the `MatchedLiftingSurface` and `MatchedNacelle` based on the geometry at the blend location and specified blending parameters.

6.7.3. Wing-Nacelle Geometrical Integration

Once the nacelle and wing have been successfully matched to provide a single, clean intersection curve, their geometries become suitable for integration through blending. This matching step is essential, as it ensures the existence of a well-defined shared edge between the two components, which is a strict requirement for any subsequent blending operation.

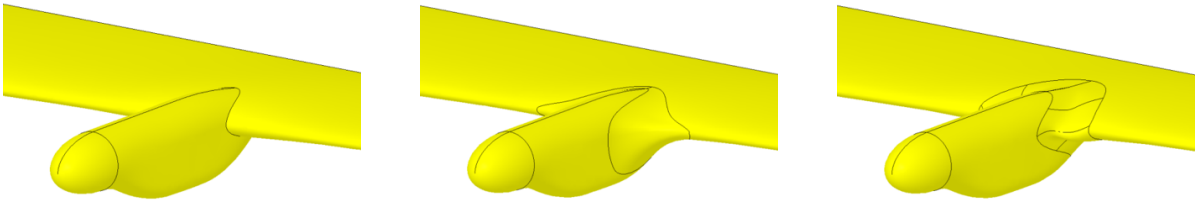
Three integration strategies are implemented in the tool:

- *Fusing*: keeps the intersection line as a sharp, G^0-continuous connection.
- *Filleting*: applies a rolling-ball fillet of constant radius to enforce full G^1 tangency along the entire integration edge.
- *Blending*: uses the novel `BlendSolid` algorithm to provide parametric control over trimming distances and blend sharpness, aiming for quasi-tangency.[6]

The geometric effect of each strategy is illustrated in figure 6.14. The fuse method (figure 6.14a) retains a non-tangent intersection, while the fillet method (figure 6.14b) produces a stable blending region with predictable, constant curvature, at the expense of limited user control. The `BlendSolid` approach (figure 6.14c) introduces flexible trimming parameters as the user-selected sharpness σ, but as shown in section 4.5.5, it fails to maintain the original LE shape and exhibits large angular deviations, with an RMS deviation of $8.69°$.

[6] A full continuity evaluation of `BlendSolid` is presented in section 4.5.4.

In summary, only the filleting strategy currently provides acceptable aerodynamic shape maintainability and guaranteed G^1 continuity, making it the preferred approach for wing–nacelle integration in the present version of the tool.



(a) Fused interface: G^0-continuous sharp edge retained.

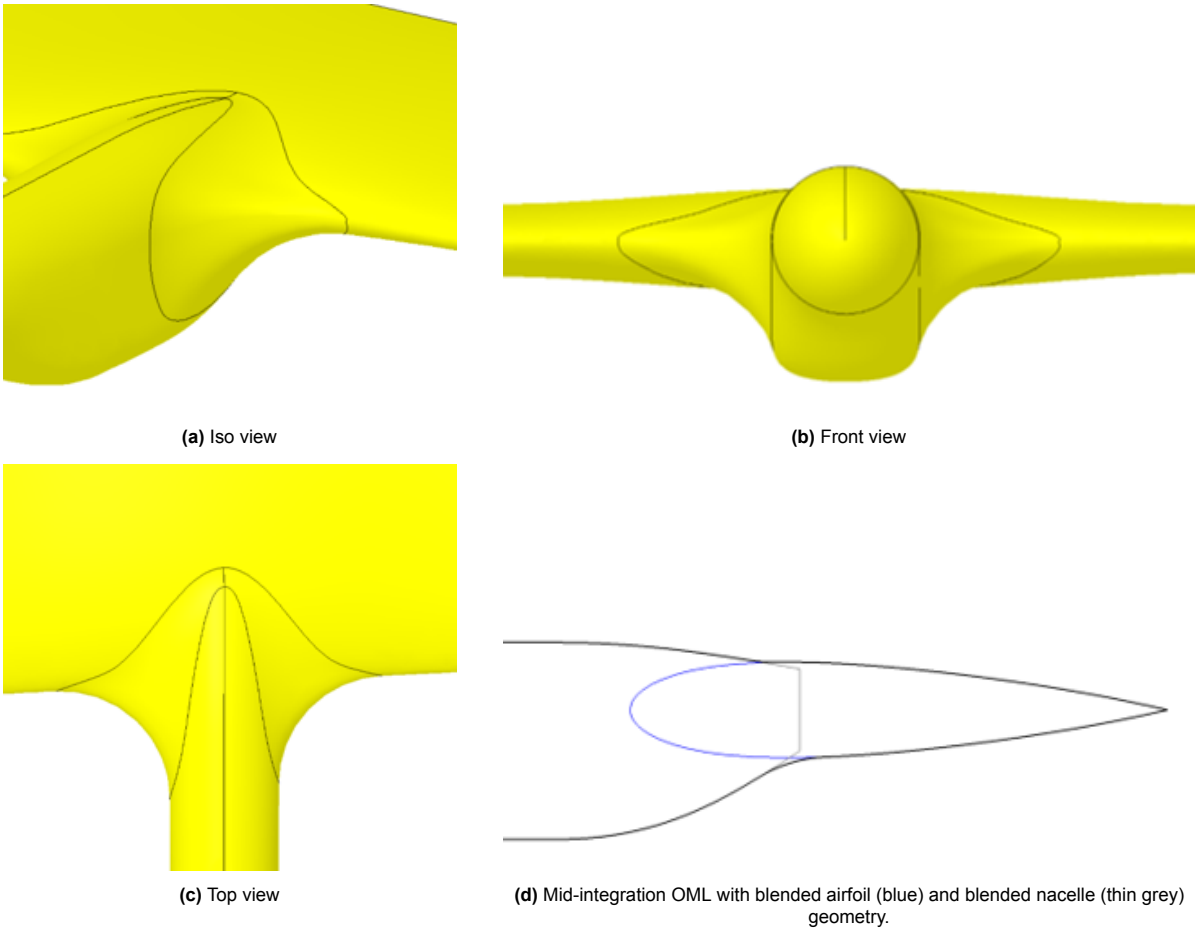
(b) Rolling-ball fillet with constant radius of 0.5 m ensuring G^1 continuity.

(c) `BlendSolid` blend using trimming distances of 0.2 m and constant $\sigma = 2.0$.

Figure 6.14: Comparison of three wing–nacelle integration strategies on the matched *baseline* configuration.

To further assess the filleted configuration, multiple views of the resulting integration are shown in figure 6.15. In the isometric view (figure 6.15a), the local shape of the leading edge is generally well preserved. However, as the radius increases, the blend reduces local airfoil shape maintenance, illustrated in figure 6.16.

The sectional view in figure 6.15d reveals that the final transition region at upper and lower surface exhibits relatively high curvature. This is attributed to the current limitations in controlling blended nacelle guides (see section 6.6.3). Designers should therefore remain aware that filleting ensures smoothness but may still influence the aerodynamic cross-section more than ideally desired.



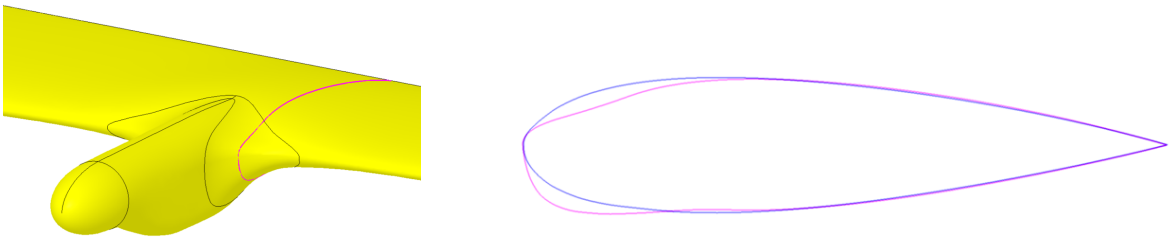
(a) Iso view

(b) Front view

(c) Top view

(d) Mid-integration OML with blended airfoil (blue) and blended nacelle (thin grey) geometry.

Figure 6.15: Wing–nacelle integration of the *baseline* configuration using a 0.5 m fillet radius, 0.5 scaling distribution, chordwise blend at maximum airfoil thickness location, nacelle transition length 0.5, and wing blend width 0.3 m.



(a) Isometric view of the local fillet-section evaluation (purple) at mid-blend location for fillet radius of 0.5 m.

(b) Side view of fillet radii at mid-blend location: 0.1 m (blue) and 0.5 m (purple).

Figure 6.16: Local airfoil-shape preservation mid-blend for fillet-based integration on the *baseline* configuration.

6.7.4. Limitations and Recommendations

In conclusion, the wing–nacelle integration approach presented in this work reliably produces blended configurations by first ensuring geometric compatibility between wing and nacelle surfaces. However, based on visual inspection and critical evaluation of the underlying methodology, several limitations are identified:

- **No explicit control over the final intersection location** as the user cannot directly specify the

exact connection line where wing and nacelle meet. Since filleting is currently the only viable blending strategy, the trimming region on each component depends solely on the rolling-ball radius and local curvature. Moreover, the robustness margin introduced during matching shifts the actual intersection relative to the intended chordwise position.

- **Assumption of nacelle fully ahead of the wing** in the current workflow presumes the nacelle volume remains entirely upstream of the wing. This restricts applicability to relatively compact FC-representative propulsion systems as for larger, aft-extending nacelles (e.g. section 8.1.2 under Havilland dash 8 Design) structural feasibility and matchability become limited.
- **Single chordwise connection point for upper and lower surfaces** in the current `MatchedNacelle` formulation, both upper and lower surfaces are tied to the same nacelle start-section position. Designs that would benefit from independent upper and or lower alignment cannot be represented.
- **Uniform wing deformation factor during matching** by the `MatchedLiftingSurface`, as discussed in section 5.5.3, results in scaling the entire cross-section uniformly. This prevents accurate local shape matching to the nacelle geometry and does not consider the actual contact curve.
- **Limited shape control in the blending region** since fillet currently offers only a single design parameter (radius). As demonstrated in figure 6.16, airfoil shape near in the integration is highly sensitive to this radius, and designers are not ensured of preservation of critical aerodynamic geometry in the blending zone.

These limitations stem primarily from the restricted controllability of fillet-based blending and the current uniform-scaling strategy in `MatchedLiftingSurface`. A future replacement or upgrade of the fillet by an enhanced `BlendSolid` algorithm would enable fully parametric control over the intermediate blending surface. However, this requires addressing its present shortcomings: *independence of blending guides compromising shape fidelity* and *quasi-tangent continuity only* (see section 4.5.5). Additionally, an improved wing-morphing capability using a modified FFD-based approach, as recommended in section 5.5.3, would allow local chordwise and thickness adaptation, enabling true geometric matching and improved aerodynamic integrity across the full wing–nacelle interface.

6.8. Capability Module Implementations

The nacelle and spinner form critical elements of the WNI configuration and significantly influence the aerodynamic characteristics of the complete system. Consequently, the `Nacelle` and `MatchedNacelle` classes both integrate the aerodynamic analysis module introduced in section 4.3.4. Each can therefore be evaluated individually within the tool. Additionally, since the `SystemAssembly` allows configuration toggles, such as disabling the intake, the aerodynamic impact of the wing–nacelle integration can be assessed in isolation as well.

As with all components presented in chapter 5, the nacelle-related components in this chapter are full geometric entities and thus directly exportable to STEP. This capability is provided through the implementation of the `GeometryExportBase` interface.

7

Scoop Intake

The final component required to complete the wing–nacelle–intake (WNI) assembly is the intake system. While several intake configurations are relevant for FC-powered aircraft (section 2.2.1), this thesis focuses on the full parametrisation of the scoop intake due to time constraints. This chapter presents the development of the scoop intake model, starting with an architectural overview in section 7.1, followed by a reusable lip component in section 7.2 and its application in two-dimensional scoop profiles in section 7.3. The three-dimensional scoop geometry and its integration with the nacelle are then described in section 7.4 and section 7.5, concluding with the associated capability modules in section 7.6.

7.1. Architectural Overview

To the authors knowledge, no existing work presents a fully parametric scoop intake model for propeller-driven or FC-powered aircraft (chapter 2). However, aerodynamic theory and experiments provide clear guidance on key intake parameters, and the conceptual similarity between intake types supports a reusable modelling framework. Consequently, only two classes are scoop-specific (`ScoopIntake` and `ScoopProfile`), while the remaining architecture (figure 7.1) is designed for general intake applicability.

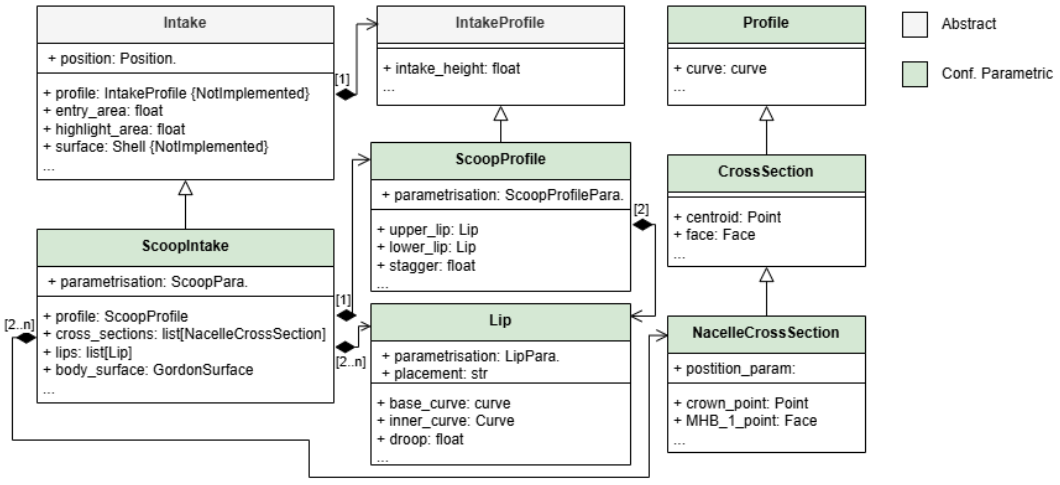


Figure 7.1: UML representation of all abstract (gray), configurable parametric (green) and specialised (blue) scoop-related classes, illustrating the structural and functional relationships.

The architecture prioritises reusability, since this package is expected to be extended first with additional intake concepts identified in section 2.2.1. At the top of the hierarchy, the abstract `Intake` class defines configuration-independent aerodynamic properties, such as `entry_area`, `exit_area`, and `wetted_fore_area`, previously highlighted as key intake performance metrics. Likewise, all intake types

can be reduced to a 2D design representation (chapter B), motivating the abstract `IntakeProfile` class, which encapsulates essential 2D geometric–aerodynamic quantities including `intake_height` and `contraction_ratio`.

The scoop intake is defined by two parametric objects: the 2D `ScoopProfile` and the 3D `ScoopIntake`, both following the *parametrisation as composition* philosophy, enabling introduction of alternative design methodologies without modifying the core geometrical and integration (section 7.5) logic. The 3D intake is generated from a single `ScoopProfile` and constructed using lips and `NacelleCrossSection` objects, allowing future extensions to more complex 3D shaping. Each scoop profile consists of an upper and a lower lip, consistent with the representation in figure 2.2.

Finally, a reusable `Lip` class is included, as most intake concepts feature a lip-like inflow element with similar geometric requirements, including turbofan nacelle cowls. The `Lip` extends the `Profile` base class and derives aerodynamic parameters from curves generated by a `LipParametrisation`. To support reuse and future expansion, all non-abstract intake components are implemented as *configurable parametric* classes.

7.2. Lips

The lip forms the forward intake contour, guiding both external flow around the airframe and internal flow into the intake. Because it is the first surface exposed to the freestream, the lip is highly influential on the overall aerodynamics of the intake as it functions a critical role in for example boundary-layer development, separation phenomena at high incidences, and therefore plays a critical role in both inlet performance[1] and overall aerodynamics.

Geometrically, a lip is represented as an open 2D curve and is therefore implemented as a specialised `Profile` (section 5.2). To ensure consistent extraction of aerodynamic characteristics, such as highlight point, leading-edge radius, and droop angle, the following requirements are imposed:

- The curve start and end lie exactly on the local y-axis.
- The midpoint between start and end coincides with the origin for consistent positioning.
- The start point has a higher y-coordinate than the end point (with clockwise requirement from `Profile`, makes sure that any unpositioned lip is facing leftwards).

With these conventions, lip-related geometric and aerodynamic quantities can be computed robustly (figure 7.2). The highlight point is identified either by maximum curvature, maximum streamwise y-position, or user override, from which properties such as camber line, droop angle, and leading-edge radius are derived, along with aft thickness and internal wall angle from the curve geometry. By convention, the lip's upper surface defines the internal intake wall. When used as an upper lip in a profile, it is flipped to preserve consistent internal wall angle and droop sign conventions.

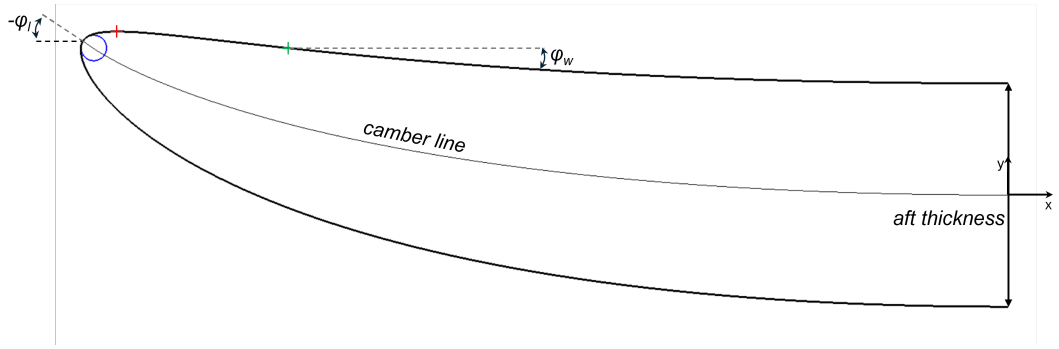


Figure 7.2: Example B-spline used as a *direct* lip curve, with automatically computed highlight (tip), leading-edge radius (blue), throat point (red), maximum internal wall angle location (green), droop angle and camber line.

[1] Full description of intake aerodynamics and flow phenomena can be found in [24].

7.2.1. Clean Lip and Lip Curve Parametrisations

A lip may be defined using any curve that satisfies the geometric constraints above. The most general option is the *direct* curve parametrisation, which accepts an arbitrary *curve* (as the B-spline in figure 7.2). However, this provides limited intuitive control based on aerodynamic design metrics from section 2.2.

To address this, a dedicated *clean lip* parametrisation is introduced. It separates lip geometry into: a curved *centerline* defined by a centerline parametrisation and two non-dimensional *inner* and *outer* surface curves that are wrapped around this centerline.

The external lip curves start at the local origin and rise smoothly upward, similar to a square-root-type function, before being displaced and bent according to the centerline. This construction is analogous to the rail-deformed profiles used in the *clean wing* parametrisation (section 5.3.3), and is therefore named accordingly.

Euler Centerline

Since the centerline determines lip curvature and droop, the abstract `CenterLineParametrisation` defines at minimum the lip *length* and *droop angle*. The current concrete implementation uses an Euler (clothoid) spiral, where curvature varies linearly with arc length, enforcing G^2 continuity:

$$\kappa(s) = \kappa_0 + a \cdot s, \tag{7.1}$$

with κ_0 the starting curvature and a the curvature rate. Setting $a = 0$ produces a circular arc, enabling intuitive control over global curvature.

A transformation maps the user inputs *length*, *droop*, and newly introduced non-dimensional[2] *shape parameter* σ_{euler} into κ_0 and a. Example variations are shown in figure 7.3 where it must be noted that the designer has full control on the length, droop and overall curvature strength of the curve, however the curve vertical height cannot be controlled.

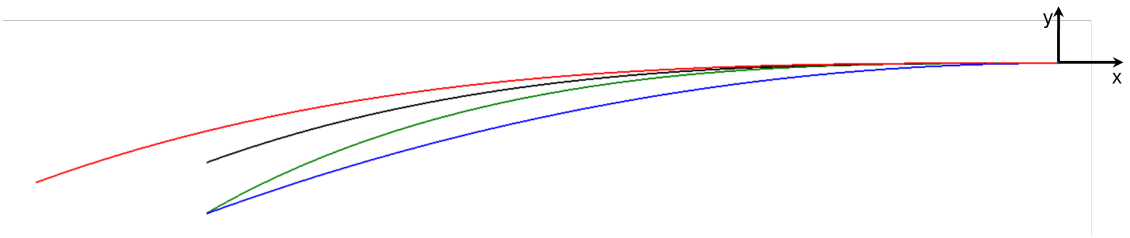


Figure 7.3: Euler centerlines compared to a baseline design (black; $l_{\text{lip}} = 1$ m, $\varphi_\ell = 20°$, $\sigma_{\text{euler}} = 0.0$), with variations in *length* (red; 1.2 m), *shape parameter* (blue; $\sigma_{\text{euler}} = 1.0$) and *droop* (green; $\varphi_\ell = 30°$).

CST Lip Curve

The second component of the clean lip parametrisation controls the inner and outer surfaces. For compatibility, each lip surface curve must:

- Start at $(0, 0)$
- Lie entirely in the positive XY-quadrant.
- End at a positive y-value to ensure finite wall thickness.
- Preferably have a vertical starting tangent for smooth LE curvature.

A primitive option is the *direct lip curve*. The preferred option is the *CST lip curve* parametrisation, based on the CST formulation in section 5.2.3. It uses a specified leading-edge radius, trailing-edge thickness, and a set of shape coefficients. Here, the class exponents are fixed to $N_1 = 0.5$ and $N_2 = 1.0$, suitable for smoothly turning airfoil-like shapes. The first CST coefficient directly controls the leading-edge radius leading to that A_0 can be computed from this radius [64]:

[2]Where $\sigma = 0$ is that a is related to the overall size of the curve and $\sigma = 1.0$ a circle.

$$A_0 = \sqrt{2 \cdot LE_{\text{radius}} / l_{\text{lip}}}. \tag{7.2}$$

Combined with the Euler centerline, this provides full control over lip length, droop, global curvature strength, LE radius, aft thickness, and detailed upper/lower surface shaping. An example is shown in figure 7.4[3]. As some parametrisations are fully dependent on each-other (e.g. the LE radius of the inner and outer lip curve has to be equal) `Descriptors` (section 4.2.1) are used to provide context to each parametrisation.

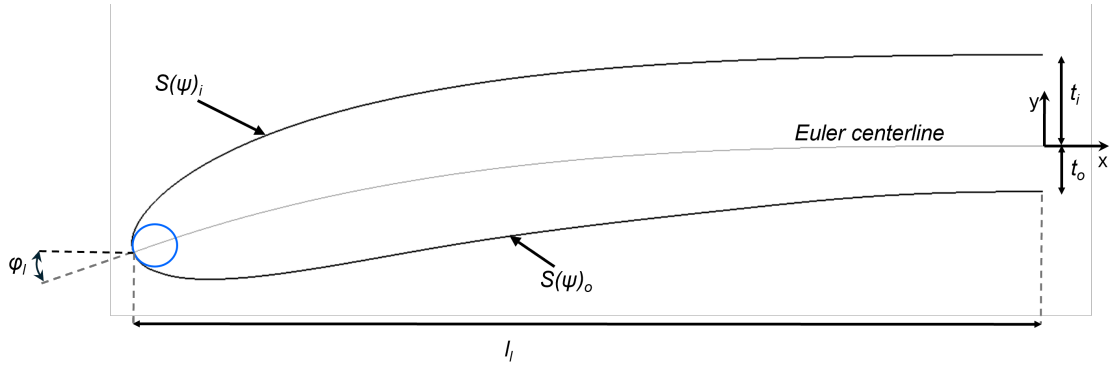


Figure 7.4: Example *clean lip* design showing centerline (gray), highlight-based LE radius (blue, 0.025 m), droop ($20°$), length (1.0 m), aft outer-wall thickness (0.05 m), aft inner-wall thickness (0.1 m), and CST-based shaping of both inner and outer surfaces.

7.3. 2D Scoop and Intakes

Since every intake configuration can be characterised by a single section, the intake package includes the abstract `IntakeProfile`, providing an interface for 2D aerodynamic properties such as *intake height* (h_c), *highlight height* (h_h) and subsequently computed *contraction ratio* (CR). As introduced in the beginning of this chapter, the only fully implemented concrete subclass is the `ScoopProfile`.

7.3.1. Scoop Profile

The `ScoopProfile` defines a scoop via two lips: an upper and a lower one, created by a chosen *scoop profile* parametrisation. From these `Lip` objects, the characteristic 2D scoop quantities from figure 2.2 are computed, including *stagger* θ_s, *intake height* h_c, *highlight height* h_h, *duct height* h_d, and *length* l_c.

Since `ScoopProfile` is a *configurable parametric* object, any lip parametrisation can be substituted. The simplest implementation is the *direct scoop profile*, where two independently parametrised `LipParametrisation` objects are positioned at a user-defined vertical offset or *duct height*, using a placement reference (inner/middle/outer walls) to ensure a consistent separation.

7.3.2. Clean Scoop Profile

A second parametrisation, the *clean scoop profile*, enforces that both lips follow the `CleanLip` formulation (section 7.2.1). This makes the scoop profile directly controlled using the aerodynamic design parameters introduced in section 2.2: stagger θ_s, highlight height h_c, upper/lower lip droop ($\varphi_{l,u}$, $\varphi_{l,l}$), upper/lower leading-edge radii ($LE_{\texttt{radius,u}}$, $LE_{\texttt{radius,l}}$), and intake length l_c. In addition, each lip retains its own *local* design freedoms (e.g. CST coefficients, aft thickness, centerline curvature), enabling asymmetric intake shaping.

Because the scoop profile determines mutual lip placement (e.g. based on h_c and θ_s), several `CleanLip` attributes, such as upper lip length and exact location, are computed at the profile level from the newly introduced *clean scoop profile* input parameters. This ensures geometric consistency, but requires that both lips use the same centerline and LE-radius-based *clean lip* construction.

[3] Note that is in this figure, the *centerline* and LE-radius from the *clean lip* are visualised, which are different from the `Lip` computed camber line and LE radius (as the highlight point is computed making it not perfectly coincide with the end of clean lip centerline).

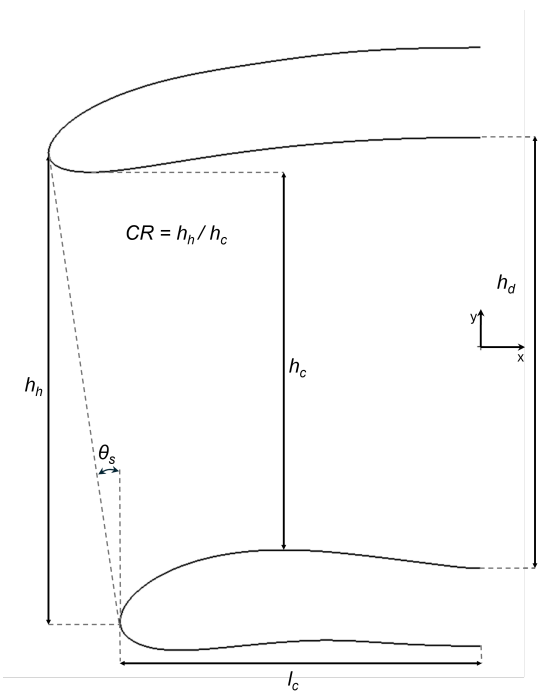


Figure 7.5: *Direct scoop profile* with $h_d = 0.5$ m and outer lip reference, constructed from two clean lips. Intake parameters (θ_s, h_c, h_h, h_d, l_c) are computed from the lip geometry.

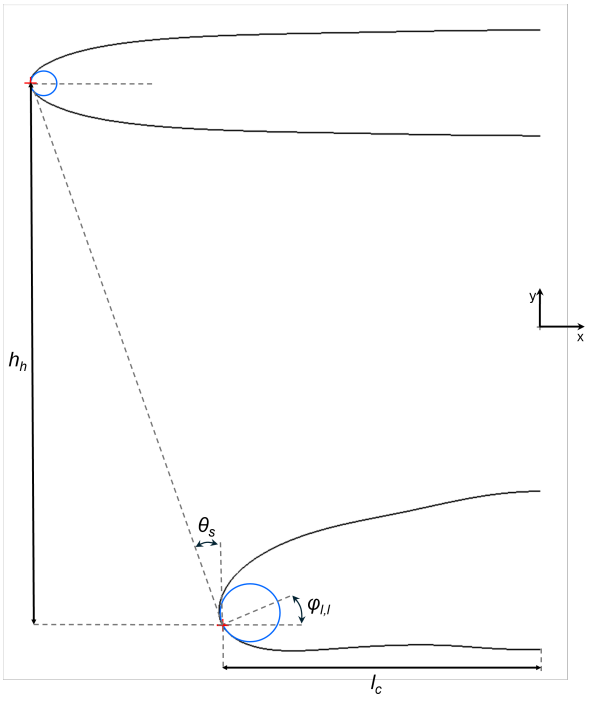


Figure 7.6: *Clean scoop profile* with $\theta_s = 20°$, $h_c = 0.5$ m, $l_c = 0.3$ m, droop ($0°$/ $30°$), and radii (0.01 m/ 0.025 m). CST shaping parameters are omitted for brevity.

7.4. 3D Scoop Intake

With the 2D scoop profile fully defined, the complete 3D intake body can now be constructed. The *configurable parametric* `ScoopIntake` generates the external surface using the adapted well-established aerodynamic-body surface construction principle by Sóbester and Forrester [62], where geometry is formed by combining *cross-sections* and *guides*. This strategy has been consistently applied throughout the full tool architecture.

In the case of the scoop, the upper and lower lips of the `ScoopProfile` with optionally extra intermediate lips act as geometric guides, defining the primary shape of the intake. The body is completed by positioning cross-sections along the these lips with at minimum, at the lip starts and ends, and optionally at intermediate locations to increase geometric fidelity.

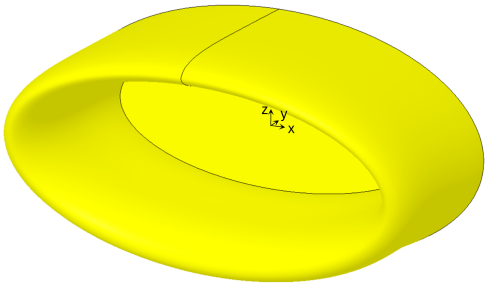
To ensure reusability, the `ScoopIntake` does not embed a specific scoop definition, but instead depends on a *scoop parametrisation*. This interface constructs: a single `ScoopProfile` at least two `NacelleCrossSection` objects and optionally extra `Lip` instances positioned out of the profile plane. The final surface is computed using a Gordon surface (section 4.4), interpolating between the lips and cross-sections as visualised in figure 7.7a.

7.4.1. Direct Scoop

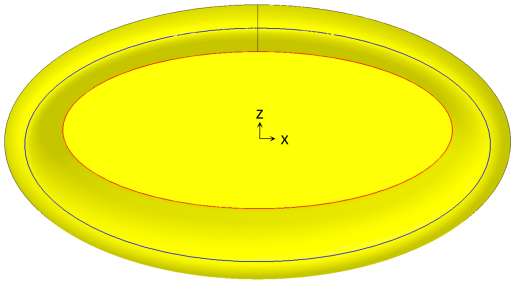
The currently single implemented *scoop parametrisation* is the *direct scoop*, which combines any `ScoopProfile` parametrisation with a single *intake-area* parametrisation that defines the frontal highlight shape (Ω_c in figure 2.2).

The intake area is defined as a non-dimensional `nacelle curve` parametrisation (section 6.4.1), scaled such that its height equals the highlight height h_h of the scoop profile. It is then placed between the highlight points of the two lips, ensuring that, from a perfect front view, the highlight ridge follows the parametric cross-section shape exactly (figure 7.7b). Finally, the same cross-section shape is positioned and scaled accordingly at the inner and outer lip end points, completing the input needed for constructing the body via a Gordon surface. Important is that the cross-section at the highlight exactly

maintains the intake-area shape and the cross-section at the internal and external ends are slightly reshaped to fit the upper and lower lips. The resulting geometry for the example scoop is shown in figure 7.7.



(a) Isometric view of the generated scoop body.

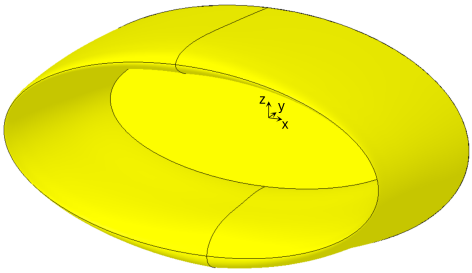


(b) Front view showing sized cross-sections at highlight (blue) and lip-end locations (red), showing a relative shape difference due misalignment of the highlight plane due to stagger.

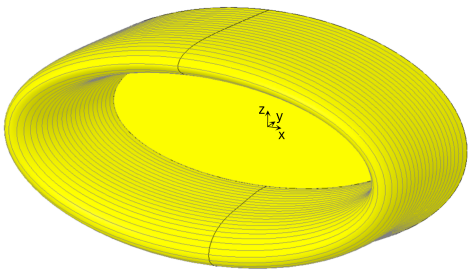
Figure 7.7: *Direct scoop* parametrisation built from the profile in figure 7.6 and a CST intake-area curve with class exponents $N_1 = N_2 = 0.5$ and shape coefficients $[0.5, -0.5]$. Coordinate system represents the intake local reference frame and thus position.

However, creating strongly curved bodies (in this case driven by the small lip LE radii) using a Gordon surface introduces noticeable interpolation error in regions in-between cross-sectional and guide constraints. Since only the boundary curves (profiles and guides) are exactly enforced, intermediate shape is an merely an interpolation of the curves, which does not preserve the intended LE shape. This can be observed in figure 7.8a, where the intermediate highlight LE shape is poorly captured and does not resemble any of the lips from the scoop profile. The same Gordon surface inaccuracy appears in the nacelle geometry validation where it is exactly quantified in section 8.2.2. A dedicated scoop validation study would quantify the effect in this case and relation to required amount of intermediate curves and is therefore strongly recommended for future work.

To mitigate this interpolation inaccuracy, the *direct scoop* parametrisation introduces a user-defined number of additional interpolated intermediate cross-sections (i.e. frontal area shapes), which are supplied as extra Gordon surface profiles[4]. Increasing this number significantly improves shape fidelity, as shown in figure 7.8b. As the direct scoop is defined by a single frontal area, intermediate cross-sections are manually interpolated, rather than lip profiles, as the latter would require complex interpolation of the upper and lower lips for each intermediate profile.



(a) Scoop geometry using no intermediate cross-sections.



(b) Scoop geometry using 40 extra interpolated intermediate cross-sections.

Figure 7.8: Effect of inserting intermediate cross-sections on LE fidelity for the scoop in figure 7.7.

[4]Within the context of the Gordon surface formulation (section 4.4), the term *profiles* refers to the set of curves defining the surface interpolation, and should not be confused with the profile objects used elsewhere in the tool (e.g. `Profile` and `ScoopProfile`).

7.5. Nacelle-Scoop Integration

As introduced in section 4.1, the nacelle–scoop integration is handled by the `ScoopNacelleIntegration` object, which blends the geometry of any `NacelleBase` instance (either a *configurable parametric* nacelle or a *specialised* matched nacelle from section 6.1) with the scoop intake developed in section 7.4. Because the wing–nacelle integration (section 6.7.3) operates successfully on any matched geometry, the resulting blended nacelle–scoop configuration can be further integrated into the full wing system, completing the WNI assembly.

The geometric integration follows the same core strategy as in section 6.7: first ensure a clean geometric match, then apply a blending operation. A transitional external duct is therefore constructed between the scoop and a user-defined reattachment position on the nacelle. This ensures that both surfaces align geometrically, providing robust conditions for the subsequent blending stage.

Following the typical workflow of the `SystemAssembly`, the scoop and nacelle are initially designed independently and only then combined. Because the intake lies at the foremost region of the nacelle, it is directly exposed to the free-stream flow and demands smooth but controllable geometry. Consequently, the primary integration requirement can be stated as: The nacelle–scoop integration must produce a smooth, well-controlled geometric transition that preserves the general characteristics of both the scoop and nacelle while ensuring geometrical continuity.

7.5.1. Algorithm

An overview of the `ScoopNacelleIntegration` workflow is shown in figure 7.9. The process begins by positioning the scoop relative to the nacelle using a *scoop location* (expressed as a lengthwise ratio along the nacelle middle line) and a *diverter height*, i.e. the vertical separation between the nacelle lower surface and the upper aft-surface of the scoop. At present, the scoop cannot be located outside the nacelle YZ-plane (i.e. no rotation around the nacelle centerline is permitted) and cannot be oriented other then be aligned with the nacelle local coordinate system.

A *reattachment cross-section* is then constructed via an `InsertedNacelleCrossSection` (section 6.6.2) placed at a user-specified *reattachment location*. As this section is initially too large to serve as the scoop duct termination, it is uniformly scaled down by a *margin* to remain fully inside the nacelle surface, before being trimmed inward from both ends (and thus the crown point) using a *trim ratio*. The remaining endpoints are re-connected with a smooth curve to ensure a closed, well-conditioned duct boundary.

Next, new guides are generated via interpolation between the positioned scoop and the reattachment cross-section, after which a Gordon surface (section 4.4) is constructed to form the external duct. Due to the interpolation-related fidelity limitations of Gordon surfaces highlighted in section 7.4.1 and quantified in section 8.2.2, the user may specify an increased number of guides to suppress shape distortion in strongly curved regions. As a final note, the final duct surface is not created by stitching the scoop and duct geometry (as tangency is not guaranteed), but is created as a single Gordon surface by extending the new duct guides to lie on the original scoop surface.

Finally, a geometrical blend between the duct surface and the (optionally blended) nacelle surface is created using the parametric `BlendSolid` (section 4.5). While the underlying shape-control parameters remain as defined in section 4.5.3, their interpretation is specialised for this integration case. Instead of a single sharpness parameter σ, two independent sharpness values are defined: one at the diverter location and one at the nacelle–scoop *shoulders*, i.e. where the duct intersects the nacelle laterally[5]. These values are connected using a σ-law to obtain a smoothly varying sharpness transition. This preserves overall control authority while improving blending behaviour and control at the most critical geometric interaction zones.

[5] Shoulders are detected by locating the closest points on the duct–nacelle intersection curve to the nacelle's MHB points on an inserted cross-section at the scoop location.

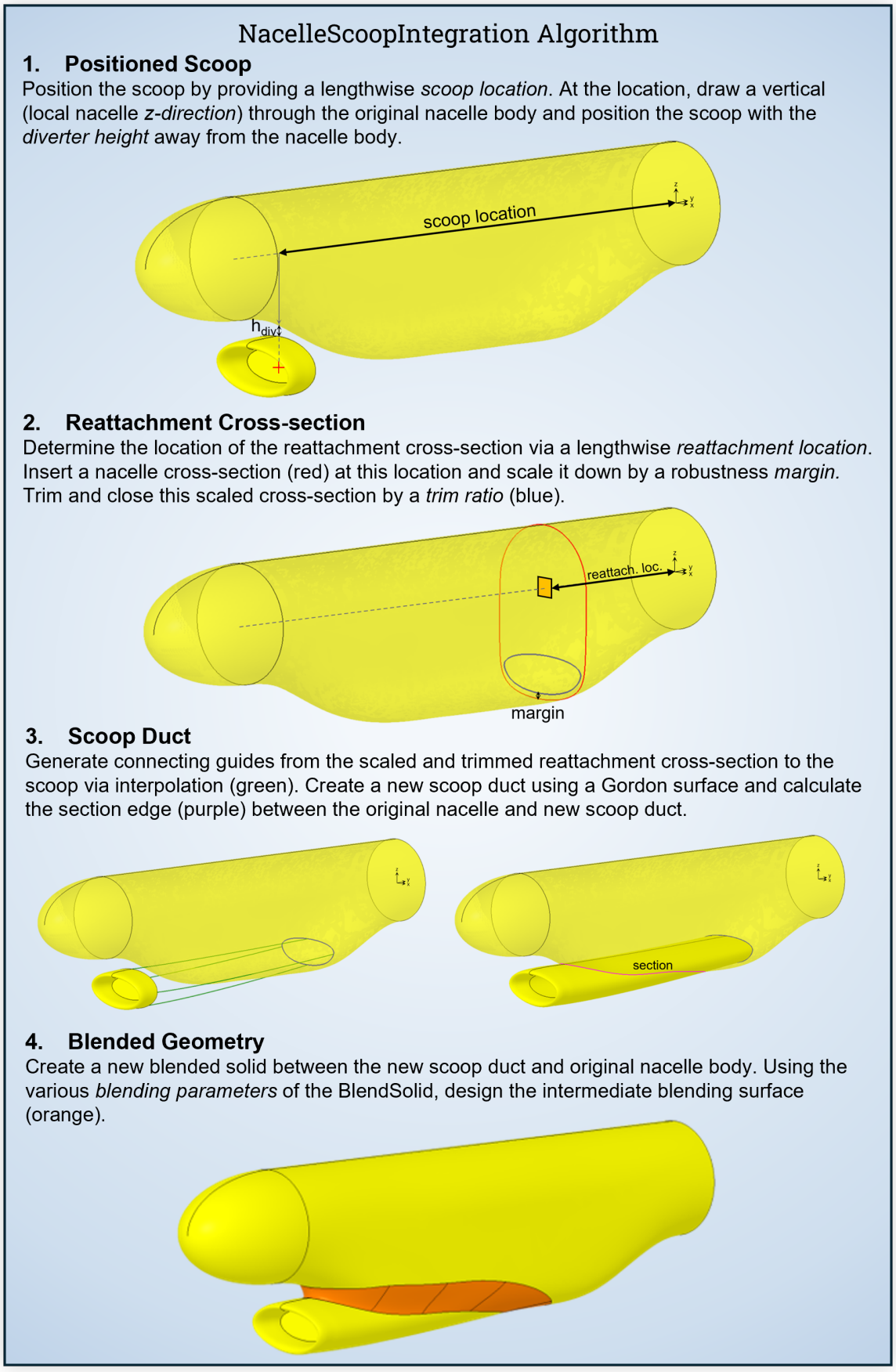


Figure 7.9: Illustrative workflow for constructing the nacelle-scoop integration by positioning of the scoop, definition of reattachment cross-section, external duct construction via Gordon surface, and final transition blending using `BlendSolid`.

7.5.2. Limitations and Recommendations

In conclusion, the nacelle–scoop integration approach developed in this work demonstrates robust geometric construction with smooth transitions and intuitive designer control. Nonetheless, several key limitations remain:

- **Lack of exact dimensionally prescribed features** as the nacelle–scoop integration cannot be sized directly by the designer. Quantities such as *scoop length* (l_c) and *diverter height* (h_{div}) are not explicit inputs but emerge from geometric relations involving *scoop location*, *reattachment location*, trimming, and blend sharpness.
- **Restricted blending shaping** since intermediate shaping of the duct–nacelle transition remains limited. Current controls do not support variational trimming along the edge, differential σ distribution at both sides of the edge, or exact intermediate location interpolating of blended surface (section 4.5.3).
- **Scoped range of nacelle geometries** as the current `BlendSolid` implementation operates only on a single closed intersection curve between exactly two faces on both solids. As a result, applicability is constrained primarily to nacelles with relatively planar or gently curved lower surfaces.
- **Constrained scoop positioning** as only axial positioning and vertical offset can be controlled. No rotation of the scoop around the nacelle centerline and differential scoop orientation is supported, reducing flexibility in scoop positioning.
- **No diverter-less designs** can be generated, as the nacelle-scoop integration enforces non-overlapping nacelle and scoop geometries, inherently resulting in the presence of a diverter. In addition, divert-less designs may require direct coupling of scoop and nacelle geometry.
- **Scoop quantification is fixed to one**, implying that dual-scoop configurations are not supported. This limitation follows directly from the *constrained scoop positioning* as the scoop position is now fixed to the lower surface of the nacelle and *restricted geometric applicability* of the `BlendSolid` (section 4.5.5), as twin-scoop layouts would likely lead to overlapping blend regions.
- **No cross-component parameter coupling** as the scoop and nacelle are designed entirely independently. No higher-level parameters exist to jointly influence both geometries (e.g., the *peripheral extent* concept from section 2.2 under Scoop Intake).
- **Only quasi-tangent surface continuity** is supported as the blend achieves smoothness but not strict geometric G^1 continuity. As measured in section 4.5.4, RMS angular deviations of approximately $1.0°$ remain between the blended surfaces of this complexity.
- **Dependency on intermediate duct guides** is needed since the Gordon surface interpolation inaccuracy for strongly curved scoop geometries requires additional guide curves, which may degrade robustness and convergence (as similar intermediate b blending guides do in section 4.5.4).
- **(Overlap risk with wing–nacelle blend)** since the nacelle–scoop transition geometry can interfere with the wing–nacelle blending zone, restricting feasible combinations of design and integration parameters. This limitation originates from the overall system architecture (section 8.1.2) rather than the nacelle–scoop integration alone as it emerged from the manual verification step in section 8.1.2.

Recommendations

Following these limitations, it can be concluded that a significant portion stem from constraints in the underlying geometric algorithms. Therefore, the first set of recommendations focuses on improving these foundations:

- **Improved Gordon surface algorithm** would mitigate several observed shortcomings, particularly regarding quasi-tangency, robustness, and the accuracy of intermediate surface interpolation.
- **Extending the `BlendSolid` algorithm** would allow the designer greater influence over the blending region and yield more dimensionally controlled results, enabling more intuitive parameters such as l_{div} and l_c and enhanced shape authority. In addition, a strengthened `BlendSolid` would directly benefit from an improved Gordon surface implementation through better tangency and robustness.

Secondly, the remaining limitations arise from the current nacelle-scoop integration modelling methodology, indicating a need for richer parametrisation:

- **Extension of design parameters** with more intuitive and explicitly defined inputs would significantly enhance both geometric fidelity and accessible design space. Improved nacelle placement control with more degrees of freedom would increase feasible configurations, while more geometrically meaningful input parameters would strengthen usability and geometric accuracy. Moreover, as `ScoopNacelleIntegration` is currently a rigid *specialised* class, refactoring it into a *configurable parametric* one would generalise the parameter space and better support extendability, requiring a more abstracted integration concept.
- **Extended parametrisations** for the nacelle, the scoop, and potentially the integration itself would support more coupled design decisions, reflecting inherent aerodynamic dependencies between components as concluded from the literature in chapter 2. Such coupling can be achieved by introducing new parametrisations that build upon or replace those in this chapter and chapter 6, and by leveraging `Descriptor` objects within the *parametrisation-as-composition* paradigm (section 4.2) to permit this complex coupling. Finally, once `ScoopNacelleIntegration` becomes a *configurable parametric* class, new parametrisations tailored to diverse integration strategies can be introduced, broadening applicability to more use-cases.

7.6. Capability Module Implementations

The full 3D scoop intake is an essential component of the WNI configuration and therefore has a considerable aerodynamic influence on the system as a whole. For this reason, the `ScoopIntake` implements the aerodynamic analysis module from section 4.3.4, enabling standalone aerodynamic simulation and assessment. Moreover, aerodynamic interactions between scoop, nacelle, and wing can be evaluated using the `SystemAssembly`, which integrates the aerodynamic module at system level. Each component can be selectively enabled or disabled, allowing the specific coupling effects (e.g. scoop–nacelle, nacelle–wing, scoop-nacelle-wing, etc.) to be analysed independently.

However, in its current implementation, the FS interface does not support the generation of flow inlet and outlet boundary conditions[6]. Consequently, full WNI aerodynamic simulations cannot yet be considered physically representative.

Consistent with the components presented in chapter 5 and chapter 6, all scoop-related components contribute geometrically to the outer mould line and are therefore exportable to STEP via the `GeometryExportBase` interface.

[6]Additionally, exhaust systems are not implemented in this work, meaning that full WNI aerodynamic analysis is beyond the present scope. As noted in section 2.2.1, modelling flow outlets should be prioritised in future work to enable complete WNI flow analysis.

8

Verification and Validation

The final chapter evaluates the tool through a comprehensive verification and validation study as these studies are crucial in determining whether the tool successfully fulfils its general objective (section 1.4.2). The verification phase (section 8.1) provides an internal assessment whether the tool is implemented in the right way, specifically in terms of its ability to generate feasible geometry across the design space, establishing performance metrics for its parametric behaviour. This analysis identifies both the tool's current applicability and its limitations, revealing dependencies on parameter ranges and highlighting model components that are robust versus those requiring further development.

The validation study (section 8.2) provides an external evaluation whether the developed tool meets the required level of capability and fidelity, thereby evaluating the physical realism of the generated WNI geometries. Successful validation demonstrates that the tool produces representative configurations for FC-powered aircraft, while initial aerodynamic validation indicates progress toward SR-5 - Aerodynamic Assessment. Together, these results show how close the tool is to a complete design-and-analysis framework for realistic WNI integrations, addressing the second problem statement in section 1.4.

8.1. Tool Verification

Before external validation, the model must first be verified to ensure that it produces computationally valid and downstream-usable geometry. Verification confirms that the tool performs as intended and that its outputs satisfy quality and compatibility criteria. The assessment procedure for verifying geometric validity is presented in section 8.1.1.

Two complementary strategies are used for verification:

1. Manual design of reference configurations, ensuring that the tool can generate physically meaningful WNI geometries at representative and uniquely distinct points within the design space.
2. Design-space coverage assessment to evaluate robustness and ensure that the tool can automatically produce valid geometries throughout the bounded design space, assessing the tools usefulness and applicability in automatic design generation workflows like optimisations and Multidisciplinary Design Optimisation (MDO) studies.

8.1.1. Geometry Assessment

Since this tool relies directly on the OpenCascade (OCC) geometric kernel, the successful creation of an OCC `TopoDS_Shape` without kernel error implies basic computational validity. However, computational validity alone does not guarantee suitability for downstream engineering workflows. Therefore, the geometrical validity conditions [70] in SR-1.1-D1 - Geometry Validity are explicitly checked for all significant shapes in the model workflow:

- **Completeness:** Each solid must form a watertight and fully connected boundary. This is verified using the ParaPy `is_valid` check, which applies `ShapeAnalysis_Shell` to detect both bad and

free edges.

- **Intersection-free:** Self-intersections, overlapping entities, and stray geometry must be absent. This is verified with `BRepAlgoAPI_Check`, checking for self-intersections, and by confirming that the final output is a true solid rather than a compound to ensure no loose geometry.
- **Manifold:** Every edge must be shared by exactly two faces as multi-manifold edges are explicitly checked. Single-manifoldness follows from the *Completeness* check, while normal consistency is ensured through solid modelling and positive-volume verification. The absence of stray elements are ruled out by self-intersection and connectivity checks from *Intersection-free*.
- **Simplicity:** Degenerate geometry (small edges or sliver faces) must be avoided. Sliver faces are detected by evaluating the area of each face against a tolerance $\epsilon = 10^{-5}\ m^2$, and small edges are identified through `BRepAlgoAPI_Check` used in the *Intersection-free* geometrical check ase well.
- **Accuracy:** Correct geometrical scaling and dimensions are enforced by geometrical detail level and parametric definitions presented throughout this work. Solver compatibility is demonstrated by the automatic panel-mesh generation and aerodynamic analysis (section 8.2.3).

A downstream capability metric related to SR-5-X-D6 - Solver Compatibility is successful automatic surface meshing. If Salome's curvature-driven unstructured mesher produces a surface mesh within a practical time limit, the geometry is considered solver-compatible. The emphbaseline configuration consistently meshes within 3 s, so an upper acceptance threshold of 20 s is adopted as exceeding this limit flags unnecessarily complex, low-quality geometry that is unsuitable for solver-based workflows.

Model Robustness assessment

To maximise diagnostic insight from the verification process, checks are not only applied to the final `solid` of the full WNI assembly but also to selected key intermediate steps throughout the workflow shown in figure 4.3. This allows identification of where potential breakdowns occur and enables identification of cause of failure.

For non-geometric slots (i.e. simple data types like numerical or logical outputs), evaluation is attempted directly. Any exception raised during evaluation triggers the creation of a `VerificationResult` that records: a failure code identifying the model stage at which the error occurred, the full slot path, the exception type, and the exception message. A complete list of failure codes is provided in table 8.1. All verification results are stored per design configuration to support systematic post-analysis.

For slots that represent geometric shapes, additional checks are performed:

1. **Kernel-level construction checks:** General construction validity checks from OCC during shape evaluation (i.e. simple evaluation of `TopoDS_shape`) (*Completeness*, *Manifold*)
2. **Explicit geometric quality checks** defined in section 8.1.1:
   (a) Absence of bad or free edges (*Completeness*)
   (b) No small edges (*Simplicity*)
   (c) No self-intersections (*Intersection-free*)
   (d) Positive enclosed volume (for solids) (*Manifold*)
   (e) No sliver faces with areas below tolerance $\epsilon = 10^{-5}\ m^2$ (*Simplicity*)
   (f) No multi-manifold edges (*Manifold*)
3. **Mesh convergence** within acceptance threshold of 20 s (for final solid only) (*Accuracy*)

Together, these checks ensure that every relevant intermediate and final geometry instance satisfies both computational and downstream-compatibility requirements, enabling early detection of modelling failures and improving traceability during automated design-space exploration. The traceability, error location (failure code), error type and message will give insights on why the error has occurred and whether it is due to a modelling mistake or lack of technological robustness (implementation of Gordon surface algorithm specifically). These causes are highlighted in subsequent sections and the final conclusion in section 8.1.4.

Table 8.1: Geometry verification failure codes used in the automated robustness assessment.

(1/2)

Code	Meaning
0	Success
Wing Generation	
10	High-level wing geometry
11	Leading-edge rail
12	Trailing-edge rail
13	Airfoil creation
14	Wing Gordon surface
15	Wing sewing solid
Nacelle Generation	
20	High-level nacelle geometry
21	Spinner profile curve
22	Spinner surface
23	Nacelle cross-sections creation
24	Guide curves
25	Nacelle body Gordon surface
26	Nacelle sewing solid
Scoop Intake Generation	
30	High-level intake geometry
31	Upper-lip points generation
32	Lower-lip points generation
33	Upper-lip curve interpolation
34	Lower-lip curve interpolation

(2/2)

Code	Meaning
Scoop Intake Generation (cont.)	
35	Intake cross-sections creation
36	Intake Gordon surface
37	Intake sewing solid
Wing–Nacelle Matching	
40	High-level WN matching
41	Reference position on wing
42	Blending airfoil creation
43	Nacelle scaling factor
44	Wing scaling factor
45	Cross-section cut
46	Blend guides creation
47	Nacelle blend Gordon surface
48	Blend nacelle sewing solid
49	Blend wing modified surface
Nacelle–Scoop Integration	
50	High-level scoop integration
51	Reattachment cross-section creation
52	Scoop replacement
53	Duct guides creation
54	Duct Gordon surface
55	Duct sewing solid
56	Nacelle-scoop blending solid
Wing–Nacelle Integration	
99	Final filleted or blend solid
Meshing	
100	Mesh divergence

A `verify_model()` function performs the full model robustness assessment, saving both the failure code (i.e. location in the geometrical generation flow), exception type and message. Custom exceptions are created for each previously introduced check from which the failed criteria can be inferred. Subsequently, the full assessment is run under the `verification_result` slot of the `SystemAssembly`. For Design-space coverage assessment (section 8.1.3), the result of each sample is saved to an external csv file.

8.1.2. Manual Design Feasibility

Representative WNI configurations are manually constructed to verify that the tool can generate geometrically meaningful concepts consistent with the principles outlined in chapter 2. Guided by reference aircraft and engineering judgement, the examples explore different regions of the design space. As the focus is on feasibility rather than validation, quantitative comparison is deferred to section 8.2.2, and detailed input parameters are omitted.

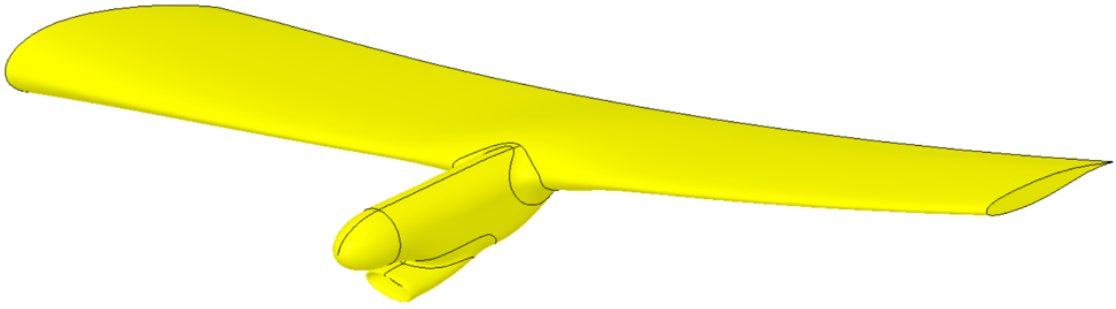
Baseline Design

The *baseline* configuration serves as the primary internal reference model used throughout this work to demonstrate all major modelling functionalities. It is not based on a specific existing aircraft but was deliberately constructed to incorporate every key parameterisation and integration feature developed in the tool. The resulting geometry is shown in figure 8.1.

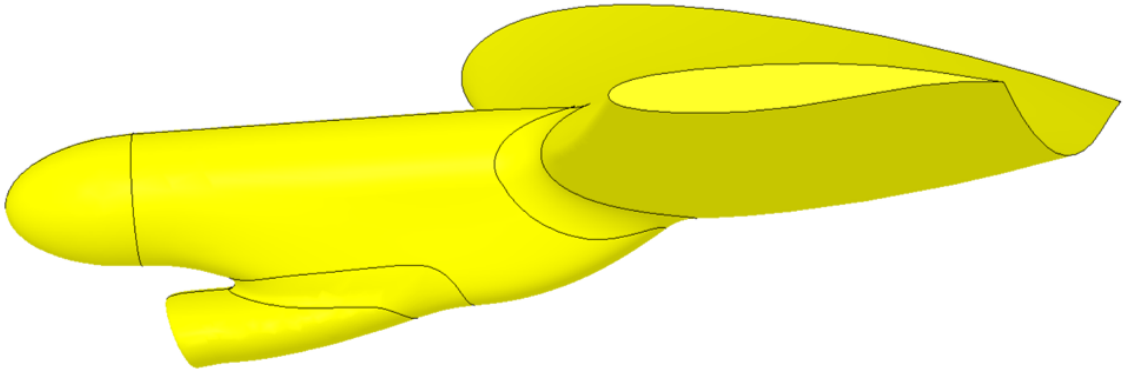
The wing is generated using the `CleanWing` parametrisation with representative twist and dihedral, as

shown in figure 8.1b. The nacelle exercises key geometric degrees of freedom, including wing resizing, large blend radii, and non-zero cant, droop, and rotation, thereby testing positioning and blending robustness. Smooth nacelle-scoop integration is achieved using the `BlendSolid` algorithm with a rounded scoop.

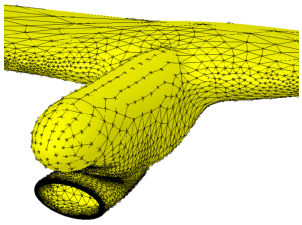
Together, these modelling choices create a realistic yet intentionally challenging design that spans the complexity of the implemented parameterisations. Successful automatic meshing, as seen in figure 8.1c, confirms the computational validity of the resulting geometry within the verification process with a full verification failure code of 0, or success.



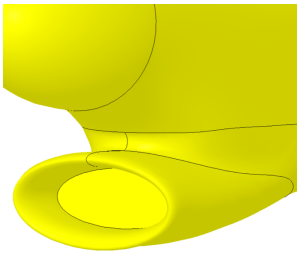
(a) Iso view



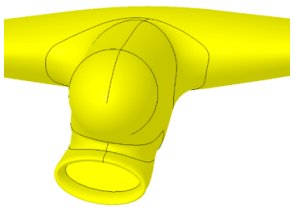
(b) Side view



(c) Nacelle focussed iso view with automatic mesh



(d) Intake focussed iso view



(e) Frontal view

Figure 8.1: Parametric model of the *baseline* design, generated using the developed tool.

Havilland dash 8 Design

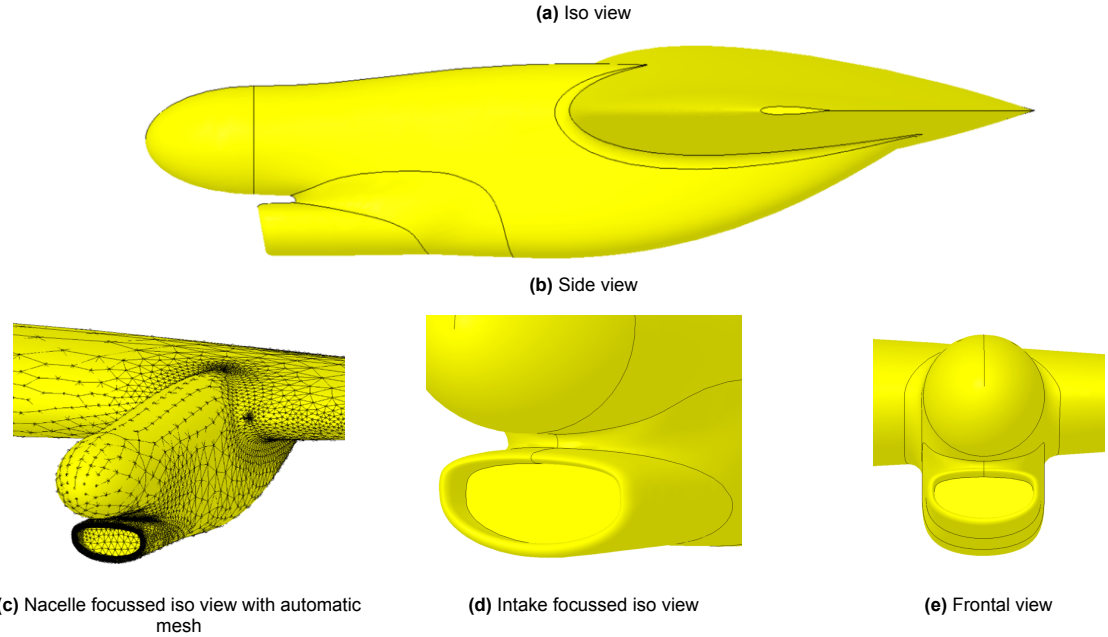
The *Havilland Dash 8* design was created to resemble the De Havilland Canada DHC-8 turboprop (figure 8.2), with the corresponding tool-generated model shown in figure 8.3. A strong visual resemblance is observed, particularly in the spinner, wing, scoop, and nacelle geometry, demonstrating that the tool can parametrically reproduce conventional turboprop WNI integrations in addition to novel FC-powered concepts. The successful surface mesh in figure 8.3c, yielding a zero geometry error code, further confirms computational geometry validity.

A limitation of the current framework is that nacelles cannot extend aft of the wing, as seen on the

real Dash 8. instead, the rear portion of the nacelle is integrated into the wing trailing region. While such integration can be aerodynamically favourable for FC-powered concepts with reduced nacelle volume requirements, it limits applicability to conventional turboprop configurations. Extending the wing-nacelle integration approach (section 6.7) to support nacelles extending beyond the wing trailing edge is therefore recommended to broaden the tool's applicability.

Another deviation occurs at the lower nacelle-scoop transition, which is straighter on the real Dash 8. Although `BlendSolid` can reproduce this, it requires precise alignment of the scoop and nacelle lower surfaces, which is difficult with the current scoop implemented parametrisations (section 7.3 and section 7.4). As scoop placement is controlled mainly by diverter height or top-surface clearance, direct control of the lower transition is limited. This indicates the need for additional parametrisations for alternative design philosophies, which can be readily integrated due to the tool's modular parametrisation framework.

Figure 8.2: Reference photograph of the De Havilland Canada DHC-8 turboprop aircraft used for the conceptual comparison.



(a) Iso view

(b) Side view

(c) Nacelle focussed iso view with automatic mesh

(d) Intake focussed iso view

(e) Frontal view

Figure 8.3: Parametric reference model resembling the De Havilland Dash 8, generated using the developed tool.

Elysian Design

To demonstrate the tool's capability to generate FC-powered, physically representative WNI configurations, the Elysian aircraft concept is recreated. Elysian is developing a fully battery-electric regional airliner designed for approximately 90 passengers with a range of 800 km, featuring four nacelles per wing due to the reduced propulsion-system volume enabled by electric power. In the present model, a single nacelle is considered, as the current implementation focuses on local wing-nacelle integration rather than nacelle multiplicity. Although the reference concept is purely battery-electric, the internal

propulsion architecture of battery-electric and FC-powered aircraft is similar in terms of component volume and placement. The nacelle scale is therefore representative of FCFC-aircraft, making this a relevant reference case for SR-2 - Realistic WNI Representation. The real Elysian concept is shown in figure 8.4.

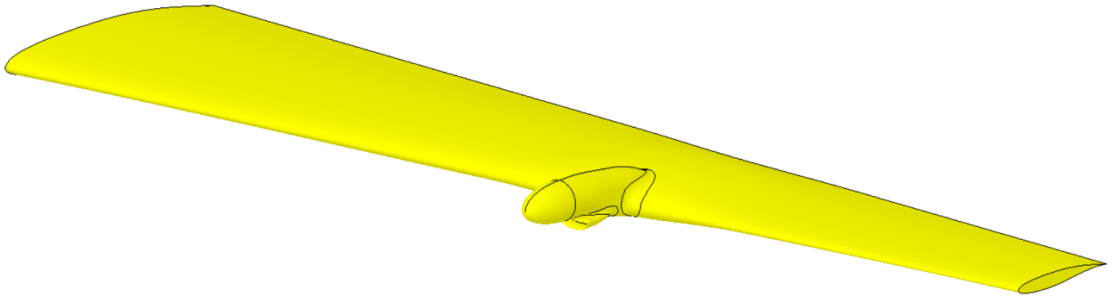
The recreated *Elysian* design includes two key enhancements over the reference concept: a scoop intake, which is absent in the published design, and optional wing-nacelle blending for improved aerodynamic continuity. The resulting geometry, visible from the isometric and side views in figure 8.5a and figure 8.5b, closely resembles the external appearance of the original Elysian demonstrator while maintaining full parametric control.

This example also reveals limitations of the current wing-nacelle blending strategy. Because the nacelles are located very near the leading edge, the present uniform wing-matching approach (section 5.5) is not ideal when a nacelle does not span a large portion of the local chord. In such cases, wing matching should be concentrated toward the leading-edge region rather than applied uniformly over the entire profile thickness. As concluded in section 5.5.3, a full free-form deformation (FFD)-based matching procedure would allow chordwise-dependent scaling, providing a more flexible and physically correct integration.

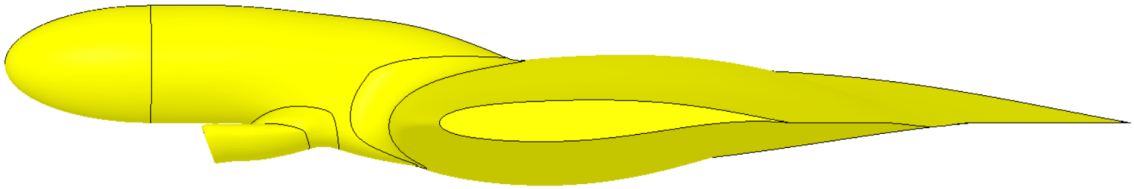
A second highlighted limitation arises from the combination of scoop and wing blending. When the scoop attaches close to the leading edge, the nacelle-scoop blend cannot geometrically overlap with the wing-nacelle fillet without causing failure of the filleting algorithm. This constrains acceptable scoop placement and blocks certain high-potential designs where the intake interface lies further aft and the duct is integrated more towards the nacelle rear. A more advanced integration approach—potentially through an extended `BlendSolid` able to handle multiple simultaneous surface blends—would remove this restriction. In addition, as noted in section 4.5.5, the current `BlendSolid` implementation is not yet robust enough to preserve leading-edge shape fidelity in wing-nacelle joining and therefore needs further development before it can replace the existing integration algorithm.

Overall, the Elysian case confirms the tool's ability to model next-generation FC-representative propulsion configurations while simultaneously identifying key directions for improved blending flexibility in future work.

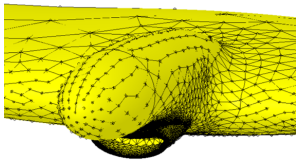
Figure 8.4: Reference photograph of the battery-electric Elysian concept.



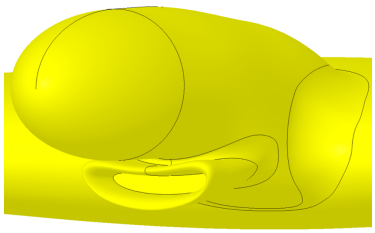
(a) Iso view



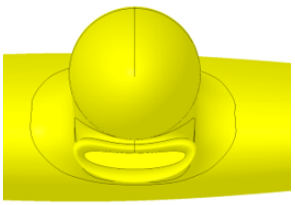
(b) Side view



(c) Nacelle focussed iso view with automatic mesh



(d) Intake focussed iso view



(e) Frontal view

Figure 8.5: Parametric reference model resembling the Elysian battery-electric concept, generated using the developed tool.

8.1.3. Design Space Coverage and Robustness

One of the primary objectives of the tool is to generate WNI configurations across a large and diverse design space, as stated in Hypothesis 1. These requirements enforce that the tool must be fully parametric and highly flexible in the designs it can create. However, flexibility comes at the cost of *parametric robustness*, defined by Sóbester and Forrester as the "ability, in terms of design space proportion, to yield physically and geometrically sensible shapes" [62]. Following a similar methodology to Hillen [59] geometrical robustness in this work is defined as the proportion of the design space yielding *physically sensible* and *geometrically valid* shapes. The criteria for *geometrically validity* are explicitly defined in section 8.1.1. *Physically sensibility* is enforced using the sampling methodology from next section.

The overall parametric robustness is quantified by sampling N_{sample} designs and performing the full geometrical assessment on each, leading to the following definition of robustness:

$$\eta_{\text{para}} = \frac{N_{\text{feasible}}}{N_{\text{sample}}} \tag{8.1}$$

Design Space Sampling

The parametric design is represented as $\mathbf{x} \in \mathbb{R}^n$, where $\mathbb{R}^n$ denotes the full design space with n input parameters. Not every point in $\mathbb{R}^n$ produces a physically meaningful configuration, therefore the physically admissible physical design space is defined as the bounded subset in equation (8.2). If a sample set of size N is defined as $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathcal{X}_{\text{phys}} \subset \mathbb{R}^n$, then the parametric robustness can be written as equation (8.4), where the indicator function in equation (8.3) represents the full geometrical verification process from section 8.1.1.

$$\mathbf{x} \in \mathcal{X}_{\text{phys}} \subset \mathbb{R}^n \tag{8.2}$$

$$I(\mathbf{x}_i) = \begin{cases} 1, & \text{if geometry and verification succeed for } \mathbf{x}_i, \\ 0, & \text{otherwise.} \end{cases} \tag{8.3}$$

$$\eta_{\text{para}} = \frac{1}{|S|} \sum_{\mathbf{x}_i \in S} I(\mathbf{x}_i), \qquad S \subset \mathcal{X}_{\text{phys}} \subset \mathbb{R}^n. \tag{8.4}$$

The parametric robustness metric depends on both the intrinsic performance of the model and the chosen sample space, and therefore cannot be expressed as a single, model-independent value. Instead, robustness must be evaluated with respect to a specific, physically sensible design space representative of typical model usage. Consequently, a unique robustness value η_{para} exists for each defined physically admissible design space $\mathcal{X}_{\text{phys}}$.

Sampling is performed using Latin Hypercube Sampling (LHS) [71] via `qmc.LatinHypercube` from the `scipy` package[1]. LHS partitions the design space to avoid clusters or sparsely filled regions typically seen in grid-based sampling and is a popular choice for such design studies [59].

The physical design space for this study is constructed by selecting which input parameters to sample and by defining appropriate bounds, in terms of *parameter specifications*. As shown in section 8.1.2, strong dependencies exist between several input parameters, specifically for dimensional ones. Therefore, bounds must be chosen such that related inputs remain physically consistent (e.g., a nacelle should not be fully inside the wing due to a small nacelle size, aft positioning and large chord).

Parameter specifications are defined as:

- `ContinuousParam`: numerical value with absolute *min* and *max*
- `BoolParam`: boolean value with a *bias* toward true
- `SequenceParam`: sequence of numbers, each with absolute *min* and *max*
- `RelativeToFloatSlotParam`: numerical value relative to another slot value in the model with relative *min* and *max*
- `RelativeSequenceToFloatSlotParam`: sequence defined relative to other float-slot values

Additionally, parameter specifications *dependencies* ensure that related inputs remain physically sensible to each-other, such as requiring equal-to, not-lower-than, or not-higher-than relationships between specific parameters (e.g., integration positioning constraints).

Finally, all parameter specifications are grouped into categories that align with the failure-code groups in table 8.1. This allows evaluation of robustness not only for the full design space $\mathcal{X}_{\text{phys}}$, but also for targeted subsets, providing insights into where the model is strong and where further development is required. All parameter specifications and groups are listed in chapter D.

Results

The results of the parametric robustness study are presented in figure 8.6. The analysis is performed within a bounded physical design space $\mathcal{X}_{\text{phys}}$, defined through a set of input specifications listed in table D.1. The motivation and reasoning behind the selection of this bounded space are detailed in chapter D, where representative examples of successfully generated designs are also shown to provide qualitative insight into the explored design space.

Using the defined parameter bounds and the sampling strategy introduced in the previous section, a total of 200 design samples were generated and evaluated. Out of these, 59 designs converged successfully, corresponding to an overall success rate of 29.5%. A substantial fraction of failures is caused by mesh divergence, accounting for 45 cases (22.5%), while the remaining failures (48%) are attributable to geometrical modelling issues.

[1] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.qmc.LatinHypercube.html`

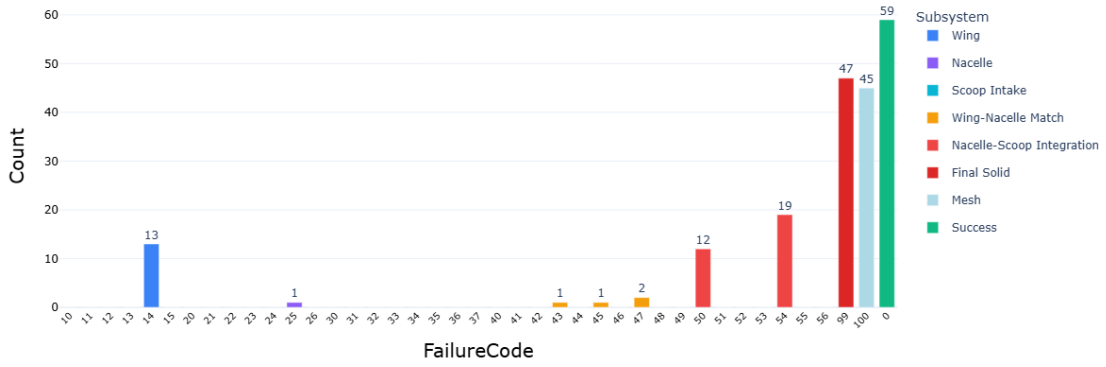


Figure 8.6: Histogram of failure codes during parametric robustness study, coloured by subsystem category.

Mesh Divergence

Mesh divergence does not provide explicit error messages, which prevents an unambiguous identification of its root cause. Nevertheless, visual inspection of the first ten designs exhibiting mesh divergence reveals that eight cases are caused by unrealistic nacelle-scoop integration geometries originating from limitations of the `BlendSolid` algorithm.

In six of these cases, the blend surface exhibits a pronounced outward bulge at the front of the scoop diverter, leading to very large angular deviations of the `BlendSolid`, on the order of several tens of degrees. In the remaining two cases, such bulges are absent, but the blending surface is excessively sharp and sheared. Manual adjustments of the parametric inputs governing both the blend solid and the design parameters resolve mesh convergence issues in all observed cases. This strongly suggests that the non-meshable surfaces are a direct consequence of the limitations of the `BlendSolid` algorithm.

The remaining two mesh divergence cases are caused by geometric overlap between the wing-nacelle and nacelle-scoop integrations. Owing to the limited number of inspected cases, no statistically rigorous quantitative attribution of mesh divergence causes can be made.

Geometrical Failures

Among the geometrical failures, 47 cases (49%) occur at the final wing-nacelle fillet stage (failure code 99). Of these, 17 failures originate from limitations of the filleting algorithm itself, while the remaining 30 are caused by a scaling limitation (section 5.5.3) within the `BlendedLiftingSurface`, which constitutes a modelling error. Further visual inspection of the filleting failures indicates that approximately half originate from insufficient robustness of the filleting algorithm, while the other half arise from modelling inconsistencies. An example is the `BlendedNacelle` *new starting cross-section* being located upstream of the cutting plane, resulting in an excessively curved nacelle geometry. As filleting error messages are not sufficiently descriptive, no exact quantitative attribution can be made.

Failures in the nacelle-scoop integration's `BlendSolid` use (failure code 56), originate predominantly (75%) from modelling errors related to the *restricted geometric applicability* of the `BlendSolid`, as discussed in section 6.6.3. The remaining failures in this category are caused by algorithmic errors in the Gordon surface construction.

All failures associated with the nacelle-scoop duct Gordon surface (failure code 54), comprising 19 cases, are attributed to insufficient algorithmic robustness of the Gordon surface implementation. The TiGL-based implementation (section 4.4.2) reports errors indicating non-intersecting input profiles or guide curves. However, in the present modelling approach, all guide curves are generated by interpolating points sampled directly from the profiles. By construction, this guarantees intersection between profiles and guides, leading to the conclusion that these failures are due to numerical or robustness limitations of the Gordon surface algorithm rather than invalid geometric inputs.

Finally, the single failure associated with the nacelle scaling factor (failure code 43) is exclusively caused by modelling errors. The remaining 17 failures (failure codes 47, 45, 25, and 14) similarly originate from algorithmic divergence of the Gordon surface, following the same reasoning outlined above.

In summary, out of the 96 geometrical failures observed, 40 cases (42%) can be attributed with high confidence to modelling errors, 39 cases (40%) to insufficient robustness of the Gordon surface algorithm, and the remaining 17 (18%) cases to failures of the filleting algorithm. While these filleting failures cannot be conclusively classified as either modelling or algorithmic errors, they are likely to be approximately evenly distributed between the two.

Effect of Starting Design

The robustness study presented in figure 8.6 uses the *baseline* configuration as the starting design, from which all input parameters are varied according to the sample specifications defined in chapter D. To assess the influence of the starting design, the same robustness analysis is repeated using the exact same sample set S, but with the *Havilland Dash 8* and *Elysian* configurations as starting points. These additional studies yield overall success rates of 33% and 36.5%, respectively.

The distributions of failure codes across the three robustness studies exhibit strong similarity. Minor differences are observed in that the *Havilland Dash 8* study shows a relatively higher occurrence of wing-nacelle filleting failures (failure code 99) and fewer mesh divergence failures (failure code 100). The *Elysian* study, in contrast, exhibits a noticeably lower number of mesh divergence failures, which explains its slightly higher overall success rate.

As these values are close to the 29.5% success rate obtained for the *baseline* configuration and due to the similarity of failure distributions, it can be concluded that the observed *parametric robustness* is largely independent of the chosen starting design. This indicates that the adopted sampling strategy and parameter bounds define a design space that is sufficiently broad such that all three manually constructed reference designs are effectively contained within it. Consequently, the bounded design space $\mathcal{X}_{\text{phys}}$ can be considered large enough to encompass a wide variety of distinct configurations. Visual inspection of some generated designs in section D.1 further strengthens this conclusion.

8.1.4. Conclusion and Recommendations

The developed tool demonstrates robust parametric behaviour when designs are created manually or when automatically generated designs are interactively corrected. This is evidenced by the successful construction of several manually defined reference configurations and by the large extent of the design space explored in the parametric robustness study. Together, these results confirm that the tool is capable of generating geometrically diverse and substantially different configurations.

However, fully automatic design generation across the entire bounded design space cannot yet be achieved with complete robustness. When sampling within a design space that is sufficiently large to confidently encapsulate all three manually defined reference designs, the observed *parametric robustness* is approximately 30-35%. This indicates that, while the design space allows for the generation of vastly distinct configurations, only about one third of automatically sampled designs can be expected to converge successfully without user intervention.

As the *parametric robustness* is inherently dependent on the definition of the bounded sample space, no single absolute performance metric can be assigned to the tool. The reported success rate should therefore be interpreted as indicative rather than definitive. Nevertheless, for fully automated design exploration, the results suggest that a success rate on the order of one third is a realistic expectation for the current implementation.

Despite the indicative nature of the absolute performance, the verification study enables clear conclusions to be drawn regarding the distribution of failure causes. Approximately one third of all failures originate from limitations of the `BlendSolid`, around one quarter from insufficient algorithmic robustness of the Gordon surface, roughly 35% from modelling errors, and the remaining portion from filleting algorithm divergence.

Although no single definitive quantitative robustness metric can be extracted, the verification study clearly identifies several priority areas for improvement:

- **Improved Gordon surface implementation** is strongly recommended, as a significant fraction of failures can be attributed to robustness limitations or likely implementation issues in the TiGL-based Gordon surface (section 4.4).
- An **extended** `BlendSolid` formulation would improve the quality and smoothness of the generated blending surfaces, thereby increasing suitability for downstream meshing and simulation workflows. Specific important limitations are *quasi-tangency* and the absence of an *optimal blending solution*, all discussed in section 4.5.5.
- **Improved integration modelling** is required to enable fully automated design exploration. A substantial portion of failures originates from modelling inconsistencies that become apparent when random samples push the geometric limits of the tool. Addressing these issues would also increase the physical validity of the resulting blending solutions as in its current stage no realistic wing-nacelle fully blending solutions can be generated.
- **Extended model features** would broaden the applicability of the tool beyond the current scope of fuel-cell-powered aircraft, enabling exploration of a wider range of propulsion and integration concepts.

8.2. Validation

The primary objective of this tool is to generate valid external geometries that are physically representative of WNI configurations for FC-powered aircraft. Therefore, a validation study has been conducted to assess the fidelity of the generated geometry relative to an experimentally tested and computationally documented reference design. This effort directly addresses SR-2-D3 - Geometry Validation by evaluating whether the model outputs can replicate a real configuration.

The validation focuses on two aspects:

1. **Geometric validity:** confirming that the generated geometry is suitable for engineering simulation and representative of existing experimental configurations.
2. **Modelling methodology validity:** validating that the parametrisation methodology and design philosophy enables accurate reproduction of real designs in a structured, intuitive, and efficient manner.

Because the OpenCascade kernel is a mature CAD technology widely used in aerospace applications, the geometric robustness of individual surfaces is expected. The key contribution of this study is therefore assessing whether the *design philosophy* and *parametric modelling workflow* developed in this thesis can accurately represent realistic designs, consistent with the intended use of the tool.

8.2.1. Validation Case

The chosen validation model is the TU Delft fropeller-fing-flap (TUD-PWF) wind-tunnel model. This model is designed to gain insight into primary flow phenomena in propeller-wing-flap interactions and how they differ from more traditional propeller-wing[2] and multi-element aerodynamics as the application of propellers in high-lift performance for next generation aircraft concepts[3] is emphasised [72]. A comprehensive aerodynamic analysis of the configuration is documented in the doctoral research of Duivenvoorden [73], to which the reader is referred for full experimental results and flow-field interpretation.

Motivation for Study Choice

This validation case studies aerodynamic integration of propulsion with fixed-wing lifting surfaces in a context explicitly linked to future electric and distributed propulsion aircraft. As discussed in section 2.1, electric drivetrains are a core element of FC-powered aircraft, making this configuration highly representative. The case therefore satisfies SR-2 - Realistic WNI Representation, SR-2-D3 - Geometry Validation and SR-2-D4 - Knowledge Integration.

A limitation is that the reference model does not include an inlet system, neither a scoop nor a conventional intake. Due to the lack of detailed intake reference data, a full WNI validation including inlet

[2] Briefly discussed within the context of this work in section 2.3.
[3] Distributed electrical propulsion is a prime example of these concepts.

geometry remains recommended future work. For the same reason, blending parametrisations (sections 6.6 and 6.7) are not validated here. This is expected, since such integration concepts are at the core of the novelty and motivation behind the tool.

Experimental Setup

The model is an unswept and untapered wing spanning the full test section clamped from wall-to-wall with a span of 1.248 m (almost equivalent to test section height). The wing features the NLF-Mod22(B) airfoil, seen in figure 8.7 of which detailed coordinated data is available, with 0.3 m chord. This airfoil includes a nested slotted flap, capable of nested, $15°$ and $30°$ deflection settings. Since the tool does not contain any high-lift devices, only the nested flap configuration is relevant. The model contains a cylindrical nacelle with of a length where the propeller is at $0.85D$ from the wing LE. The propeller used is the TUD-XPROP-S with 203 mm diameter. The full model setup including technical drawing with relevant dimensions can be seen in figure 8.8. On a final note, the full CAD geometry files of the wing with flap and nacelle can be found on [74] and propellor geometry data including spinner, is found on [75].

Full experimental procedures and corrections are described in [72].

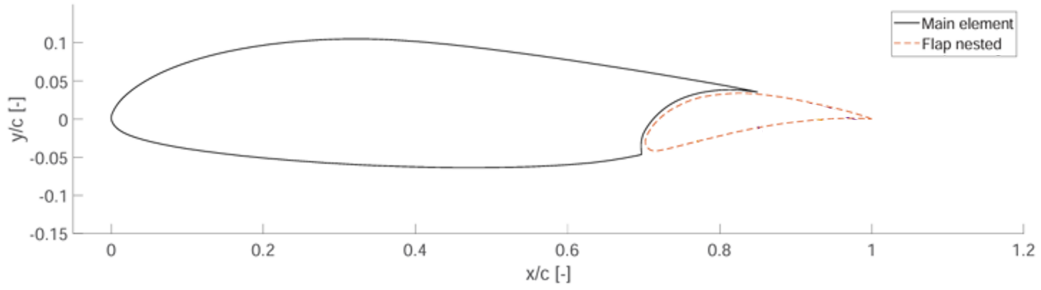


Figure 8.7: NLF-Mod22(B) airfoil with nested flap [72] (modified).

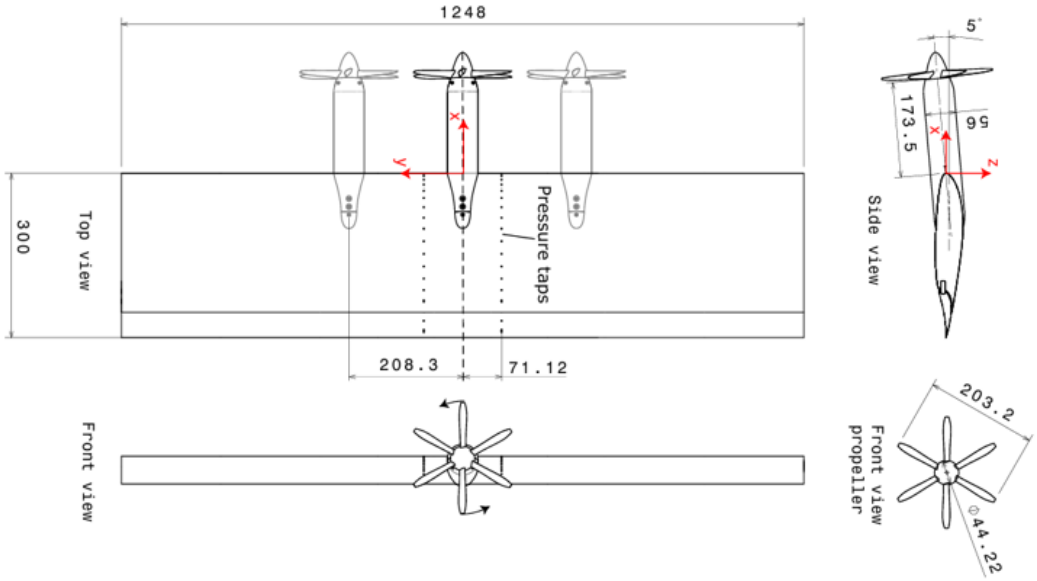


Figure 8.8: Technical drawing of the TUD-PWF reference model with key dimensions (mm) [73].

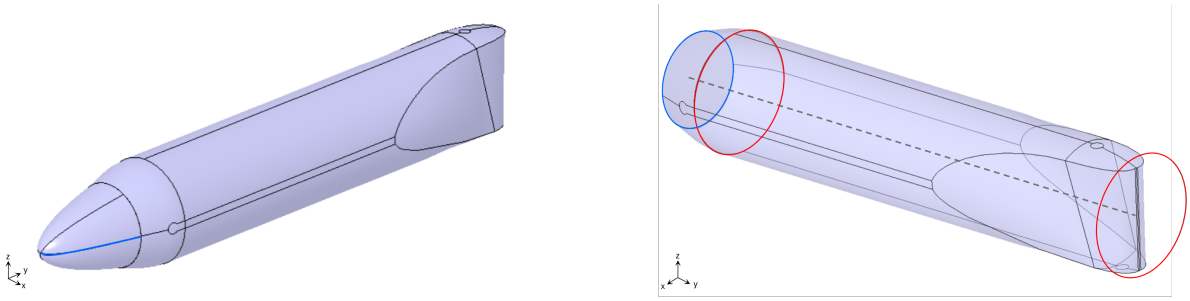
The experiments run are with combinations of propeller and nacelle on- and off configurations and at various flap angles. The experiment takes flow measurements while sweeping over the angle-of-attack. In this study, only the polar results for model configuration with nacelle and wing with nested flap are relevant, denoted CF 9 in [72], as flaps and propellers are out of scope for the tool in this stage.

Rebuilding the Geometry in the Tool

The reference STEP geometry is imported into the tool via the `STEPReader` and aligned to the tool's global coordinate system, with the wing-root leading edge placed at the origin, streamwise direction along y, and spanwise along x. The reconstruction uses the same parametrisations introduced in previous chapters, where all required values are extracted either directly from figure 8.8 or through inspection of the reference CAD.

A `RectangularPlanform` and the clean-wing parametrisation (section 5.3.3) are used with zero twist and dihedral. The NLF-Mod22(B) airfoil is imported and fitted to CST using the approach from section 5.2.4. Because the reference dataset includes both wing and flap geometry at zero flap deflection, the gap is removed by discarding points inside the slot region, generating a single continuous profile.

The nacelle is a concrete `Nacelle` with two circular cross-sections at the start and in the middle, both with a diameters from the technical drawing and a final circular cross-section at the end with diameter matching the spinner's. This last cross-section is the result of the spinner at the end of the nacelle as seen in figure 8.9b. For that reason, the nacelle is created using two circular CST cross-sections with nacelle diameter, located at start and 0.907 of full nacelle length which comes from analyses of the loaded reference geometry. The nacelle length itself is obtained by measuring the middle line from the reference geometry computationally. Finally, the spinner is created using the *direct* spinner curve parametrisation where the input curve is one of the revolving edges of the spinner reference geometry seen in figure 8.9a. The diameter of this spinner is furthermore obtained from the technical drawing.

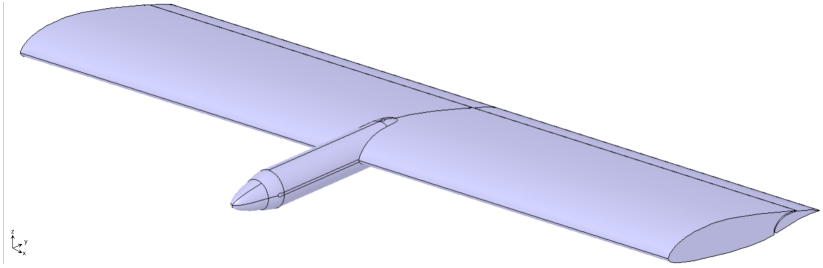


(a) Frontal isometric view where the edge used for the spinner shape parametrisation is highlighted in blue.

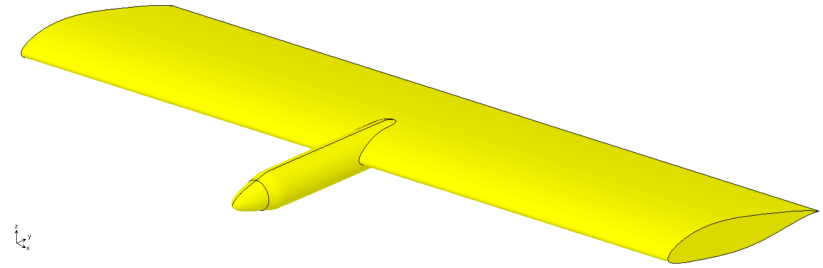
(b) Backward isometric view where the cross-section used for the spinner radius is highlighted (blue) and the nacelle cross-sections for the tool nacelle (red).

Figure 8.9: TUD-PWF nacelle reference geometry, visualised in the ParaPy viewer.

The nacelle is positioned via visual inspection of the geometry. Subsequently, to integrate the nacelle with the wing, the process from section 6.7 is used where the nacelle is cut at 0.082% with an uniform scale distribution (nacelle is fully matched towards the wing). The *fuse* option for wing-nacelle integration is used as no blending is applied at the wing-nacelle boundary. This manual computation of model inputs leads to a final geometry seen in figure 8.10.



(a) Imported reference geometry.



(b) Parametric reconstruction generated using the tool.

Figure 8.10: TUD-PWF validation model within the tool.

8.2.2. Geometry Validation

Geometric validity of the wing-nacelle configuration is assessed by direct comparison with a reference wind-tunnel CAD model. The generated surface is uniformly sampled with approximately 20 million points at equal surface-area density, avoiding curvature-induced bias and ensuring all regions contribute equally. For each point, the shortest distance to the reference geometry is computed using CloudCompare[4], yielding a continuous deviation field and global error statistics. As the reference model includes limited blending, global G^1/G^2 continuity is not evaluated here. Instead, continuity is analysed separately for the `BlendSolid` algorithm in section 4.5.4.

The geometric validation results are summarised in table 8.2 and figure 8.11. The RMS deviation is 0.391 mm (0.13% c), the 95th percentile is 0.528 mm (0.18% c), and the maximum deviation is 1.475 mm (0.49% c), occurring at the lower side of the wing near the flap. Overall, the parametric reconstruction closely matches the reference geometry, with all metrics well below the SR-10 - Geometry Fidelity limit of 1% chord, while localised larger deviations highlight areas requiring further investigation presented later in this section.

Table 8.2: Geometric deviation metrics between tool-generated and reference geometry

Metric	Value	Rel. to c
RMS distance	0.391 mm	0.13%
Mean	0.250 mm	0.08%
Max	1.475 mm	0.49%
P95 distance	0.528 mm	0.18%
Std. dev.	0.300 mm	0.10%
Points amount	20 million	N.A.

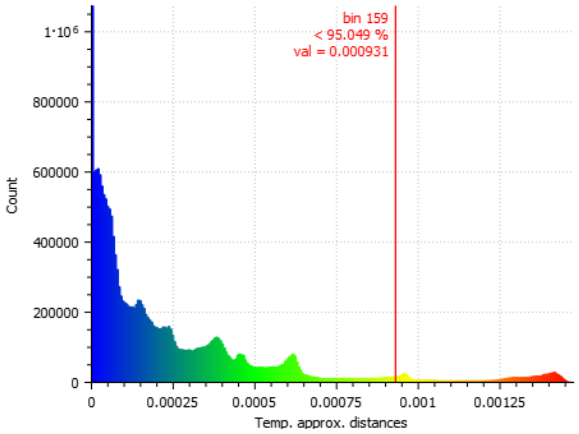
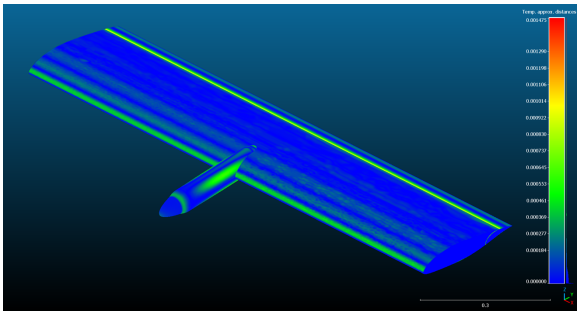


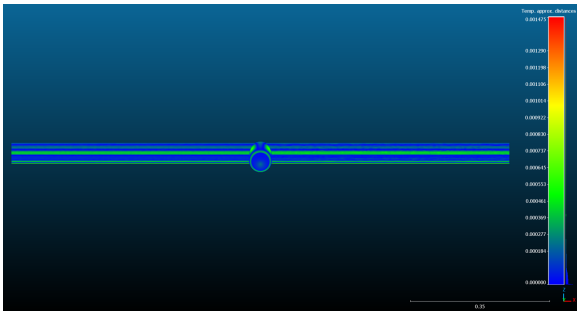
Figure 8.11: Histogram of point-to-surface distances computed by CloudCompare.

[4] `https://www.cloudcompare.org/`

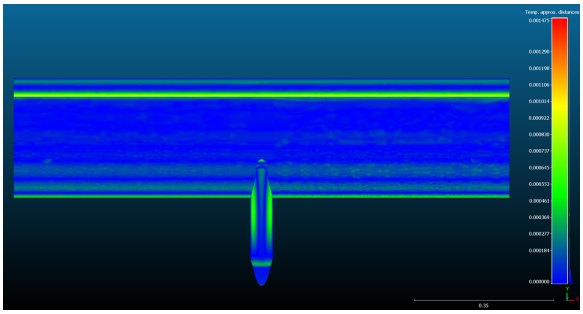
The spatial deviation distribution is shown in figure 8.12. The upper wing surface matches the reference closely, with a localised deviation at the flap slot, which is not modelled in the tool making this an expected deviation. Larger deviations occur on the lower surface, particularly near but not at the flap, and remain nearly constant along the span, indicating the origin of error to be from the two-dimensional airfoils rather than three-dimensional effects. These deviations are likely due to airfoil instantiation inaccuracies arising from mismatches between the airfoil coordinate data and the reference STEP geometry. Aside from these systematic differences, overall geometric agreement is very good.



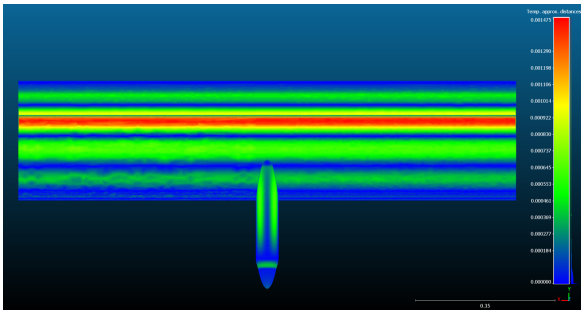
(a) Isometric view.



(b) Front view.



(c) Top view.



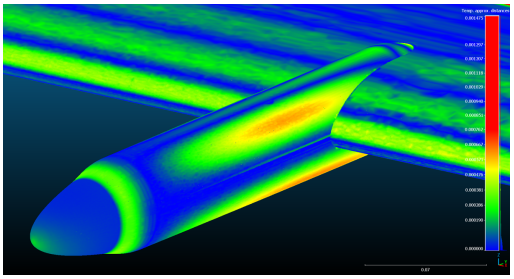
(d) Bottom view.

Figure 8.12: Point-to-surface distances between tool-generated geometry and reference CAD for TUD-PWF model, visualised in CloudCompare.

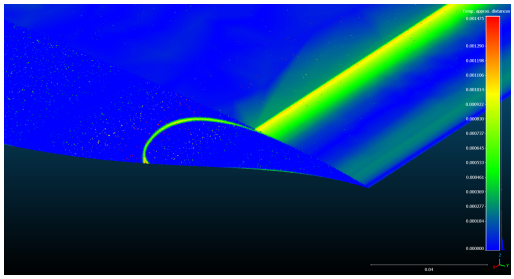
A second region of interest is the nacelle, particularly the corners and spinner-nacelle transition. As shown in figure 8.13a and figure 8.13c, the largest deviations occur in the middle between prescribed nacelle cross-sections, where the Gordon surface is unconstrained by both profiles and guides. At the crown, keel, and MHB lines, deviations are near zero, while between these guides the surface bulges outward, producing a lobed error pattern. This effect, absent on the wing, confirms it as a characteristic of implemented Gordon surface algorithm interpolation with sparse and curved cross-sections and guides.

A further discrepancy is visible near the spinner-nacelle junction on the nacelle side of the surface. This arises because the nacelle is reconstructed solely from cross-sectional data, without explicitly constraining the longitudinal profile in this region. This choice preserves the flexibility of the `NacelleFromCrossSections` parametrisation for redesign purposes, at the cost of a small local deviation from the reference model. The magnitude of this deviation remains well below the global RMS error.

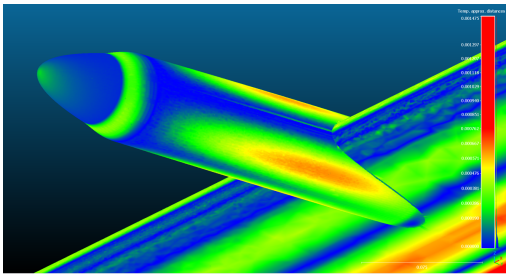
At the wing-nacelle intersection, the deviations are very small and close to zero. This indicates that the geometry-matching and fusion procedure from section 6.7 produces a realistic and accurate representation of the local integration region for this test case.



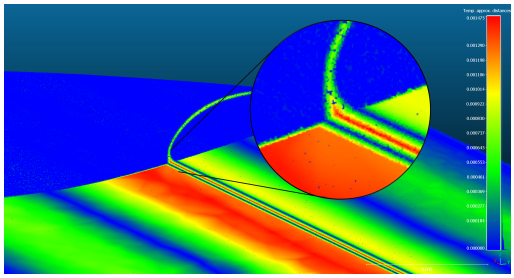
(a) Nacelle-focussed top view (colour scale shifted for clarity).



(b) Trailing-edge-focussed top view.



(c) Nacelle-focussed bottom view (colour scale shifted for clarity).



(d) Trailing-edge-focussed bottom view.

Figure 8.13: Zoomed views of key regions showing point-to-surface distances in CloudCompare.

The third key region is the wing trailing edge, again particularly near the flap slot. In this area, deviations increase significantly, reaching the global maximum on the lower surface. It should be noted that only a small fraction of points actually lie inside the geometric gap itself. Most large deviations are associated with points on the lower wing surface not part of the slot. Consequently, the presence of the flap slot and the lack of an explicit high-lift device in the tool may influence the maximum but have only a limited effect on the global RMS and mean error.

The increased deviations near the upper flap slot arise directly from the geometric offset visible in figure 8.7, where the nested flap sits below the main-element trailing edge. Other larger discrepancies on the lower surface and near the leading edge are spanwise uniform and therefore mainly attributable to airfoil modelling inaccuracies. Removing these spanwise-constant regions (above $\sim$0.5 mm) would reduce the maximum deviation by about 65% and lower the RMS and mean deviations to roughly 0.20 mm and 0.15 mm, representing approximate reductions of 50% and 40%, respectively.

In conclusion, the tool can generate a fully parametric TUD-PWF geometry with conceptual-design fidelity, achieving RMS deviations of about 0.1% c. The dominant errors stem from airfoil reconstruction and Gordon-surface interpolation between curved guides and sparse cross-sections. A CAD-based initialisation (e.g. STEP airfoil extraction) is expected to reduce these errors by more than a factor of two, after which Gordon-surface behaviour would dominate with residual deviations around 0.2%. Overall, the proposed wing, nacelle, and integration methodology provides a realistic, robust, and flexible framework for parametric external geometry generation.

8.2.3. Aerodynamic Validation

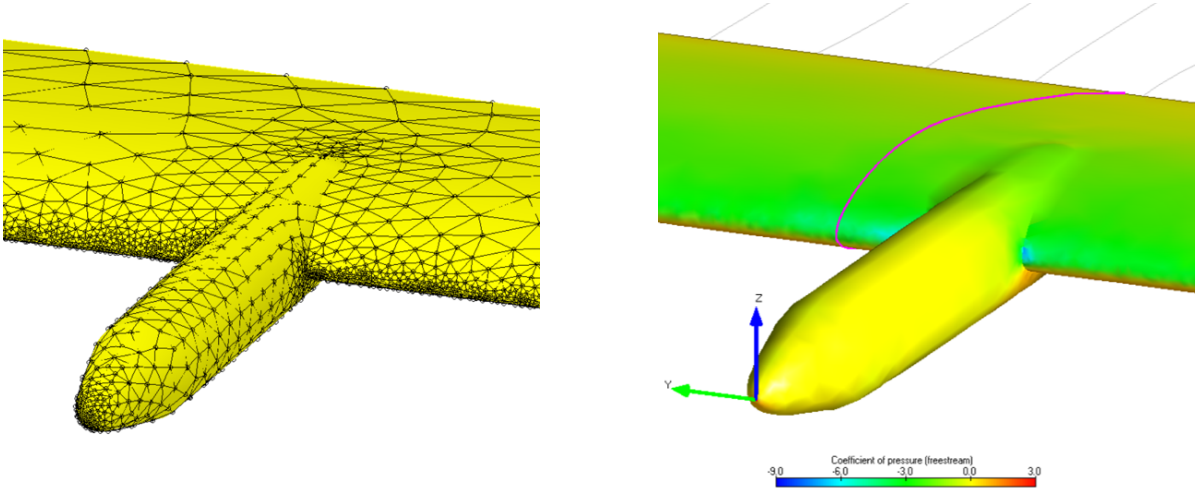
As an additional validation step, the aerodynamic results obtained directly from the integrated analysis module are compared with the experimental data of the validation case. For this purpose, configuration 9 from the TUD-PWF campaign is used, which corresponds to the wing with nacelle installed and no propeller. Based on the previous section, the geometry generated by the tool is considered representative of the actual wind-tunnel model and is therefore used as input for the aerodynamic simulation.

The simulation uses the automated aerodynamic workflow from section 4.3.4 with the default unstructured surface mesher (section 4.3.3), without manual refinement or mesh optimisation with the result seen in figure 8.14a. No mesh-convergence study is performed, so mesh effects cannot be isolated from other error sources. The results should therefore be interpreted as an *order-of-magnitude* validation, aimed at assessing whether the automatically generated geometry and default settings reproduce

experimental aerodynamic trends rather than providing a fully validated solver setup.

FlightStream (FS) is a vorticity panel method that enhances the original potential flow solution with novel features[5] like compressibility models, integral BL models, viscous-coupling, and separation models [76]. However FS cannot model wind-tunnel sidewalls by means of explicit boundary conditions. As wall suction was used in the experiments, the central wing flow may be approximated as an infinite wing. To emulate this, the tool model wingspan is increased by a factor of 6.41 to a total span of $8\,\mathrm{m}$, beyond which computation time becomes prohibitive. Rather than comparing global C_L, validation focuses on local aerodynamic coefficients and pressure distributions near the nacelle at the pressure-tap spanwise location on the starboard side (figure 8.8), which are also available in FS (figure 8.14b). The simulated flow conditions match the experiment and for the exact numerical values, the reader is referred to [72, 73].

To avoid uncertainties in complex viscous modelling with the assessment of geometric validity, the simplest solver configuration is used. For that reason the viscous-inviscid coupling and separation modelling in FS are deliberately disabled for this aerodynamic study. FS's separation model is highly sensitive to mesh quality, and the developers strongly recommend dedicated mesh-convergence studies when using these advanced options. To conclude, the final simulation is an inviscid potential-flow vorticity-based panel method simulation with decoupled BL analyses and no flow separation modelling.



(a) Automatic unstructured surface mesh generated by the meshing module in the tool.

(b) Pressure field around the wing–nacelle integration in FlightStream at $\alpha = 11.002°$, with the starboard validation section highlighted in purple.

Figure 8.14: Simulation setup for the TUD-PWF configuration generated by the tool.

The comparison of lift polars in figure 8.15a shows very good agreement in the linear lift regime ($\alpha \in [-5°, 6°]$), with differences below ~ 0.03. At higher angles of attack, discrepancies increase due to missing separation, the inviscid nature of the simulation, and the likely due to the lack of mesh optimisation, without a single dominant cause identifiable as that is out of scope for this validation study.

The comparison of pressure distributions in figure 8.15b at a high angle of attack of $\alpha = 10°$, corresponding to the end of the linear region of the lift polar, shows good overall agreement between simulation and experiment. Both the magnitude of the pressure peaks on the upper and lower surfaces and the general shape and trend of the pressure distributions are captured well by the simulation, indicating that the dominant lift-generating flow features are reproduced correctly.

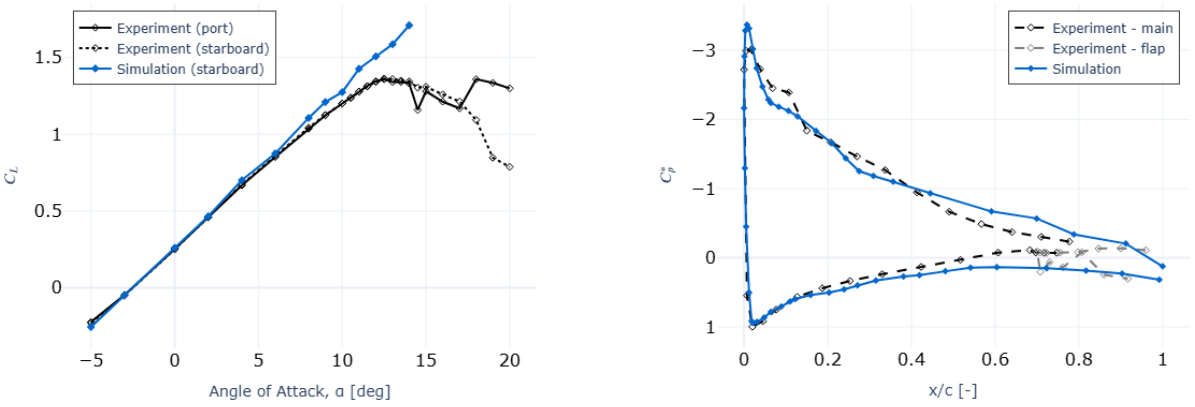
Three main discrepancies are observed. First, the simulation predicts a consistently higher pressure differential between upper and lower surfaces than the experiment, likely due to the absence of viscous-inviscid coupling. Without BL feedback, the inviscid solution permits stronger acceleration and pressure recovery, effectively increasing the effective camber and thickness, particularly toward the aft region

[5] Full list of all FlightStream's features and underlying aerodynamic theory can be found in [76].

which is consistent with the streamwise growth of the BL thickness. This provides a plausible explanation for the slightly higher lift coefficient predicted in the lift polar (figure 8.15a).

Second, relatively large local fluctuations are observed on the lower surface. These can partly be explained by geometric deviations between the experimental and simulated airfoil profiles (section 8.2.2), while unmodelled viscous, stall, and mesh effects may also contribute to the observed fluctuations.

Finally, the experimental pressure distribution shows clear main-element-flap interaction effects near the hinge region, including reduced pressure differential and increased fluctuations. These are absent in the simulation, which models the wing as a single smooth surface without a discrete flap, and therefore cannot capture these local flow phenomena.



(a) Local lift polar at starboard side.

(b) Pressure distribution at starboard side for α = 10°.

Figure 8.15: Simulated aerodynamic data from FlightStream compared with uncorrected windtunnel data [72].

Two definite qualitative conclusions can be drawn. First, the pressure field in figure 8.14b confirms that FlightStream captures the primary wing-nacelle interaction effects on the generated geometry, despite the simplified setup. Second, and most importantly in the context of this thesis, the automatically generated geometry is shown to be fit for aerodynamic analysis without any hand-tuned subsequent geometry processing a the resulting predictions are of the correct order of magnitude and reproduce the experimental trends well in the attached-flow regime.

This exercise is not a full validation of the integrated aerodynamic analysis module, but an intermediate demonstration of the tool's potential to evolve into a complete aerodynamic framework for WNI configurations. While the current implementation is not yet suitable for reliable quantitative evaluation, the equivalence between the aerodynamic results show that the underlying geometry model is robust and that the automated workflow provides a solid basis for future, properly validated simulation capabilities with high potential.

8.2.4. Conclusion and Recommendations

A key objective was to demonstrate that the tool can generate realistic external geometries for FC-powered WNI integrations (SR-2 - Realistic WNI Representation). Geometry validation shows that a representative wind-tunnel model is reproduced with typical deviations of about $0.1\,\%$ of the chord. Remaining errors mainly stem from manual initialisation and Gordon surface limitations, confirming the methodology's suitability for conceptual wing and nacelle design. However, this conclusion cannot yet be extended to fully blended WNI configurations.

The built-in aerodynamic analysis module shows strong potential toward satisfying SR-5 - Aerodynamic Assessment. Although current predictions are not yet fully reliable due to automatic meshing and limited solver tuning, the results capture correct aerodynamic trends and agree well with experiments in attached-flow conditions. This confirms the suitability of the generated geometry and provides a solid basis for further development toward an integrated design and analysis workflow.

Based on the findings of this validation effort, the following recommendations are made:

- **Extended WNI validation cases:** The present study only assessed a simplified wing-nacelle configuration without an intake. Future work should include cases with scoop intakes and blended integrations to fully validate the novel modelling capabilities introduced in this thesis.
- **Automated geometry initialisation from CAD:** Many deviations originated from manual model setup and reference model recreation. A robust geometry-based initialisation module would remove user-induced inaccuracies and significantly improve both fidelity and usability.
- **Improved Gordon surface implementation:** The TiGL-based approach exhibits degraded shape conformity between guides. A reimplementation should explicitly address these inaccuracies as well as the requirements introduced by the preceding recommendations (e.g., blended integrations, and algorithmic robustness). In summary, a more robust Gordon surface formulation would represent a highly valuable upgrade to the tool.
- **Dedicated meshing strategy:** Aerodynamic performance strongly depends on mesh quality. A module tailored to the tool's specific geometrical characteristics is required to enable a dependable and automated aerodynamic assessment workflow.
- **Improved aerodynamic integration and validation:** The aerodynamic validation relied on several simplifying assumptions, such as 2D sectional comparison, oversized span approximation for wall-suction conditions, and limited control over internal FlightStream solver settings. As confidence in the present aerodynamic results is low, further research is recommended into solver configuration, coupling strategies, and workflow validation. Progress in this area is inherently coupled to improved meshing capability.

9

Conclusions and Recommendations

This final chapter concludes all findings in this research with concrete answers to the research questions, evaluation of hypotheses and an assessment of to what extent the posed research objective has been fulfilled. Finally, based on these findings and with specific focus on the emerged limitations throughout the report, recommendations are formulated for continued development of the tool and future research.

9.1. Conclusion

The starting point of this work was the recognition that propulsion installation plays a decisive role in aerodynamic efficiency, particularly for FC-powered aircraft, where the absence of conventional turbofan engines opens opportunities for more aerodynamically integrated nacelles and novel intake concepts.

However, the literature review revealed three explicit research gaps (section 1.4):

1. Electrified FC propulsion fundamentally alters nacelle placement, intake function, and TMS-driven sizing requirements compared to conventional ICE architectures. As a result, retrofitting ICE engine nacelles for FC propulsion leads to aerodynamically suboptimal configurations.
2. No generic or systematic methodology currently exists for assessing the integration of FC systems with respect to external nacelle-wing-intake geometry and overall aerodynamic performance.
3. *Unconventional*[1] intake configurations, which may offer significant aerodynamic advantages for FC-powered aircraft, have received little attention in recent research as they may be part of the solution making FC-powered flight possible.

To address these gaps, the scope of the thesis was deliberately limited to the outer mold line of WNI systems, while excluding internal duct and detailed subsystem sizing and external geometry coupling. Within this scope, the research objective was *to develop an extendable parametric tool capable of generating a vast selection of computationally valid external geometries for wing-nacelle-intake studies of fuel cell-powered aircraft.*

Methodology and Requirements

While CADx is well suited for detailed prototyping, KBE was selected as the foundation due to its ability to encode design logic, automate geometry generation, and support scalable parametrisation and tool integration. Its primary drawback, namely its limited native geometric flexibility, was recognised as a risk during tool development given the expected shape complexity.

System Needs were derived from the thesis hypotheses and external stakeholder expectations, from which a fully traceable System Requirements model was constructed. These requirements guided architectural choices throughout the thesis, and their compliance is evaluated in chapter E. Overall, the

[1] Definition of *unconventional* intake configurations is found in chapter 1.

Requirement package provided a rigorous framework connecting research hypotheses to needs finally to concrete system requirements.

Parametric Model

The `SystemAssembly` acts as the top-level object, managing the parametric wing, nacelle, and intake subsystems and their integration through a staged workflow. Components are first generated independently and then combined via matching and blending operations to form a complete WNI configuration, while still allowing individual components or arbitrary combinations to be analysed through configurable inclusion settings.

Several capability modules enhance tool usability and disciplinary scope: coordinate data import, newly introduced enhanced STEP geometry export, automatic meshing, and a FlightStream interface for aerodynamic analysis. Smooth and complex surface generation relies on Gordon surfaces as smooth geometry is found to be critical for aerodynamic simulation.

The introduced concept of *parametrisation-as-composition* decouples core geometry objects from their design views, enabling replaceable parametrisations without structural code changes. It originates from the idea that parametrisation is a *design view* of which unlimited options exists of which some are more suitable for a specific context than others. This flexibility is further supported through `Descriptor` contextualisation to manage parameter dependencies. Together, these mechanisms ensure future extendability as WNI knowledge evolves.

A new `BlendSolid` algorithm enables quasi-tangent blending of any two overlapping solids. A dedicated continuity study demonstrated robust performance and quasi-tangency (RMS angular deviation of $\tilde{1}.0^{\circ}$) for low-complexity intersections and reduced unacceptable tangency fidelity for highly curved shapes while enabling intuitive parametric control of the blend. Limitations prevent the use of this algorithm for wing-nacelle integration but the overall performance enables its use for the nacelle-scoop blend. These limitations are highly coupled to the use of the current Gordon surface algorithm from TiGL [60].

Wing

The wing architecture provides reusable parametric elements such as planforms, lifting surfaces, profiles, and airfoils, each supporting interchangeable parametrisations. CST airfoils and an *airfoil-on-rails* positioning define the baseline airfoil parametrisation and the implementation of the *clean wing* concept enables robust and intuitive design of any lifting surface. To enable future integration, the `BlendedLiftingSurface` locally scales the wing while preserving sectional shape. Key limitations include uniform scaling limitations and the absence of quantified impacts on geometrical fidelity.

Nacelle

The nacelle package follows the same architectural principles and contains spinners, internal system containers, and nacelle cross-sections. Internal volumes can be sized parametrically, but no automatic coupling exists between subsystem geometry and nacelle geometry. Prominent parametrisations are elliptical spinners and CST nacelle cross-sections, although the framework allows for intuitive extension of this list. A `BlendedNacelle` supports robust partial shape modification, though guide control remains restricted.

Wing–nacelle integration relies on geometric matching followed by rolling-ball filleting as the newly introduced `BlendSolid` does not result in required LE shape fidelity and continuities. While robust in execution, shape preservation in the blended region degrades for larger radii, and exact wing-nacelle intersection locations as well as wing scaling cannot be prescribed. As such, the methodology enables integration workflows but does not yet produce geometrically ideal blended solutions leading to the unsuitability of this exact parametric view for highly blended wing and nacelle designs.

Scoop Intake

The intake package introduces reusable `Lip`, profile, and body objects, all *configurable parametric* to support future intake concepts with new parametrisations. The *clean lip* parametrisation provides aerodynamic shape control via Euler-curve centerlines, LE radii and CST shape definition. The `ScoopProfile` derives key intake metrics from paired upper and lower lips, and a `ScoopIntake` generates the 3D geometry using this profile and an intake frontal area definition. Both scoop profile and full 3D object, contain an intuitive parametrisation based on the preliminary parametrisation of the literature study.

Due to high curvature of the scoop LE, Gordon surfaces require additional intermediate profiles to maintain LE shape. A nacelle–scoop integration routine generates a external duct matching scoop with nacelle geometry and subsequently blends it via `BlendSolid`. Limitations include constrained scoop positioning, lack of exact geometric control (e.g. diverter height), limited parameter coupling, and quasi-tangent continuity.

Verification and Validation

Verification assessed geometry validity (completeness, manifoldness, simplicity, robustness, and meshability) through a step-wise evaluation of quantified geometrical checks at around 40 workflow stages. This assessment methodology enables exact analyses of failure location in the workflow with reason, allowing a more structured approach to model fixing.

Manual design feasibility was confirmed through three contrasting configurations, all resembling reference aircraft. Visual inspection revealed some core model limitations specifically for blending operations and integrations and the applicability of the tool to turboprop designs. Additionally, this study concludes that manual design with the tool is robust and leads to vast selection of realistic designs in a large design space.

Parametric robustness was evaluated through automated design sampling. Convergence according to the previously defined criteria, proved sensitive to user-defined parameter ranges. As a significant portion of failures are caused by unexpected Gordon-surface divergence, the robustness of the tool is highly connected to the Gordon surface algorithm, indicating insufficient robustness for fully automated design studies in the current implementation.

Validation was performed using the TU Delft TUD-PWF benchmark, a relatively simple wing-nacelle design representative of FC- and electrically powered aircraft. Geometric accuracy was strong (RMS 0.391 mm or 0.13% chord), where deviations are mainly resulted from airfoil initialisation inaccuracies and Gordon surface interpolation inaccuracies. Aerodynamic predictions using the same experiment, reproduced linear lift trends convincingly but diverged near stall due to unknown reasons as no exact study has been conducted on mesh quality, solver settings and analyses of full aerodynamic results. Thus, the current implementation cannot yet function as an generic aerodynamic assessment tool for FC-representative WNI systems, but shows high potential in this aspect.

9.1.1. Research Questions Answers

RQ 1 — Framework and Methodology

How can a parametric modelling tool be formulated to automatically generate physically valid OMLs for WNI systems representative of FC-powered aircraft?

The tool can be formulated through a Knowledge-Based Engineering (KBE) architecture supported by MBSE-for-KBE to derive a traceable set of functional and non-functional requirements directly from the research objective and stakeholder needs. This allows the resulting methodology to explicitly encode design logic, geometric consistency, and practical workflow constraints into the system architecture.

The implemented approach separates the wing, nacelle, and intake into modular components, each capable of generating a physically valid OML from intuitive design parameters using the inherent geometric construction rules of KBE. These subsystems are integrated via a unified methodology based on *geometrical matching followed by parametric surface blending*, enabling the automatic creation of full 3D WNI geometries.

The tool's correctness in generating FC-representative OMLs is demonstrated through a reference validation case, showing that accurate wing–nacelle geometry can be generated. While not yet validated for full WNI configurations, these results confirm that the chosen framework and methodology provide a technically sound foundation.

RQ 2 — Identification of Aerodynamically Attractive Designs

How can the developed tool be used to identify WNI configurations for FC-powered aircraft that exhibit improved aerodynamic performance metrics compared to conventional podded nacelle designs?

The tool enables designers to explore WNI configurations by systematically adjusting parametrisations that directly influence key aerodynamic characteristics. This includes a rich set of parameters all of

which with known aerodynamic effects. Thus, the tool supports early-stage identification of promising design trends through informed, qualitative aerodynamic reasoning.

However, extensive aerodynamic performance assessment requires validated simulation capability, improved mesh generation and assessment methods, and integrated support for simulating and modelling inlets and exits. Direct comparison to conventional turbofans is currently not possible, as such architectures lie outside the scope of the implemented parametrisations and components. Therefore, at this stage the tool enables conceptual aerodynamic exploration but not quantitative aerodynamic assessment.

RQ 3 – Assessment Methodology for Fuel Cell Propulsion Integration
To what extent can the developed tool be applied to systematically analyse the aerodynamic performance and spatial integration of fuel cell subsystems in aircraft?

Two criteria must be met for systematic aerodynamic assessment of FC subsystem integration:

1. Ability to initialise representative geometries including internal subsystems accurately.
2. Ability to automatically run reliable aerodynamic analyses.

The tool partially fulfils the first criterion. Manual verification shows that a vast selection of complex WNI geometries including internal subsystem containers can be generated robustly, although automated and accurate initialisation (e.g., via CPACS or geometry import) is not yet implemented and would lead to accurate reference design generation.

The second criterion is not fulfilled. The aerodynamic solver interface does not presently capture inlet and outlet physics nor is mesh quality sufficiently controlled for reliable results. Additionally, there is no explicit coupling between subsystem sizing and external aerodynamic shaping. Despite these limitations, the newly introduced concepts, *parametrisation-as-composition*, *descriptors* and the `BlendSolid`, create a scalable framework for meaningful subsystem-geometry and full WNI interaction in future extensions.

RQ 4 – Parametrisation of Intake Configurations
Which geometrical parameters must be incorporated into the tool to enable research and into configurations of WNI systems?

The literature-derived parameters implemented in this work (e.g droop, lip radius, duct height, stagger, etc.) constitute the first structured parameter set tailored specifically to (un)conventional FC-relevant intakes. Since the aerodynamic discipline is still evolving for these configurations, the parameter set remains intentionally incomplete and flexible. Furthermore, this work includes many proven parametrisations for wing and nacelle. Through the *parametrisation-as-composition* concept, new parametrisations can be introduced without modifying the core architecture, ensuring continuous refinement as research progresses.

RQ 5 – Valid Geometry Generation
What level of geometric fidelity is required for the generated geometries to be considered usable for aerodynamic analysis and other downstream analyses or design tools?

The required level of fidelity is characterised by five geometric quality properties: *completeness*, *validity*, *manifoldness*, *simplicity*, and *accuracy*. These metrics are formally encoded into the verification workflow using automated geometric checks for self-intersections, free edges, sliver faces, and other mesh-relevant pathologies. Mesh convergence testing is used as an additional verification step to ensure compatibility with downstream aerodynamic tools.

Furthermore, parametric STEP export, including both primary OML surfaces and supporting reference geometry, guarantees usability in subsequent design environments. Therefore, the thesis formalises a concrete geometric fidelity threshold and provides built-in automated mechanisms to evaluate compliance for any generated configuration.

RQ 6 — Extendability of Parametrisations and Disciplines
How can the tool architecture be designed to remain extendable, both in terms of additional parametrisations and the inclusion of new disciplinary modules?

Extendability is inherently supported by the *parametrisation-as-composition* paradigm, which ensures that parametrisation choices remain interchangeable modules separate from the core geometry logic. Dependencies are managed by lightweight descriptor objects, enabling complex parametric interactions while maintaining modularity.

Disciplinary expansion is supported through a common `Simulation` interface, which defines standardised data handling and automation mechanisms for adding solver-based analyses. Although the current implementation covers only meshing and aerodynamic panel-solver implementations, the architectural foundation supports the integration of new disciplines like structures, thermal management, propeller aerodynamics, or MDO workflows without re-designing existing components.

RQ 7 — Knowledge and Understanding of Fuel Cell-powered Aircraft
To what extend can the tool contribute to systematic knowledge reuse and improved understanding of propulsion integration for FC-powered aircraft?

Knowledge reuse is enabled by encoding aerodynamic, geometric, and integration principles directly into parametrisations and geometrical integration modules. This ensures that design intent becomes shareable, traceable, and applicable across multiple use-cases. The tool structure promotes systematic identification of coupling effects through parameter studies and automated geometry generation of both the components individually and the various integration combinations.

Although quantitative aerodynamic insights are not yet fully supported, the tool already contributes a reusable knowledge base and offers a platform from which future research into aerodynamic integration for FC-powered configurations can progress effectively.

9.1.2. Evaluation of Hypotheses and Research Objective

Hypothesis 1 is directly tied to the primary purpose of the tool: its capability to generate representative external geometries. This is addressed by RQ 1 — Framework and Methodology, which confirms the feasibility of automated OML generation within the intended domain. Moreover, the condition *"...valid external geometries..."* is explicitly satisfied by RQ 5 — Valid Geometry Generation, as the tool consistently produces geometrically valid configurations, showing full robustness under manual design and an automatic parametric robustness of XX%. Therefore, Hypothesis 1 can be generally accepted, with the exception of its claim to *"...enable the identification of aerodynamically effective configurations..."* which is only qualitatively supported at present. While the implemented parametrisations allow exploration of aerodynamically attractive concepts, the limited aerodynamic validation constrains quantitative assessment, as summarised by RQ 2 — Identification of Aerodynamically Attractive Designs.

Hypothesis 2 concerns extendability and knowledge capture for WNI configurations. Since these aspects guided the architecture from the start, RQ 6 — Extendability of Parametrisations and Disciplines demonstrates that extendability is structurally ensured and already practically enabled. Additionally, the current comprehensive and rich parametrisation set and the knowledge-encoding nature of the modelling framework support the intended goal of *"...enhancing the understanding of intake design, engine integration, and FC aircraft technology."* This contribution is directly recognised in RQ 7 — Knowledge and Understanding of Fuel Cell-powered Aircraft, leading to full acceptance of Hypothesis 2.

In conclusion, the collective answering of the research questions results in broad acceptance of both hypotheses as Hypothesis 1 being accepted with noted limitations on aerodynamic performance evaluation, and Hypothesis 2 fully accepted. Consequently, the research objective has been successfully achieved: the *"Development of a first version of an extendable parametric tool capable of generating a vast selection of physically sensible and valid external geometries for wing-nacelle-intake integrations, representative of fuel cell-powered aircraft."*

Success of the research objective is also notable from the assessment of the system requirements in section E.3. Summarised in table E.1, it can be seen that most of the requirements are fulfilled while mapping notable area of improvements. The most notable limitations are observed in SR-2, where aerodynamic realism due to limited validation remain limited, and SR-6, where model initialization lacks

automated and accurate workflows. Furthermore, SR-5 (aerodynamic assessment fidelity) and SR-3 (parameter definition robustness) are identified as key areas for further improvement. Overall, the results indicate a mature and extensible design framework with clearly defined directions for future enhancement.

Addressing of Problem Statements

When revisiting the original problem statements, it must be acknowledged that they are not fully resolved by the current version of the tool. However, each problem is substantively addressed, and the tool establishes a critical foundation on which complete solutions can be built.

Firstly, the tool contributes directly to the redesign of propulsion integration for FC-powered aircraft. It embeds parametric modelling strategies explicitly aimed at innovative WNI integration beyond conventional turbofan architectures. This enables structured exploration and qualitative aerodynamic evaluation of configurations better suited to fuel cell propulsion, shifting away from the large turbofan geometries that historically constrained installation concepts. Nevertheless, the ability to consistently generate fully blended configurations, expected to provide the highest aerodynamic benefits, is not yet mature due to the current wing-nacelle blending limitations. Therefore, while the tool can already support improved conceptual design and analysis for FC propulsion integration, its full potential for novel integrated configurations has not yet been realized.

Secondly, as previously stated in RQ 3 — Assessment Methodology for Fuel Cell Propulsion Integration, the tool does not yet function as a generic aerodynamic and spatial integration assessment environment for fuel cell subsystems. Current constraints in aerodynamic solver fidelity, subsystem-geometry coupling, and automated geometry and design initialization limit its application in automatic comprehensive aerodynamic performance analysis. However, these limitations have been clearly identified, and the architecture is explicitly designed to enable future integration of these capabilities, positioning the tool as an expandable platform for further development in this direction.

Finally, the enabling of research into unconventional intake concepts is only partially achieved. A new methodological and parametrisation framework for scoop intakes and nacelle integration has been introduced, but several high-potential intake concepts identified in section 2.2.1 are not yet implemented. Despite this, the underlying architecture is highly extensible and well-suited to accommodate future intake developments by reuse of various concepts and components presented in this work. The tool therefore provides a promising research environment for advancing and analysing innovative intake concepts aligned with fuel cell propulsion integration employing all benefits from novel technology and techniques.

9.2. Recommendations

This work introduces a first version of a parametric tool capable of generating physically valid outer mold lines for WNI configurations representative of FC-powered aircraft. However, as concluded in the previous section, several limitations prevent full closure of the identified research gaps. These limitations affect algorithmic robustness, modelling flexibility, aerodynamic analysis capability, and overall usability.

To guide future developments and ongoing research within this domain, the following recommendations are grouped into four coherent research directions. Each group forms the basis for a future contribution, collectively aimed at improving both the capability and value of the tool.

9.2.1. Algorithms and Model Architecture

The current Gordon surface implementation in TiGL introduces a dominant bottleneck for robustness, accuracy, and surface continuity, as demonstrated (section 8.1.3, section 7.4, section 8.2.2, section 4.5.4). Error reporting is limited, intermediate surfaces require workarounds such as additional cross-sections, and the algorithm is prone to silent failures. A full re-implementation or replacement with a more feature-rich industrial lofting algorithm, supporting explicit continuity constraints, shape preservation, and improved numerical stability, is strongly recommended. Of which industry examples are: CATIA v5's

Loft[2] and Autodesk's Loft[3].

Similarly, further development of the `BlendSolid` algorithm is essential. While the current prototype demonstrates strong potential for general parametric blending within KBE systems, its quasi-tangency behavior, limited applicability, and limited intermediate shape maintenance require improvement. The current `BlendSolid`, shows high potential for a general parametric blending solution to be used in geometry focussed KBE applications, closing the gap between KBE and CADx in terms of geometric capabilities, especially when combined with an upgraded surface generation core as a newly improved Gordon surface. Example of similar feature-rich industry algorithm is CATIA v5's `Blend Surface`[4].

In addition, Free-Form Deformation (FFD) is recommended as a new modelling method, particularly for wing–nacelle integration, offering both improved smoothness and local shape control without sacrificing airfoil or nacelle fidelity. A generic, reusable FFD module could become a cornerstone technology for future blended configurations.

Finally, architectural improvements are advised: rigid *specialised* integration objects should be converted into fully *configurable parametric* objects, decoupling parametrisation logic from core geometry representation. This ensures better extendability and aligns all model components with the *parametrisation-as-composition* methodology.

9.2.2. Aerodynamics and Meshing

Achieving the second problem statement requires that the tool evolve into a fully automatic aerodynamic assessment environment for WNI integrations which further addresses Hypothesis 1 in terms of the direct identification of aerodynamically attractive WNI designs. For that reason, a custom *meshing module* is needed to ensure consistently high-quality meshes tailored to each geometrical entity. This module should operate at the component level (independent from parametrisations) and support both internal and external simulations.

Following meshing improvements, the aerodynamic solver integration must be expanded. FlightStream is a promising solver candidate as earlier validation results from section 8.2.3 suggest good potential, but full exposure of solver settings, inlet and outlet interfaces, and comprehensive validation studies are required to ensure accuracy and reliability. Since many solver settings rely on high-quality meshing, this step goes hand-in-hand with the meshing module extension.

The introduction of a *general automatic initialization module* is strongly recommended. Such an initialization capability is crucial for realistic aerodynamic evaluation and strongly supports solving the second problem statement in future versions. This module should support both:

- **Reference-geometry initialisation** for geometry validation, shape refitting, and transformation of static designs into parametrised form.
- **CPACS-based initialisation**, enabling the tool to serve as a general assessment platform for arbitrary WNI geometries.

9.2.3. Validation and Design Applications

The current validation is limited to a single wing–nacelle experiment without an intake, leaving the core design methodology of scoop geometries unvalidated. Thus it is recommended that additional geometrical validation cases, including full WNI configurations with scoop intakes, are performed to assess the accuracy of the parametrisations and integration routines. New design studies and use-cases emphasise where in what areas the tool is lacking in terms of core logic and currently implemented parametrisations and will thus prioritise key areas of improvement. The *Reference-geometry initialisation* from previous section highly increases the accuracy of these new studies. Expanded aerodynamic validation must be conducted once mesh and solver fidelity improve.

Meanwhile, the tool can already support design studies into integration effects and aerodynamic viability of FC-powered WNI configurations, within the currently presented scope, where identification of key design parameters and their effect are central. Due to STEP export, geometry refinement in

[2] `https://dshelp-embed.3ds.com/2025/english/CATIA_P3/online/CATIAfr_C2/icmugCATIAfrs.htm`
[3] `https://help.autodesk.com/view/fusion360/ENU/?guid=SFC-LOFT`
[4] `https://dshelp-embed.3ds.com/2025/english/CATIA_P3/online/CATIAfr_C2/icmugCATIAfrs.htm`

downstream environments is also enabled, promoting applicability to early-stage conceptual design workflows. For that reason, these design studies in combination with new geometry validation studies are highly recommanded to identify critical areas of modelling improvement.

9.2.4. Model Features and Component Extensions

Extending the tool's applicability and design space requires expanding its library of components and parametrisations. All should be constructed as reusable *configurable parametric* modules leveraging existing elements such as lips and cross-sections. Many promising WNI concepts identified in chapter 2 are not yet implemented, including:

- Leading-edge intakes
- Ring intake configurations
- Exit geometries

Furthermore, additional disciplinary models should be integrated into the `SystemAssembly`, including but not limited to:

- Structural analysis and load modelling
- Internal FC subsystem sizing, placement and coupling to geometry
- Performance and thermal–aerodynamic coupling of cooling systems

These extensions would elevate the tool from a geometry generator into a fully multidisciplinary KBE environment, enabling systematic assessment of both internal and external integration for FC-powered aircraft.

References

[1] F. Zera. *Universal Hydrogen first flight*. Photograph taken at Grant County International Airport, Moses Lake, WA, USA. Used with written permission from photographer via email (November, 2025). 2023. url: https://www.zeraphoto.com/.

[2] ATAG. *Facts & figures*. 2024. url: https://atag.org/facts-figures (visited on 11/29/2024).

[3] McKinsey & Company. *Hydrogen-powered aviation: A fact-based study of hydrogen technology, economics, and climate impact by 2050*. Tech. rep. Clean Sky 2 and Fuel Cells & Hydrogen 2 Joint Undertakings, 2020. doi: 10.2843/471510.

[4] C. Hughes and C. Gear. *FlyZero - Our Vision for Zero-Carbon Emission Air Travel*. Tech. rep. FZO-ALL-REP-0004. Aerospace Technology Institute, 2022. url: https://www.ati.org.uk/flyzero-reports/?search=Our+Vision+for+Zero-Carbon+Emission+Air+Travel+.

[5] P. J. Ansell. "Hydrogen-Electric Aircraft Technologies and Integration: Enabling an environmentally sustainable aviation future". In: *IEEE Electrification Magazine* 10.2 (2022), pp. 6–16. doi: 10.1109/MELE.2022.3165721.

[6] H. Hoheisel. "Aerodynamic aspects of engine-aircraft integration of transport aircraft". In: *Aerospace Science and Technology* 1.7 (1997), pp. 475–487. doi: https://doi.org/10.1016/S1270-9638(97)90009-2.

[7] D. L. Dagget, S. T. Brown, and R. T. Kawai. *Ultra-efficient Engine Diameter Study*. Tech. rep. 2003-212309. Seattle, WA: NASA, 2003. url: https://ntrs.nasa.gov/api/citations/20030061085/downloads/20030061085.pdf.

[8] J.L. Felder. "NASA Electric Propulsion System Studies". Presentation. 2015.

[9] H. D. Kim, A. T. Perry, and P. J. Ansell. "A Review of Distributed Electric Propulsion Concepts for Air Vehicle Technology". In: *2018 AIAA/IEEE Electric Aircraft Technologies Symposium (EATS)*. Cincinnati, OH: IEEE, 2018. doi: 10.2514/6.2018-4998.

[10] R. de Vries and R. Vos. "Aerodynamic Performance Benefits of Over-the-Wing Distributed Propulsion for Hybrid-Electric Transport Aircraft". In: *Journal of Aircraft* 60 (2023), pp. 1201–1218. doi: 10.2514/1.C036909.

[11] H. Maho. "Aerodynamics of wing-integrated ram-air duct for propeller aircrafts". MA thesis. Delft: TU Delft, 2025. url: https://resolver.tudelft.nl/uuid:39d98e6c-9e3b-4d4c-86f0-774fef917968.

[12] L.L.M. Veldhuis. "Propeller Wing Aerodynamic Interference". PhD thesis. Delft: TU Delft, 2005. url: https://resolver.tudelft.nl/uuid:8ffbde9c-b483-40de-90e0-97095202fbe3.

[13] S. Tiwari, M. J. Pekris, and J. J. Doherty. "A review of liquid hydrogen aircraft and propulsion technologies". In: *International Journal of Hydrogen Energy* 57 (2024), pp. 1174–1196. doi: https://doi.org/10.1016/j.ijhydene.2023.12.263.

[14] G. Inacio et al. *Feasibility Study of Using Liquid Hydrogen Tanks as Energy Carriers and Cooling Agents for a Small Aircraft Powered by PEMFCs*. SAE Technical Paper 2022-01-0009. SAE International, 2022. doi: 10.4271/2022-01-0009.

[15] T.L.C. Hoogerdijk. "Aircraft Integration of Air-Based Thermal Management Systems for Propulsive Fuel Cell Systems". MA thesis. Delft: TU Delft, 2023. url: https://repository.tudelft.nl/record/uuid:592ad30f-93b3-4d01-9fec-9786616dd9c4.

[16] A. Scoccimarro. "Preliminary design methods for the thermal management of fuel cell powered aeroengines". MA thesis. Delft: TU Delft, 2023. url: https://repository.tudelft.nl/record/uuid:9a16e0f8-722d-4a17-9f79-96cea2de6906.

[17] Kozulovic D. “Heat Release of Fuel Cell Powered Aircraft”. In: *Global Power and Propulsion Technical Conference Chania 2020* 10 (2020), pp. 767–775. doi: 10.33737/gpps20-tc-99.

[18] G.L.M. Vonhoff. “Conceptual Design of Hydrogen Fuel Cell Aircraft”. MA thesis. Delft: TU Delft, 2021. url: https://repository.tudelft.nl/record/uuid:8bd63dec-b67b-496b-92bc-3d5c07ff859f.

[19] C. K. Sain, J. Hänsel, and S. Kazula. “Preliminary Design of Air and Thermal Management of a Nacelle-Integrated Fuel Cell System for an Electric Regional Aircraft”. In: *2023 IEEE Transportation Electrification Conference & Expo (ITEC)*. Detroit, MI: IEEE, 2023, pp. 1–8. doi: 10.1109/ITEC55900.2023.10187105.

[20] D.J. Juschus. “Preliminary Propulsion System Sizing Methods for PEM Fuel Cell Aircraft”. MA thesis. Delft: TU Delft, 2021. url: https://repository.tudelft.nl/record/uuid:fdc14875-175e-4a4b-8cde-f2f4b6028194.

[21] M. Zupanič. “Conceptual Design of Fuel Cell Commuter Aircraft”. MA thesis. Delft: TU Delft, 2023. url: https://repository.tudelft.nl/record/uuid:d499f4bb-921a-4e3d-a676-614961104172.

[22] Aerodynamics Research Branch NACA Headquarters. *Bibiography of NACA and other rperots on air inlets and internal flows*. Tech. rep. NACA, 1948. url: https://ntrs.nasa.gov/citations/19930085495.

[23] A. Sóbester. “Tradeoffs in Jet Inlet Design: A Historical Perspective”. In: *Journal of Aircraft* 44.3 (2007), pp. 705–717. doi: 10.2514/1.26830.

[24] E. Regeling. *Literature Review. Preliminary Parametric Tool with a Focus on Geometry Generation for Intake–Propeller–Wing Systems in the Context of Hybrid–Electric Fuel Cell Aircraft*. Tech. rep. MSc thesis project deliverable. Delft: TU Delft, 2024.

[25] Inc. ZeroAvia. *ZeroAvia Successfully Completes Initial Dornier 228 Flight Test Campaign*. 2023. url: https://zeroavia.com/complete-flight-test/ (visited on 10/24/2024).

[26] Universal Hydrogen Co. *Product*. 2024. url: https://hydrogen.aero/product/ (visited on 10/24/2024).

[27] R. Elmendorp, R. Vos, and G. La Rocca. “A conceptual design and analysis method for conventional and unconventional airplanes”. In: *29th Congress of the International Council of the Aeronautical Sciences, ICAS 2014*. St. Petersburg: ICAS 2014, 2014.

[28] Division of Research Information. *Index of NACA Technical Publications: 1915-1949*. Tech. rep. NACA, 1949. url: https://ntrs.nasa.gov/citations/20160003570.

[29] A. Mota Mera. “Aerodynamic design for a pusher propeller spinner”. MA thesis. Delft: TU Delft, 2020. url: https://repository.tudelft.nl/record/uuid:08b51dbf-70fe-4839-82ef-fb3f813cea32.

[30] R.R. Prabhu. “Experimental investigation of the influence of spinner shape on fan performance in subsonic uniform and non-uniform flow”. MA thesis. Delft: TU Delft, 2021. url: https://repository.tudelft.nl/record/uuid:440faec2-fd6f-49a3-92a2-86fe3ef50924.

[31] G. La Rocca. “Knowledge Based Engineering”. In: *Multidisciplinary Design Optimization Supported by Knowledge Based Engineering*. West Sussex: John Wiley & Sons, Ltd, 2015. Chap. 9, pp. 208–257. isbn: 9781118897072.

[32] C.B Chapman and M. Pinfold. “Design engineering-a need to rethink the solution using knowledge based engineering”. In: *Knowledge-Based Systems* 12.5 (1999), pp. 257–267. doi: https://doi.org/10.1016/S0950-7051(99)00013-1.

[33] G. La Rocca. “Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design”. In: *Advanced Engineering Informatics* 26.2 (2012), pp. 159–179. doi: https://doi.org/10.1016/j.aei.2012.02.002.

[34] W.J.C Verhagen et al. “A critical review of Knowledge-Based Engineering: An identification of research challenges”. In: *Advanced Engineering Informatics* 26.1 (2012), pp. 5–15. doi: https://doi.org/10.1016/j.aei.2011.06.004.

[35] J. Esanakula, N. V. Sridhar, and V. Rangadu. “Knowledge Based Engineering: Notion, Approaches and Future Trends”. In: *American Journal of Intelligent Systems* 5 (2015), pp. 1–17. doi: 10.5923/j.ajis.20150501.01.

[36] D. Cooper and G. La Rocca. “Knowledge-based Techniques for Developing Engineering Applications in the 21st Century”. In: *7th AIAA Aviation Technology, Integration and Operations Conference*. Reston, VA: AIAA, 2007, pp. 1–22. doi: 10.2514/6.2007-7711.

[37] A. Ortner-Pichler and C. Landschützer. “Integration of parametric modelling in web-based knowledge-based engineering applications”. In: *Advanced Engineering Informatics* 51 (2022), p. 101492. doi: https://doi.org/10.1016/j.aei.2021.101492.

[38] ParaPy B.V. *ParaPy*. 2024. url: https://parapy.nl/ (visited on 11/29/2024).

[39] M. Stokes. *Managing Engineering Knowledge MOKA: Methodology for Knowledge Based Engineering Applications*. London: Professional Engineering Publishing, 2001. isbn: 9781860582950.

[40] A. R. Kulkarni et al. “An MBSE approach to support Knowledge Based Engineering application development”. In: *Aerospace Europe Conference 2023 – 10□□ EUCASS – 9□□ CEAS Number of pages16*. Lausanne: EUCASS association, 2023.

[41] F. M. Mendes. “A Model-Based Systems Engineering Framework for developing Knowledge Based Engineering Applications”. MA thesis. Delft: TU Delft, 2023. url: https://repository.tudelft.nl/record/uuid:c9b8e9ad-3410-4d93-8a4c-b09b3313f0ad.

[42] A. Aleksandravičienė and A. Morkevičius. *MagicGrid Book of Knowledge: A Practical Guide to Systems Modeling Using MagicGrid from Dassault Systèmes*. 2nd ed. Kaunas: Vitae Litera, 2021. isbn: 9786094545542.

[43] E. Gamma et al. “Strategy: Object Behavioral”. In: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994, pp. 315–323. isbn: 9780201633610.

[44] E. Gamma et al. “Abstract Factory: Object Creational”. In: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994, pp. 87–95. isbn: 9780201633610.

[45] G. La Rocca. “Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization”. PhD thesis. Delft: TU Delft, 2011. url: https://resolver.tudelft.nl/uuid:45ed17b3-4743-4adc-bd65-65dd203e4a09.

[46] E. Obert. *Aerodynamic Design of Transport Aircraft*. Amsterdam: IOS press, 2009. isbn: 9781607504078.

[47] D. P. Raymer. *Aircraft Design: A Conceptual Approach*. 6th ed. Reston, VA: American Institute of Aeronautics and Astronautics, 2018. isbn: 9781624104909.

[48] J. Roskam. *Airplane Design. Pt. 1. Preliminary sizing of airplanes*. Ottawa, KS: Roskam Aviation and Engineering, 1985.

[49] E. Torenbeek and H Wittenberg. *Synthesis of Subsonic Airplane Design*. Delft: Delft University Press, 1982. isbn: 9789024727247.

[50] D. Küchemann and J. Weber. *Aerodynamics of Propulsion*. New York: McGraw-Hill, 1953.

[51] J. Shedon and E. L. Goldsmith. *Intake Aerodynamics*. 2nd ed. Oxford: Blackwell Science, 1999. isbn: 9780632049639.

[52] DARcorparation. *FlightStream®*. 2024. url: https://www.darcorp.com/flightstream-aerodynamics-software/ (visited on 11/10/2024).

[53] E. D. Olson and C. W. Albertson. “Aircraft High-Lift Aerodynamic Analysis Using a Surface-Vorticity Solver”. In: *54th AIAA Aerospace Sciences Meeting*. San Diego, CA: AIAA, 2016. doi: 10.2514/6.2016-0779.

[54] L. King, R. Hartfield, and V. Ahuja. “Aerodynamic Optimization of Integrated Wing-Engine Geometry using an Unstructured Vorticity Solver”. In: *33rd AIAA Applied Aerodynamics Conference*. Dallas, TX: AIAA, 2015. doi: 10.2514/6.2015-2880.

[55] V. Ahuja and R. Hartfield. “Predicting the Aero Loads Behind a Propeller in the Presence of a Wing Using Flightstream”. In: *15th AIAA Aviation Technology, Integration, and Operations Conference*. Dallas, TX: AIAA, 2015. doi: 10.2514/6.2015-2734.

[56] J. H. Koning. “Development of a KBE application to support aerodynamic design and analysis”. MA thesis. Delft: TU Delft, 2010. url: https://resolver.tudelft.nl/uuid:e7d206a5-2e1f-4287-a8b6-27258316bf79.

[57] T. Van den Berg. “Parametric modeling and aerodynamic analysis of multi-element wing configurations”. MA thesis. Delft: TU Delft, 2009.

[58] N. L. M. Luijk. “Constrained Aerodynamic Shape Optimisation of the Flying V Outer Wing”. MA thesis. Delft: TU Delft, 2023. url: https://resolver.tudelft.nl/uuid:fc2bbe10-6796-4337-81cc-5971b324d50e.

[59] M. Hillen. “Parametrisation of the Flying-V Outer Mould Line”. MA thesis. Delft: TU Delft, 2020. url: https://resolver.tudelft.nl/uuid:f4863ae4-2792-4335-b929-ff9dfdb6fed5.

[60] M. Siggel et al. “TiGL: An Open Source Computational Geometry Library for Parametric Aircraft Design”. In: *Mathematics in Computer Science*. Vol. 13. Springer Nature, 2019, pp. 367–389. doi: https://doi.org/10.1007/s11786-019-00401-y.

[61] M. Siggel et al. *The TiGL Geometry Library and its Current Mathematical Challenges*. 2017. url: https://elib.dlr.de/117498/1/TiGL-Uni-Koeln.pdf.

[62] A. Sóbester and A. I. J. Forrester. *Aircraft Aerodynamic Design - Geometry and Optimization*. Southampton: John Wiley & Sons, 2015. isbn: 9781523123407.

[63] T. W. Sederberg and S. R. Parry. “Free-form deformation of solid geometric models”. In: *ACM SIGGRAPH Computer Graphics* 20 (1986), pp. 151–160. doi: 10.1145/15886.15903.

[64] B. M. Kulfan. ““Fundamental” parametric geometry representations for aircraft component shapes”. In: *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. Portsmouth, VA: AIAA, 2006, pp. 1–45.

[65] C. A. E. Heimans. “Aerodynamic Analysis of Engine Integration during the Preliminary Design Phase”. MA thesis. Delft: TU Delft, 2021. url: https://resolver.tudelft.nl/uuid:a3e588a9-9d21-4170-a973-c53eb3b45afc.

[66] P. Proesmans. “Preliminary Propulsion System Design and Integration for a Box-Wing Aircraft Configuration”. MA thesis. Delft: TU Delft, 2019. url: https://resolver.tudelft.nl/uuid:0d2ebc46-09ee-493f-bb4c-c871133bff6f.

[67] M. M. Otting. “A nacelle design method for turbofan engines”. MA thesis. Delft: TU Delft, 2020. url: https://resolver.tudelft.nl/uuid:d24d8bfc-0b07-407a-b767-e834ff63a96e.

[68] S. S. Tambe et al. “54th 3AF International Conference AERO2019”. In: *Coherent vortex structures over a rotating spinner under non-axial inflows at low Reynolds number*. FP45-AERO2019-Tambe. Paris, 2019.

[69] D. Biermann and E.P. Hartman. *Tests of Five Full-Scale Propellers in the Presence of a Radial and a Liquid-Cooled Engine Nacelle, Including Tests of Two Spinners*. Tech. rep. NACA, 1938. url: https://ntrs.nasa.gov/citations/19930091719.

[70] Simulargе Inc. *Preparing CAD Geometry for CFD Simulations: Essential Steps and Best Practices*. 2024. url: https://www.simulargе.com/blog/preparing-cad-geometry-for-cfd-simulations-essential-steps-and-best-practices (visited on 11/29/2024).

[71] M. Mckay, R. Beckman, and W. Conover. “A Comparison of Three Methods for Selecting Vales of Input Variables in the Analysis of Output From a Computer Code”. In: *Technometrics* 21 (1979), pp. 239–245. doi: 10.1080/00401706.1979.10489755.

[72] R. R. Duivenvoorden et al. “Experimental Investigation of Aerodynamic Interactions of a Wing with Deployed Fowler Flap under Influence of a Propeller Slipstream”. In: *AIAA AVIATION 2022 Forum*. AIAA 2022-3216. Chicago: American Institute of Aeronautics and Astronautics (AIAA), 2022. doi: 10.2514/6.2022-3216.

[73] R. R. Duivenvoorden. “Aerodynamic Phenomena of Propeller-Wing-Flap Interaction”. PhD thesis. Delft: TU Delft, 2025. doi: https://doi.org/10.4233/uuid:1a705c0a-b0aa-4ddd-a3508c663cf2fdb0.

[74] R. R. Duivenvoorden et al. *TU Delft Propeller-Wing-Flap (TUD-PWF) Wind Tunnel Model Geometry*. 2025. doi: 10.5281/zenodo.13344026.

[75] N. Van Arnhem et al. *TUD-XPROP-S propeller geometry*. 2022. doi: 10.5281/zenodo.6355670.

[76] Altair Engineering Inc. *FlightStream Theory Manual*. Online documentation. Altair Engineering Inc. 2024. url: https://flightstream-theory.altair.com/.

[77] S. F. Racisz. *Development of wing inlets*. Tech. rep. NACA, 1946. url: https://ntrs.nasa.gov/citations/19930092761.

[78] Aeronautics-Guid. *Aircraft Turbine Engine Inlet Systems*. 2024. url: https://www.aircraftsystemstech.com/p/turbine-engine-inlet-systems-engine.html?utm_content=cmp-true (visited on 11/10/2024).

[79] A. Dubois et al. "Design of an Electric Propulsion System for SCEPTOR's Outboard Nacelle". In: *16th AIAA Aviation Technology, Integration, and Operations Conference*. Washington, DC: AIAA, 2016. doi: 10.2514/6.2016-3925.

[80] W.J. Nelson. *Wind Tunnel Investigation of Rear Underslung Fuselage Ducts*. Tech. rep. NACA, 1943. url: https://ntrs.nasa.gov/citations/19930092658.

[81] S. J. Miley. "Aerodynamics of Liquid-Cooled Aircraft Engine Installations". In: *SAE Transactions* 94 (1985), pp. 698–707. doi: https://doi.org/10.4271/850896.

[82] D. Alex. *North American YF-93*. 2016. url: https://www.militaryfactory.com/aircraft/detail.php?aircraft_id=1308 (visited on 11/19/2024).

[83] R.W. Gilbey. *Drag and pressure recovery characteristics of auxiliary air inlets at subsonic speeds*. ESDU 86002. Engineering Sciences Data Unit, 1986.

[84] Mike Pratt. *Introduction to ISO 10303 - The STEP Standard for Product Data Exchange*. Tech. rep. Gaithersburg: Journal of Computing and Information Science in Engineering, 2000. url: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=821426.

[85] B. Nagel et al. "Communication in Aircraft Design: Can we establish a Common Language?" In: *28th congress of the International Council of the Aeronautical Sciences*. Brisbane: International Council of the Aeronautical Sciences (ICAS), 2012, pp. 1–13. isbn: 9780956533319.

A

Fuel Cell technological evolution

An overview of various fuel cell technology predictions from a literature review on liquid hydrogen aircraft technology research can be found in table A.1. FC-powered flight is highly dependent on the technology readiness level of FC stacks and Balance-of-Plant (BoP), while predictions show that sufficiently high power densities and efficiencies are possible [13].

Table A.1: Fuel Cell Technological Evolution predictions from the last 20 years.[1]

Source, publication year		Year	Type	System SP $[kW/kg]$	Stack PD $[kW/l]$	η	BoP	State
NASA	2003	2003	LTPEM	0.5	-	-	-	Estimate
NASA	2004	2004	LTPEM	0.3	-	-	-	Estimate
NASA	2009	2009	LTPEM	0.81	-	-	37%	Estimate
Antares DLR-H2	2009	2009	LTPEM	0.55	-	52%	-	Flown
Renau et al	2016	2016	HTPEM	0.18	-	-	-	Estimate
Westenberger	2016	2016	LTPEM	0.6	-	-	-	Estimate
Kadyk et al	2018	2018	-	1.6	-	-	20%	Estimate
NASA (Patent)	2018	2018	SOFC	1.5	7.5	-	-	Estimate
FCH and Clean Sky 2	2020	2020	-	0.75	-	-	-	Estimate
Unifier19	2020	2020	-	0.8	-	-	-	Estimate
Collins and McLarty	2020	2020	SOFC	1.128	-	-	-	Estimate
HyFlyer I	2020	2020	LTPEM	0.59	-	52%	-	Flown
Vonhoff	2021	2021	LTPEM	0.8	-	50%	-	Estimate
HyPoint	2021	2021	HTPEM	1.8	-	45%	40%	Estimate
AIA	2022	2021	LTPEM	1.5	-	-	60%	Estimate
HyPoint	2021	2021	LTPEM	0.75	-	47%	75%	Flown
Unifier19	2020	2025	-	2.13	-	-	-	Projection
HyPoint	2021	2025	HTPEM	3	-	48%	45%	Projection
FlyZero	2022	2026	LTPEM	2	-	60%	71%	Projection
Unifier19	2020	2030	-	3.46	-	-	-	Projection
FlyZero	2022	2030	HTPEM	2	-	-	33%	Projection
FlyZero	2022	2030	LTPEM	2.5	-	65%	72%	Projection
Kadyk et al	2018	-	-	8	-	-	-	Projection
Unifier19	2020	2035	-	4.8	-	-	-	Projection
FlyZero	2022	2035	HTPEM	3	-	-	40%	Projection
NASA	2009	2035	LTPEM	1.7	-	-	37%	Projection
FlyZero	2022	2035	LTPEM	3	-	70%	73%	Projection

[1] References to all projects can be found at [13].

Table A.1 (continued): Fuel Cell Technological Evolution predictions from the last 20 years (Automotive sector).

Source, publication year		**Year**	**Type**	**System SP** $[kW/kg]$	**Stack PD** $[kW/l]$	η	**BoP**	**State**
Toyota	2015	2002	LTPEM	0.28	1.1	-	-	SOA
Toyota	2015	2008	LTPEM	0.36	1.4	-	-	SOA
General Motors	2007	2007	LTPEM	0.4	-	-	-	SOA
Toyota,	2015	2015	LTPEM	0.8	3.1	-	-	SOA
Hyundai NEXO	2018	2018	LTPEM	-	3.1	-	-	SOA
Toyota	2021	2021	LTPEM	1	5.4	-	-	SOA
Honda	2021	2021	LTPEM	0.8	3.1	-	-	SOA
PowerCell	2021	2021	LTPEM	1.2	3.3	-	-	SOA
ElringKlinger	2022	2022	LTPEM	-	4	-	-	SOA
Horizon Tech	2022	2022	LTPEM	0.77	4.7	60%	75%	SOA

B

Preliminary Parametrisations and Visual Aids for the Intake Configurations from the Literature Review

This appendix presents the preliminary parametrisations and representative images for all intake configurations discussed in chapter 2. The intent is to provide a consolidated visual reference for future model extensions, rather than to restate the full aerodynamic rationale behind each parameter. A complete, detailed discussion of all parameters, their aerodynamic implications, and corresponding literature sources is provided in the literature review report [24].

B.1. General Intake Parametrisation

Figure B.1 shows the preliminary parametrisation used for a conventional podded duct, with the corresponding geometric variables listed in table B.1. These parameters reflect general intake characteristics and form the basis from which type-specific parametrisations are derived.

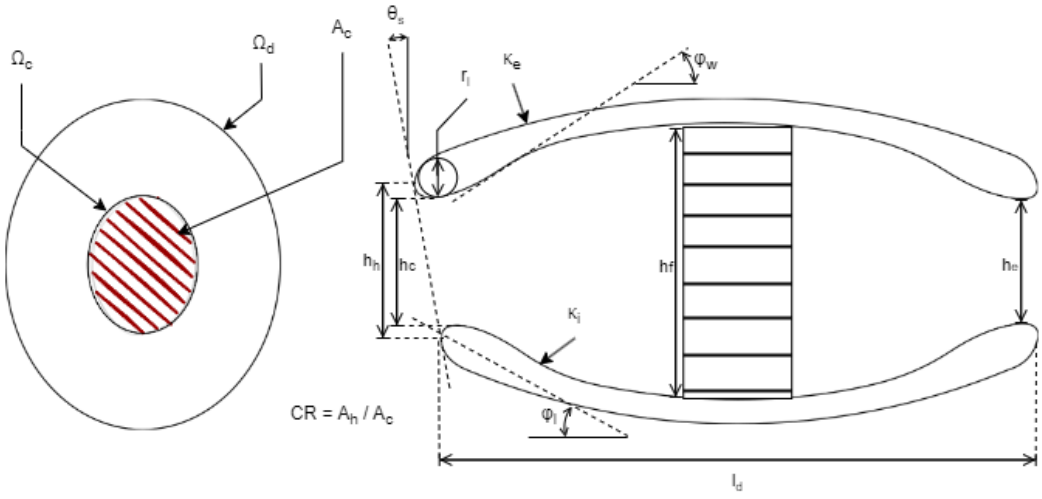


Figure B.1: Schematic front view and internal geometry of a podded duct and radiator with a preliminary parametrisation.

Table B.1: Initial design variables for intakes in general

Geometry Parameter	Symbol	Remark
Exit area	A_e	Used for flow matching
Intake area	A_c	Determines entry flow speed
Radiator area	A_f	Critical in cooling performance
Highlight area	A_h	Effects lip radii
Lip droop	φ_l	For both upper and lower lip
Stagger	θ_s	-
Contraction ratio	CR	Similar to lip radius
Wall angle	φ_w	For both upper and lower internal wall
External curvature	κ_e	For both upper and lower surface
Internal curvature	κ_i	Both for upper and lower surface
Cross-sectional shape intake	Ω_c	Can be complex involving many parameters
Cross-sectional shape duct	Ω_d	Can be complex involving many parameters
Lip radius	r_l	-
Cowl/duct length	l_d	Upper and lower are different due stagger

B.2. Leading-edge Intake

Figure B.2 and figure B.3 illustrate the geometry and preliminary parametrisation of a leading-edge intake. The associated design variables are listed in table B.2. These visualisations support the summary of aerodynamic principles presented in the literature review.

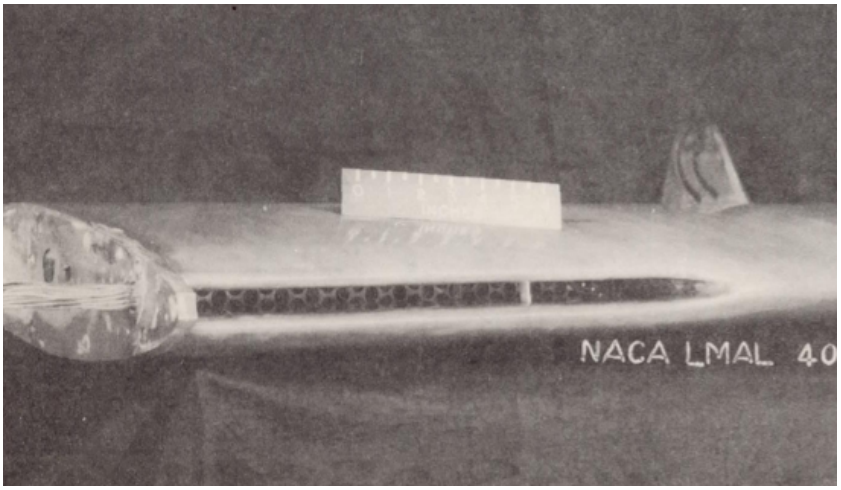


Figure B.2: Three-quarter view of a LE model, inverted showing fairing between ducted and plain airfoil sections. [77]

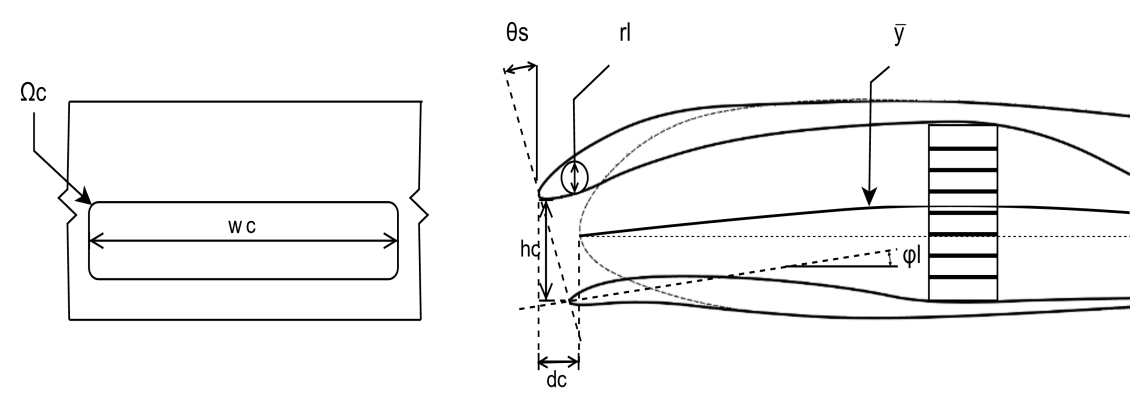


Figure B.3: Schematic front view and internal geometry of a LE intake and radiator with a preliminary parametrisation, on top of the original airfoil of the wing.

Table B.2: Initial design variables for Leading-Edge intakes.

Geometry Parameter	Symbol	Remark
Intake height	h_c	-
Intake width	w_c	-
Stagger	θ_s	-
Cant angle horizontal	θ_c	Relevant for side-slip and propeller slipstream
Camber	$\overline{y}$	Function of x: $\overline{y}(x)$
Lip droop	φ_l	For both upper and lower lip
Lip radius	r_l	For upper and lower lip
Cross-sectional shape intake	Ω_c	Can be complex involving many parameters
Forward shift intake plane	d_c	Increases chord of airfoil locally

B.3. Scoop Intake

Figures figure B.4 and figure B.5 provide a representative example and preliminary parametrisation of a scoop intake. The complete set of geometric variables used for studies and future model development is listed in table B.3.

Figure B.4: Example of a scoop intake on a turboprop engine. [78]

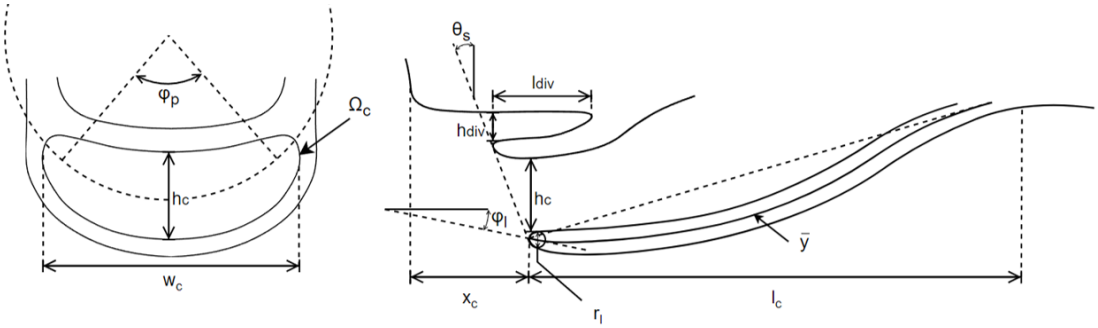


Figure B.5: Schematic front view and internal geometry of a scoop intake with a preliminary parametrisation.

Table B.3: Initial design variables for scoop intakes.

Geometry Parameter	Symbol	Remark
Intake length	l_c	-
Intake height	h_c	-
Intake width	w_c	Can be computed result from intake area and height
Lip radius	r_l	For upper and lower lip
Lip droop	φ_l	For both upper and lower lip
Stagger	θ_s	-
Diverter height	h_{div}	-
Diverter length	l_{div}	Has effect on internal curvature
Intake axial location	x_c	Influences l_c
Camber	$\overline{y}$	Representation of external curvature
Cross-sectional shape intake	Ω_c	Can be complex involving many parameters
Peripheral extend	φ_p	Different representation of width

B.4. Ring Intake

A representative ring intake and its associated preliminary parametrisation are shown in figure B.6 and figure B.7. Table B.4 summarises the corresponding design variables used for aerodynamic interpretation and future tool expansion.

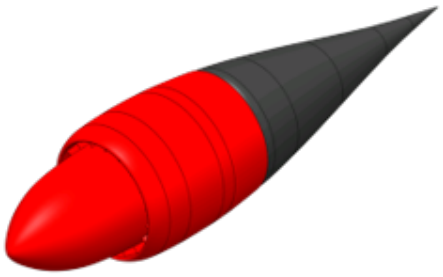


Figure B.6: Full nacelle with ring intake for cooling of a fully electrical propulsion system. [79]

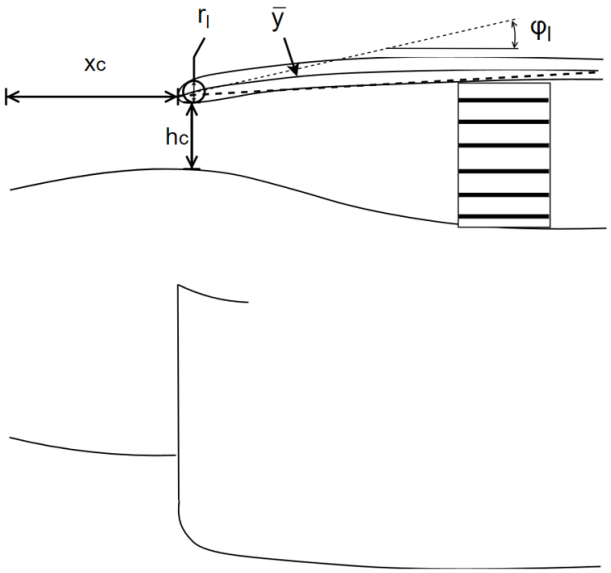


Figure B.7: Schematic side internal view of a ring intake with a preliminary parametrisation.

Table B.4: Initial design variables for ring intakes.

Geometry Parameter	Symbol	Remark
Intake height	h_c	-
Intake axial location	x_c	Influences l_c, or length nacelle
Lip radius	r_l	For upper and lower lip
Lip droop	φ_l	For both upper and lower lip
Intake axial location	x_c	Influences l_c
Camber	$\overline{y}$	Representation of external curvature

B.5. Underschlung Intake

Figure B.8 and figure B.9 show a rear-fuselage underslung intake together with a simplified geometric sketch. Table B.5 lists the parameters relevant for this intake type.

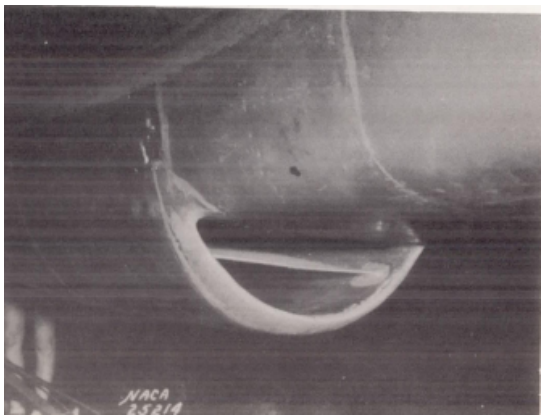


Figure B.8: View of a rear underslung fuselage duct with internal BL diverter [80]

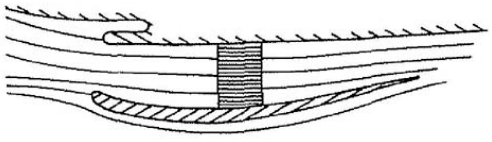


Figure B.9: Drawing of an underslung intake with external BL diverter [81]

Table B.5: Initial design variables for underslung intakes.

Geometry Parameter	Symbol	Remark
Intake length	l_c	-
Intake height	h_c	-
Intake width	w_c	Can be computed result from intake area and height
Lip radius	r_l	For upper and lower lip
Lip droop	φ_l	For both upper and lower lip
Stagger	θ_s	-
Diverter height	h_d	-
Diverter length	l_d	Has effect on internal curvature
Intake axial location	x_c	Influences l_c
Camber	$\overline{y}$	Representation of external curvature
Cross-sectional shape intake	Ω_c	Can be complex involving many parameters
Peripheral extend	φ_p	Different representation of width

B.6. Flush Intake

Representative figures for flush and NACA-type intakes are shown in figure B.10 and figure B.11. These images illustrate the geometric characteristics referenced in the literature review, even though flush intakes are not parametrised further in this thesis.

Figure B.10: A NACA intake used as primary intake on the YF-93. [82]

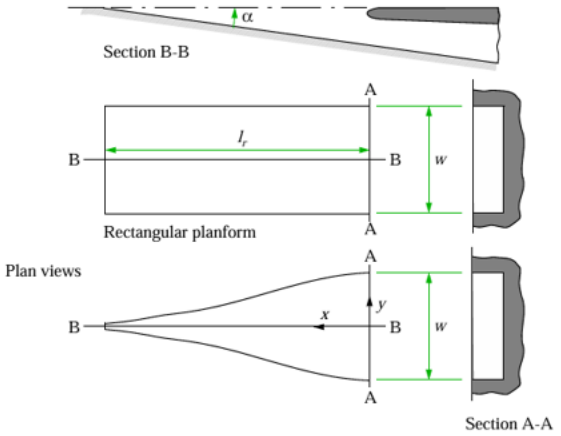


Figure B.11: Plan view of flush intakes including the NACA intake. [83]

C

STEP Export Algorithm

This appendix includes a detailed description of the newly implemented STEP exporter algorithm from section 4.3.2. In OCC, an full object hierarchy can be created using compounds. The algorithm begins by creating a single *root compound* corresponding to the top-level, or root, object in the model tree. It then calls a recursive *walk_object()* function in order to walk through the entire model. It performs two main operations for each object:

1. **Shape processing:** all shapes belonging to the object, both helper and root geometries, are processed (labels, colours, etc.) and added to the current compound.
2. **Child processing:** each child object in the hierarchy is handled in turn, invoking the same walk function recursively to construct and attach its own compound to the parent.

Once all shapes and child components have been processed, the algorithm returns the fully assembled root compound. This compound is subsequently written to a STEP file using the OCC `STEPCAFControl_Writer`, preserving colours, labels, and attribute names.

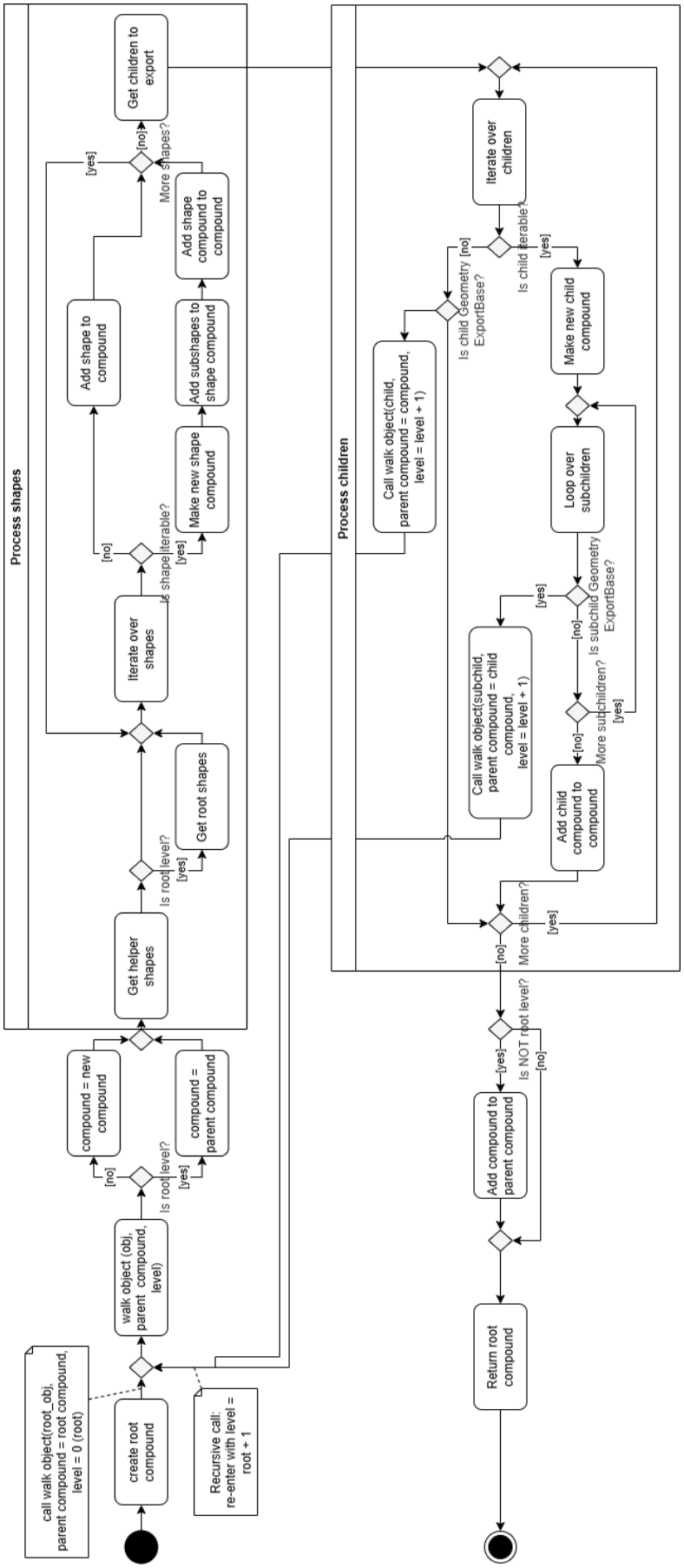


Figure C.1: Flow diagram of the compound-construction algorithm for collecting all geometry and children in the model tree during STEP export.

D

Full Parameter Specifications for Robustness Study

This appendix documents the complete set of parameter specifications employed in the parametric robustness study introduced in section 8.1.3. Together, these specifications provide an explicit definition of the bounded, physically admissible design space $\mathcal{X}_{\text{phys}}$ used for sampling.

Table D.1 presents an overview of all sampled input parameters, grouped by subsystem. Each parameter is assigned a *type* that directly corresponds to the parameter specification classes defined in section 8.1.3 under Design Space Sampling: *continuous* (`Continuous`), *relative* (`Relative to <slot>`), *sequence* (`Sequence of <length>`), or a combination of *relative* and *sequence*. For each parameter, a bounded sampling interval is specified, which may be defined either in absolute terms or relative to another slot value.

In addition, inter-parameter dependencies are introduced to enforce physically realistic configurations during sampling. These dependencies restrict the admissible range of certain parameters based on the values of others and are expressed using simple constraint relations: *equal to* ($= \texttt{other parameter}$), *less than or equal to* ($\leq \texttt{other parameter}$), or *greater than or equal to* ($\geq \texttt{other parameter}$). Collectively, these constraints ensure that all sampled designs remain within a realistic and consistent region of the design space, which is essential for a meaningful robustness assessment.

Table D.1: Parametric robustness sampling design space overview.

Name	Type	Bounds	Dependency
Wing			
AR	Continuous	$[6,\ 14]$	–
Span	Continuous	$[10,\ 18]$	–
Taper	Continuous	$[0.2,\ 1.0]$	–
Sweep	Continuous	$[0,\ 20]$	–
Twist	Sequence of 3	$[-7,\ 2]$	–
Dihedral	Sequence of 3	$[-0.2,\ 0.35]$	–
Airfoils			
Airfoil 1 shape coef. (upper)	Sequence of 5	$[0.1,\ 0.5]$	–
Airfoil 1 shape coef. (lower)	Sequence of 5	$[-0.5,\ -0.1]$	–
Airfoil 2 shape coef. (upper)	Sequence of 5	$[0.1,\ 0.5]$	–
Airfoil 2 shape coef. (lower)	Sequence of 5	$[-0.5,\ -0.1]$	–
Airfoil 3 shape coef. (upper)	Sequence of 5	$[0.1,\ 0.5]$	–
Airfoil 3 shape coef. (lower)	Sequence of 5	$[-0.5,\ -0.1]$	–
Airfoil 4 shape coef. (upper)	Sequence of 5	$[0.1,\ 0.5]$	–
Airfoil 4 shape coef. (lower)	Sequence of 5	$[-0.5,\ -0.1]$	–
Airfoil 5 shape coef. (upper)	Sequence of 5	$[0.1,\ 0.5]$	–
Airfoil 5 shape coef. (lower)	Sequence of 5	$[-0.5,\ -0.1]$	–
Airfoil 2 chord ratio	Continuous	$[0,\ 1]$	–
Airfoil 3 chord ratio	Continuous	$[0,\ 1]$	–
Airfoil 4 chord ratio	Continuous	$[0,\ 1]$	–

Continued on next page

Name	Type	Bounds	Dependency
Airfoil 2 cant angle	Continuous	$[-20, 20]$	–
Airfoil 3 cant	Continuous	$[-20, 20]$	–
Airfoil 4 cant	Continuous	$[-20, 20]$	–
Airfoil 1 dihedral angle	Continuous	$[-30, 30]$	–
Airfoil 2 dihedral angle	Continuous	$[-30, 30]$	–
Airfoil 3 dihedral angle	Continuous	$[-30, 30]$	–
Airfoil 4 dihedral angle	Continuous	$[-30, 30]$	–
Airfoil 5 dihedral angle	Continuous	$[-30, 30]$	–
Nacelle			
Length	Relative to *MAC*	$[0.75, 1.5]$	–
Spanwise location	Continuous	$[0.1, 0.9]$	–
Chordwise location	Continuous	$[0.1, 0.5]$	–
Offsets	Relative to *MAC* sequence of 3	$[-0.1, 0.1]$	–
Droop angle	Continuous	$[-10, 20]$	–
Nacelle cross-sections			
Cross-section 1 class factor (upper)	Continuous	$[0.1, 0.5]$	–
Cross-section 1 class factor (lower)	Continuous	$[0.1, 0.5]$	–
Cross-section 1 width	Relative to *Nacelle length*	$[0.1, 0.3]$	–
Cross-section 1 cant angle	Continuous	$[-20, 20]$	–
Cross-section 1 dihedral angle	Continuous	$[-20, 20]$	–
Cross-section 2 class factor (upper)	Continuous	$[0.1, 0.5]$	–
Cross-section 2 class factor (lower)	Continuous	$[0.1, 0.5]$	–
Cross-section 2 width	Continuous	$[0.1, 0.3]$	$=$ *Cross-section 1 width*
Cross-section 2 cant angle	Continuous	$[-20, 20]$	–
Cross-section 2 dihedral angle	Continuous	$[-20, 20]$	–
Cross-section 3 class factor (upper)	Continuous	$[0.1, 0.5]$	$=$ *Cross-section 2 class factor (upper)*
Cross-section 3 class factor (lower)	Continuous	$[0.1, 0.5]$	$=$ *Cross-section 2 class factor (lower)*
Cross-section 3 width	Continuous	$[0.1, 0.3]$	$=$ *Cross-section 1 width*
Cross-section 3 cant angle	Continuous	$[-20, 20]$	–
Cross-section 3 dihedral angle	Continuous	$[-20, 20]$	–
Spinner			
Radius	Relative to *Cross-section 3 width*	$[0.25, 0.75]$	–
Aspect ratio	Continuous	$[0.5, 4]$	–
Scoop			
Location	Continuous	$[0.5, 0.9]$	–
Diverter height	Relative to *Nacelle length*	$[0.005, 0.03]$	–
Stagger	Continuous	$[-20, 40]$	–
Height	Relative to *Nacelle length*	$[0.075, 0.15]$	–
Length lower lip	Relative to *Nacelle length*	$[0.04, 0.12]$	–
Upper lip droop	Continuous	$[-10, 20]$	–
Lower lip droop	Continuous	$[-10, 30]$	–
Upper lip radius	Relative to *Nacelle length*	$[0.001, 0.004]$	–
Lower lip radius	Relative to *Nacelle length*	$[0.001, 0.004]$	–
Outer lower lip thickness	Relative to *Length lower lip*	$[0.1, 0.3]$	–
Inner lower lip thickness	Relative to *Length lower lip*	$[0.1, 0.3]$	–
Outer upper lip thickness	Relative to *Length lower lip*	$[0.1, 0.3]$	–
Inner upper lip thickness	Relative to *Length lower lip*	$[0.1, 0.3]$	–
Lower lip shape coef. (outer)	Sequence of 3	$[0.05, 0.4]$	–
Lower lip shape coef. (inner)	Sequence of 3	$[0.05, 0.4]$	–
Upper lip shape coef. (outer)	Sequence of 3	$[0.05, 0.4]$	–
Upper lip shape coef. (inner)	Sequence of 3	$[0.05, 0.4]$	–
Frontal area class factor (upper)	Continuous	$[0.1, 0.5]$	–
Frontal area class factor (lower)	Continuous	$[0.1, 0.5]$	–
Frontal area shape coef. (upper)	Sequence of 2	$[0.3, 1.0]$	–
Frontal area shape coef. (lower)	Sequence of 2	$[0.3, 1.0]$	–
Wing-nacelle matching			
Blend width	Continuous	$[0.05, 0.3]$	$\leq$ *Nacelle spanwise location* $- 0.05$
Nacelle cut location	Continuous	$[0.1, 0.8]$	$\leq$ *Scoop location* $- 0.15$
Chord position	Continuous	$[0.1, 0.9]$	$\leq$ *Nacelle chordwise location*
Nacelle-scoop integration			
Reattachment location	Continuous	$[0.3, 0.6]$	$\leq$ *Scoop location* $- 0.25$
σ_{blend} front	Continuous	$[0.5, 3.0]$	–
σ_{blend} sides	Continuous	$[0.5, 2.0]$	–
Wing-nacelle integration			
Fillet radius	Relative to *Nacelle length*	$[0.02, 0.2]$	$\leq$ *NS - Reattachment location* $- 0.05$

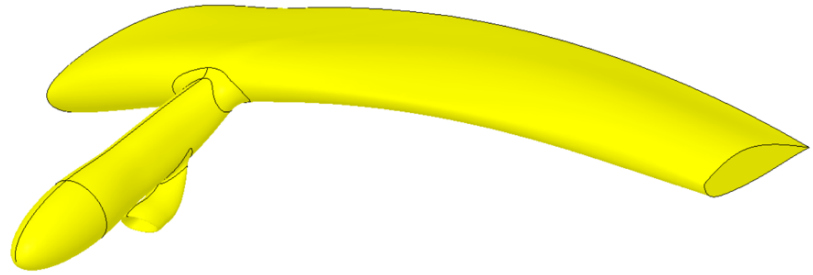
Continued on next page

Table D.1 lists all input parameters used as sample specification, organised by subsystem. For detailed definitions of the individual parameters and their physical interpretation, the reader is referred to the modelling chapters in the main body of the thesis (chapters 4 to 7). Several deliberate choices were made in defining the input parameter set and its associated bounds:

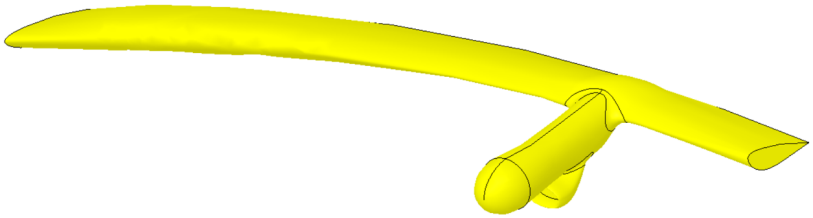
- The root (airfoil 1) and tip (airfoil 5) sections do not include cant angle or chord ratio parameters, as their chord lengths and orientations are fixed by their placement at the wing root and tip, rendering such variations physically meaningless.
- Airfoil spanwise locations are not varied. Allowing these locations to change may result in overlapping airfoils, leading to unnecessary geometric failures. Varying the shapes and orientations of five fixed airfoil sections was deemed sufficient to generate a large and representative wing design space.
- Nacelle length and offset parameters are defined relative to the wing MAC. This ensures a realistic relative scaling between wing and nacelle, while still allowing both relatively small and relatively large nacelles to be sampled within the specified bounds.
- Nacelle cant and rotation angles are excluded, as these parameters were found to have a disproportionate impact on the robustness of the wing-nacelle matching and integration procedures. The current wing-nacelle integration approach is known to be sensitive to such variations.
- A physically realistic nacelle shape is enforced through dependencies between nacelle cross-section class factors and widths. The two intermediate cross-sections are constrained to share identical CST class factors, and all cross-sections are required to have the same width.
- Nacelle cross-section CST shape coefficients are omitted, as (pseudo-)random sampling of these parameters led to excessive local curvature variations in the nacelle and resulted in unrealistic nacelle geometries.
- The scoop geometry is scaled relative to the nacelle, ensuring consistent intake proportions across the sampled design space.
- Scoop lip thicknesses are defined relative to the lower lip length to preserve realistic geometric proportions.
- Additional dependencies are introduced in the matching and integration subsystems to prevent physically meaningless configurations, such as a scoop reattachment location positioned upstream of the scoop itself.

D.1. Examples Generated Designs

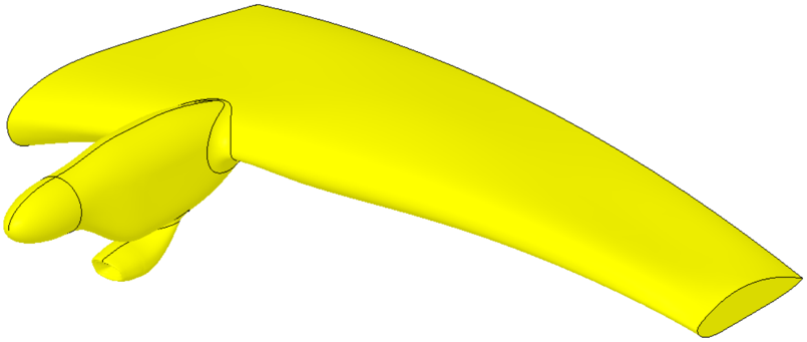
A few geometrically successful designs from the 200 generated designs from section 8.1.3 under Results are shown in figure D.1 highlighting the large design space used in the robustness study.



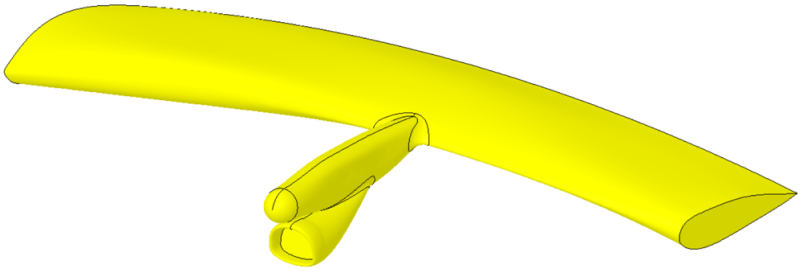
(a) Design 1 (success)



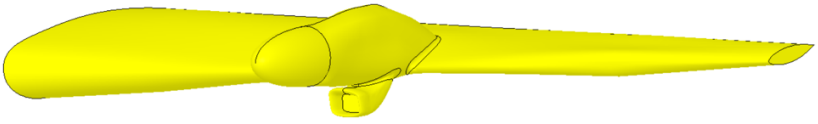
(b) Design 2 (success)



(c) Design 3 (success)



(d) Design 4 (success)



(e) Design 5 (mesh divergence)

Figure D.1: Some designs from the parametric robustness study.

TODO: failure cases

E

Needs and Requirements

This appendix contains the complete Requirements Package underlying the development of the parametric KBE tool. It is organised into three parts. First, the Needs package presents the full set of Internal and External Needs from which the tool's functionality is derived in section E.1. Second, the System Requirements provide the corresponding specifications, structured according to their relation to the two hypotheses in section E.2. Finally, section E.3 summarises how each requirement is addressed in the thesis, including references to the relevant sections in the main body.

E.1. Needs

The Needs package contains Internal and External Needs.

E.1.1. Internal Needs

The Needs diagram for all the Needs related to Hypothesis 1 can be seen in figure E.1 and Needs connected to Hypothesis 2 in figure E.2. Below each diagram, all the individual Needs are listed with a tag, title, description and a more elaborate clarification.

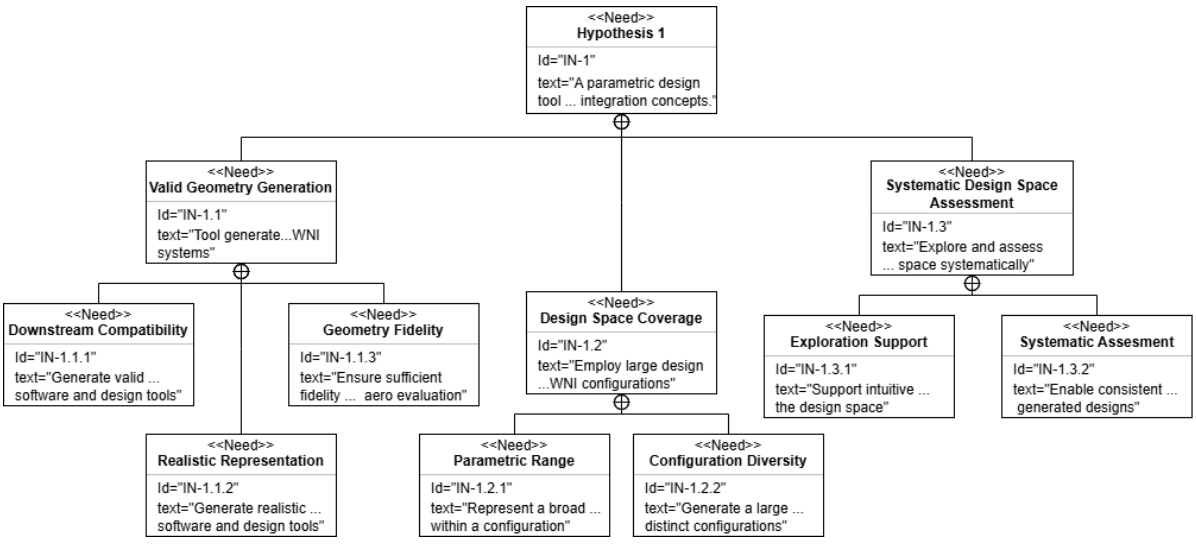


Figure E.1: Needs diagram for internal needs based on Hypothesis 1

IN-1 - Hypothesis 1

A parametric design tool, capable of generating valid external geometries for wing-nacelle-intake systems, will significantly expand the design space and enable the identification of aerodynamically effec-

tive configurations through the systematic exploration of unconventional integration concepts.

The first top-level Internal Need is Hypothesis 1. Since Needs are equivalent to external functionalities of the tool, and the hypotheses contain all the required functionalities of the tool in order to complete the research objective in section 1.4.2, the first top-level Need is the first hypothesis.

IN-1.1 - Valid Geometry Generation
Generate valid external geometry for wing-nacelle-intake systems.

This Need addresses the tool's ability to create geometrically and computationally valid external surfaces, as stated in "...capable of generating valid external geometries..." in Hypothesis 1.

IN-1.1.1 - Downstream Compatibility
Generate valid geometries compatible with downstream analysis software and design tools.

To ensure interoperability, the geometries produced by the tool must be directly compatible with downstream design and analysis tools and software.

IN-1.1.2 - Realistic Representation
Generate realistic outer mould lines representative of FC-powered aircraft.

This Need specifies that generated geometries must accurately reflect the physical and aerodynamic characteristics of FC-powered aircraft, ensuring realism and relevance for subsequent analyses.

IN-1.1.3 - Geometry Fidelity
Ensure sufficient level of detail for the generated geometries for aerodynamic evaluation.

The geometric detail must be high enough to capture relevant three-dimensional flow effects around integrated WNI systems, ensuring adequate accuracy for preliminary aerodynamic studies.

IN-1.2 - Design Space Coverage
Create a large design space for diverse wing-nacelle-intake configurations.

This Need relates to Hypothesis 1 that the tool should generate designs "throughout a large the design space...". The tool must be capable of generating a wide variety of geometries, enabling exploration of both *conventional* and *unconventional* configurations[1].

IN-1.2.1 - Parameter Range
Include a broad range of geometric parameters within a configuration.

Within each configuration, the tool must allow variation across numerous geometric parameters to enable meaningful and a broad variety of design modifications and comprehensive parametric studies.

IN-1.2.2 - Configuration Diversity
Generate a large number of distinct configurations.

The tool should be capable of producing distinctly different configuration families. In this context, a configuration family is distinguished by the type and amount of components it contains. Examples of distinct configurations are a wing with a circular blended nacelle or a square nacelle with an underbelly scoop intake.

IN-1.3 - Systematic Design Space Assessment
Explore and assess the design space systematically.

This Need stems from the requirement in Hypothesis 1 to enable "systematic exploration and identification of aerodynamically effective configurations."

IN-1.3.1 - Exploration Support
Support intuitive and systematic exploration of the design space.

The tool must provide an intuitive workflow for creating and modifying designs, where parameters are meaningful to aerodynamic designers and enable repeatable exploration.

[1] Exact definitions of *conventional* and *unconventional* configurations are defined in chapter 1 for the context of this thesis.

IN-1.3.2 - Systematic Assessment

Enable consistent assessment of generated designs.

This Need specifies that the tool should serve as a generic aerodynamic assessment platform for propulsion integration in FC-powered aircraft, directly addressing the second problem statement in section 1.4.

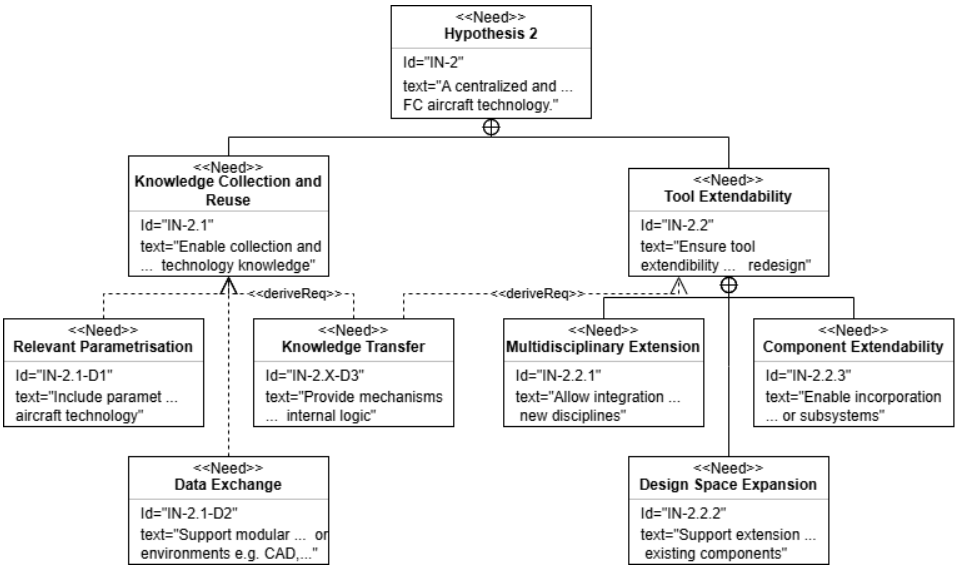


Figure E.2: Needs diagram for internal needs based on Hypothesis 2

IN-2 - Hypothesis 2

A centralized and extendable design tool will facilitate systematic knowledge collection and reuse, thereby improving design efficiency and enhancing the understanding of intake design, engine integration, and FC aircraft technology.

This top-level Internal Need corresponds directly to Hypothesis 2. Following the same reasoning as in IN-1 - Hypothesis 1, this hypothesis represents the core external functionalities required for the tool to fulfil the research objective defined in section 1.4.2.

IN-2.1 - Knowledge Collection and Reuse

Enable collection and reuse of knowledge critical for intake design, engine integration, and FC-aircraft technology knowledge.

This Need addresses the phrase "...facilitate systematic knowledge collection and reuse..." from Hypothesis 2. It represents a broad requirement that the tool must contribute to structured knowledge generation, retention, and reuse within the field. To make this abstract goal more concrete, three specific Needs are derived below. This Need directly address the third problem posed in section 1.4 as this tool will enable research on these unconventional WNI configurations for FC-powered flight.

IN-2.1-D1 - Relevant Parametrisations

Include parametrisations relevant to studies on intake design, engine integration, and FC aircraft technology.

To enable meaningful research on intake design, propulsion integration, and fuel-cell (FC) aircraft systems, the tool must provide parametrisations that capture the most influential geometric and aerodynamic characteristics associated with these domains. Ensuring the relevance of these parameters is essential for producing configurations that are representative of real design studies and suitable for performance evaluation.

This Need differs from section E.1.1 and section E.1.1. While the present Need concerns the relevance and meaningfulness of the chosen parameters, section E.1.1 addresses whether the number of parameters are sufficient to enable flexible parametrisations, and section E.1.1 relates to the tool's capability

to generate distinct families of configurations. The latter is often a natural consequence of implementing multiple parametrisation schemes for a single component as for instance, employing both CST and NACA 4-series parametrisations for the airfoil shape results in two distinct airfoil families: CST-based and NACA 4-series airfoils.

IN-2.1-D2 - Data Exchange
Support modular data exchange with other tools or environments (e.g., CAD, CFD, sizing).

As the tool will operate alongside other design and analysis environments, it must support standardized data exchange. This ensures interoperability across different workflows and software tools used in propulsion integration and aircraft design studies.

This Need is different from section E.1.1 as this Need relates to the fact that data transfer capability and section E.1.1 to the quality of a particular data transfer, namely the geometrical data.

IN-2.X-D3 - Knowledge Transfer
Provide mechanisms for documenting and transferring knowledge and internal logic.

Transparency and traceability are essential for the tool's role in systematic knowledge reuse. Therefore, it must include mechanisms for documenting the internal logic, assumptions, and modelling strategies to ensure that embedded knowledge can be transferred and reused effectively. This transparency enables the knowledge collection and reuse as well as improves the extendibility of the model, leading to this Need to be derived from IN-2.1 - Knowledge Collection and Reuse and IN-2.2 - Tool Extendibility.

IN-2.2 - Tool Extendibility
Ensure tool extendibility without redesign.

This Need follows from the phrase "...extendable design tool..." in Hypothesis 2. As the tool is not intended for a single fixed application, it must remain flexible and easily extendable to incorporate new parametrisations or disciplinary models without requiring fundamental architectural redesign. The tool's extendibility directly contributes to addressing Problem 1 and Problem 3 from section 1.4: it ensures that while the initial version may not yet yield superior WNI designs compared to retrofitted ICE configurations, future extensions can build toward that goal. Moreover, since the aerodynamic behaviour and design space of unconventional WNI configurations are still not well understood, the ability to extend the tool is crucial to integrate new or improved parametrisations as research in this field advances.

IN-2.2.1 - Multidisciplinary Extension
Allow integration of new disciplines.

This Need specifies that the tool's architecture must support the addition of new disciplinary modules, such as structural, or performance analyses, without altering the core framework.

IN-2.2.2 - Design Space Expansion
Support extension of the design space for existing components.

This Need requires that the tool can incorporate new or refined parametrisations within existing components, thereby expanding the possible design space without requiring fundamental model restructuring or knowing internal system implementation details.

IN-2.2.3 - Component Extendability
Enable incorporation of new geometric components or subsystems.

This Need specifies that new geometric components or subsystems, such as alternative intake types, propeller, or hybrid-electric elements, can be added to the tool's architecture while maintaining overall system integrity.

E.1.2. External Needs

External Needs originate from stakeholders rather than from the literature and are therefore not directly tied to the research topic from section 1.4. In this project, the primary stakeholders are TU Delft and the author. The External Needs diagram is shown in figure E.3.

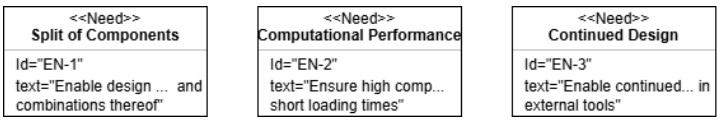


Figure E.3: Needs diagram for external needs

EN-1 -Split of Components

Enable design and analysis of individual components and combinations thereof.

This Need originates from TU Delft, requiring that individual components (e.g., wing, nacelle, or intake) can be designed and analysed independently or in combination. This flexibility allows for isolated testing and evaluation of specific subsystem interactions.

EN-2 - Computational Performance

Ensure high computational performance and short loading times.

Efficient computational performance is critical, as the tool is expected to be used for large-scale parametric studies involving numerous design iterations. High generation speed ensures the practicality of design exploration and due to the many design iterations and analyses, increases the confidence in observed aerodynamic trends.

EN-3 - Continue of Design

Enable the continue of design of generated geometries in external tools.

Because the tool may not capture all details of a complete WNI design, users must be able to continue refining and extending generated geometries in external CAD or design software. This requirement ensures design continuity and mitigates the risk of limited geometric flexibility identified in section 3.1.4.

E.2. System Requirements

The System Requirements package is divided into two subsets: System Requirements 1 and System Requirements 2. The first subset comprises Requirements derived from Internal Needs associated with Hypothesis 1, while the second contains those originating from Needs related to Hypothesis 2. External Needs are included in both System Requirements 1 and System Requirements 2.

E.2.1. System Requirements 1

The diagrams, containing all Requirements of the System Requirements 1 package are presented in figure E.4, figure E.5 and figure E.6.

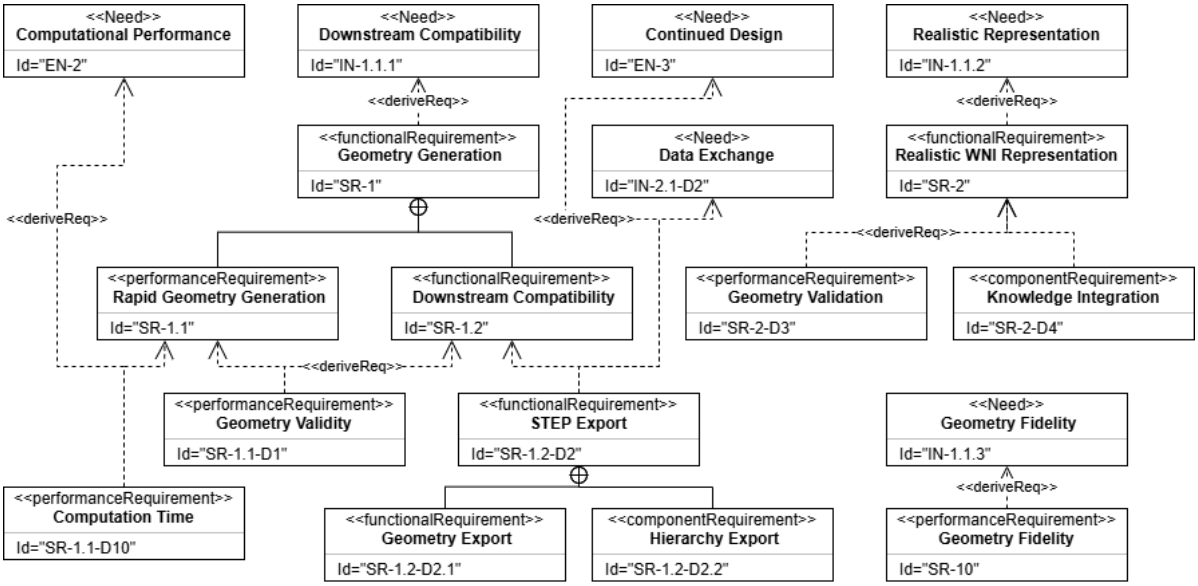


Figure E.4: System Requirements 1 package (1/3)

SR-1 - Geometry Generation

The tool shall automatically generate valid geometries that are quickly usable in downstream software.

This system requirement is derived from IN-1.1.1 - Downstream Compatibility and defines the tool's primary function: to automatically generate valid and realistic external geometries suitable for downstream applications.

SR-1.1 - Rapid Geometry Generation

The tool shall generate valid geometry rapidly.

SR-1.1-D10 - Computation Time

The tool shall compute a single design in under 10 seconds.

This performance requirement directly derives from SR-1.1 - Rapid Geometry Generation and the external need EN-2 - Computational Performance. The 10-second limit is set as the upper threshold for the generation of a single design, considering that the tool is expected to produce hundreds of designs per use case. With this constraint, total runtimes are expected to range between approximately 15 minutes and 2 hours, which is acceptable and does not significantly impede research or development workflows.

SR-1.1-D1 - Geometry Validity

Generated geometry shall be manifold, complete, valid, simple, and accurate.

This requirement ensures that the generated geometries are computationally valid and stable for subsequent processes, such as aerodynamic analysis (including meshing) and geometry export, as referenced in SR-1.2-D2 - STEP Export. Commonly used CFD (Computational Fluid Dynamics) software includes commercial solvers such as ANSYS CFX and Fluent[2], as well as open-source alternatives like OpenFOAM[3].To ensure simulation usability, the generated geometries must meet the following quality criteria [70]:

- **Completeness**: Fully connected and closed geometry with no gaps.
- **Intersection-free:**: Free from self-intersections, overlaps, or loose geometry.
- **Manifold**: The geometry must be plane-like when inspected closely, with no edges connecting more than two faces, no internal faces, no single-vertex connections, consistent normals, and no redundant elements.

[2] https://www.ansys.com/
[3] https://www.openfoam.com/

- **Simplicity**: Avoidance of excessive detail while maintaining required fidelity.
- **Accuracy**: Correct scaling and compatibility with the target software.

The underlying geometry kernel (OCC) used in ParaPy inherently enforces many of these criteria, ensuring that generated geometries meet validity requirements. Further details on geometry verification and full definition of valid geometry with specific criteria are provided in chapter 8.

SR-1.2 - Downstream Compatibility
The tool shall generate geometry compatible with downstream software.

The generated geometry must be compatible and readily usable in downstream software. Detailed requirements for geometric validity are specified in SR-1.1-D1 - Geometry Validity.

SR-1.2-D2 - STEP Export
The tool shall export geometry in STEP format including all relevant data.

For downstream compatibility, the geometry must be exportable in standard formats such as IGES, STEP, or STL, as these are commonly used in CFD workflows [70]. In this work, the STEP (Standard for the Exchange of Product Data) format is selected due to its ability to preserve geometry, assembly hierarchy, Product and Manufacturing Information (PMI), and product metadata. STEP is defined under ISO 10303 as a standard for digital product data exchange between computer-aided systems [84]. The STEP export functionality is essential for both IN-2.1-D2 - Data Exchange and EN-3 - Continue of Design.

SR-1.2-D2.1 - Geometry Export
The tool shall export all external and helper geometry to STEP.

The STEP export capability shall include both the primary external geometry (used in downstream analyses and manipulation) and all helper geometry used in the generation process. Helper geometry refers to supporting entities such as 2D lines, cross-sections, and reference points that assist in defining the external geometry but do not represent physical components.

SR-1.2-D2.2 - Hierarchy Export
The tool shall export the complete model hierarchy to STEP.

Since the exported model may include numerous geometrical entities (external and helper), the export must preserve the model hierarchy to maintain organization and traceability. This allows helper geometries to remain linked to their corresponding components in the assembly structure.

SR-2 - Realistic WNI Representation
The tool shall generate realistic geometries of WNI configurations representative of FC-powered aircraft.

Derived from IN-1.1.2 - Realistic Representation, this requirement formalizes that internal need as a specific system-level requirement.

SR-2-D3 - Geometry Validation
The geometry shall be validated against representative WNI configurations.

To ensure that generated geometries accurately represent WNI configurations, model outputs must be validated against reference designs. Validation procedures and criteria are detailed in chapter 8.

SR-2-D4 - Knowledge Integration
The model shall incorporate all the knowledge of WNI configurations for FC-powered aircraft researched in the literature study.

This requirement refers to the integration of domain-specific knowledge into the modelling approach. All parametrizations and design philosophies implemented in this work must incorporate the physics and aircraft design insights presented in [24].

SR-10 - Geometry Fidelity

The generated geometry shall maintain a fidelity level such that the smallest features correspond to at least 1% of the relative model size.

As defined in section 1.4.1, the tool aims to produce geometries with sufficient fidelity to capture global aerodynamic interactions while omitting minor details not critical to preliminary design. This requirement establishes a quantitative guideline for geometric resolution. For example, if the Mean Aerodynamic Chord (MAC)[4] is 3 meters, the smallest significant feature should be on the order of centimetres. Given that typical configurations are several meters in scale, this threshold ensures sufficient resolution for accurate aerodynamic representation without excessive geometric complexity.

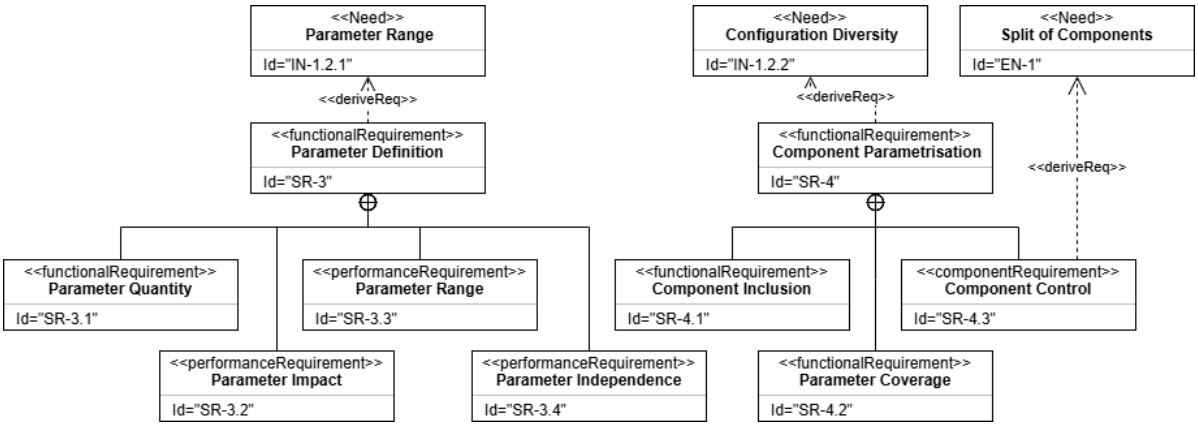


Figure E.5: System Requirements 1 package (2/3)

SR-3 - Parameter Definition

The tool shall include numerous input parameters with large value ranges that are impactful and non-constraining.

This requirement directly derives from IN-1.2.1 - Parameter Range and defines the fundamental expectation that the tool supports a broad and flexible parameter space, allowing comprehensive exploration of the design domain.

SR-3.1 - Parameter Quantity

The tool shall provide a sufficient number of input parameters, such that highly complex components include at least 10 parameters, moderately complex components at least 5 parameters, and low-complexity components at least 2 parameters.

This requirement addresses the first aspect of SR-3 - Parameter Definition, specifying that the tool must provide extensive parametrisation capabilities. The number of parameters is inherently dependent on the complexity of the component being modelled.

SR-3.2 - Parameter Impact

Each input parameter shall have a significant impact on the resulting design, such that a variation of 1% of the feasible parameter range leads to a geometric change exceeding the model fidelity threshold.

Each input parameter must meaningfully influence the generated geometry. Varying a parameter across its full range should produce a geometric change greater than the minimum resolution defined in SR-10 - Geometry Fidelity. This ensures that every parameter contributes to the overall shape definition and that no redundant or negligible parameters are included.

SR-3.3 - Parameter Range

Each input parameter shall have a large range of possible values of at least 0.75–1.25 times its default value.

[4] For designs including a wing, MAC is used as the characteristic length. For nacelle-only configurations, nacelle length serves this role, as both are typically of similar magnitude.

Each parameter must be defined over a large continuous range of values, resulting in a significant flexibility of design generation for each parameter.

SR-3.4 - Parameter Independence
Input parameters shall not constrain or conflict with each other.

Parameter independence ensures that adjusting one input does not unintentionally restrict or invalidate the permissible range of another. This requirement guarantees design flexibility and prevents hidden parameter coupling, enabling a fully parametric and modular modelling framework.

SR-4 - Component Parametrisation
The tool shall include parametrisations for all components presented in section 2.1.1, section 2.2.1 and section 2.3.1.

This requirement ensures that all components listed in the defined modelling scope are represented in the tool. Each component must be implemented with an appropriate parametrisation, consistent with the level of detail and fidelity established in the system scope.

SR-4.1 - Component Inclusion
The tool shall include all components presented in section 2.1.1, section 2.2.1 and section 2.3.1.

This Requirement specifies that all relevant physical components, those contributing to sizing, intake, and propulsion modelling, shall be present within the tool's architecture. The inclusion of all scoped components ensures full geometric and functional representation of the system.

SR-4.2 - Parameter Coverage
Each component shall be parametrised so that all parameters defined in section 2.1.1, section 2.2.1 and section 2.3.1 are included in at least one parametrisation.

For each included component, relevant aerodynamic and geometric parameters are identified in chapter 2. Because all listed parameters are considered physically significant, each must be incorporated into at least one parametrisation within the tool to ensure comprehensive design coverage and flexibility.

SR-4.3 - Component Control
Each modelled component shall be individually turned on and off.

This requirement directly derives from EN-1 -Split of Components and ensures that each modelled component can be individually enabled or disabled within the model. Such modular control allows for isolated testing and analyses, and flexible configuration of component combinations during design exploration.

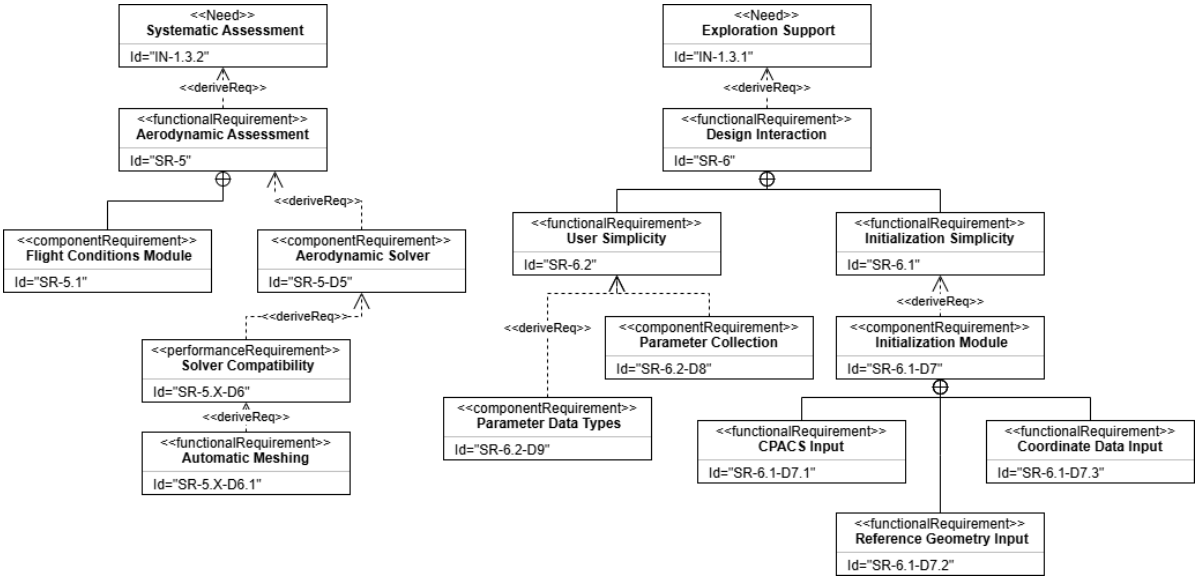


Figure E.6: System Requirements 1 package (3/3)

SR-5 - Aerodynamic Assessment

The tool shall enable preliminary aerodynamic assessment of a design without manual preprocessing, except for specifying flow conditions and solver settings.

This requirement directly relates to automatic aerodynamic capability of the tool from problem 2 in section 1.4. It states that the tool shall preprocess the designs automatically for a mid-fidelity aerodynamic analyses.

SR-5.1 - Flight Conditions Module

The tool shall include a flight-conditions module with computed and user-overrideable flow and flight parameters.

A dedicated module shall compute standard flow and flight conditions automatically while allowing user input when necessary. This flexibility ensures both repeatable and customizable aerodynamic simulations under various operating conditions.

SR-5-D5 - Aerodynamic Solver

The tool shall include a low- to mid-fidelity aerodynamic solver capable of estimating lift, drag, and other performance metrics taking into account the complete geometry with the fidelity described IN-1.1.3 - Geometry Fidelity.

The tool shall integrate a mid-fidelity aerodynamic solver capable of capturing the aerodynamic characteristics of complex geometries. This ensures that the solver reflects the geometric resolution described in SR-10 - Geometry Fidelity, enabling consistent evaluation of aerodynamic trends and flow interactions.

Additionally, the solver serves as a verification mechanism, as stable solver performance and sensible results indicate valid and computationally robust geometries. The reasoning behind the solver choice is detailed in section 4.3.4, while verification results are presented in chapter 8.

SR-5-X-D6 - Solver Compatibility

Generated geometries shall be compatible with the integrated aerodynamic solver and be thus surface-meshable.

Since the generated geometry must be compatible with the aerodynamic solver, it must be surface meshable. This is achieved by integrating an automatic meshing algorithm into the tool. Since the fidelity of the solver, as stated in SR-5-D5 - Aerodynamic Solver, is high enough that the full geometry of the model is analysable the model must be surface meshable as this mesh takes into account the full

fidelity of the model. If this meshing algorithm succeeds, the geometry is compatible with the integrated aerodynamic solver.

SR-5-X-D6.1 - Automatic Meshing
The tool shall include an automatic surface meshing module capable of exporting meshes.

The tool shall include an automated surface meshing module capable of creating and exporting solver-ready surface meshes. While mesh quality evaluation lies outside the scope of this thesis, the ability to automatically mesh and export designs ensures geometric validity and compatibility with downstream aerodynamic solvers, as discussed in chapter 8 and chapter 9.

SR-6 - Design Interaction
Design generation and modification shall be intuitive and straightforward for the user.

The tool shall provide a clear and efficient interface that allows users to generate and modify designs without extensive training or complex commands. This ensures that the focus remains on design exploration and analysis rather than on software operation.

SR-6.1 - Initialization Simplicity
Model initialization shall be easy and intuitive.

The model shall offer an accessible initialization process that enables users to start new designs with minimal input while maintaining flexibility for more advanced configurations. This reduces setup time and facilitates rapid design iteration.

SR-6.1-D7 - Initialization Module
The model shall include initialisation modules capable of creating a design from aircraft design data or reference geometry.

A dedicated initialization module shall handle data import and geometry creation from standardized sources, allowing for consistent and reproducible model setup. This supports the seamless transition between conceptual design data and parametric model generation.

SR-6.1-D7.1 - CPACS Input
The tool shall initialize a design from CPACS-formatted data.

The initialization module shall interpret Common Parametric Aircraft Configuration Scheme (CPACS) data structures directly, ensuring compatibility with established aircraft design workflows. This enables integration with existing MDO pipelines and institutional tools used in aerospace design [85].

SR-6.1-D7.2 - Reference Geometry Input
The tool shall initialize a design from reference STEP geometry.

The initialization module shall compute parametric input values from imported STEP geometry, enabling static models to be converted into fully parameterised designs. This supports further optimization and research on existing geometries.

SR-6.1-D7.3 - Coordinate Data Input
The tool shall initialize airfoils and other cross-sections and curves from coordinate data.

The initialization module shall read 2D coordinate datasets to define cross-sectional geometry, allowing users to incorporate custom airfoils or profiles. This increases flexibility and promotes consistency with existing aerodynamic datasets.

SR-6.2 - User Simplicity
Design changes shall be made easily and intuitively.

The tool shall allow direct and intuitive modification of parameters, ensuring that design updates are immediately reflected in the model and relevant to aircraft design. This supports iterative design studies and encourages active exploration of the design space.

SR-6.2-D8 - Parameter Collection
All relevant model input parameters shall be organized into a single data collection.

All inputs shall be logically grouped within a unified data container, simplifying model control and data management. This approach ensures traceability and facilitates the integration of the tool into automated workflows.

SR-6.2-D9 - Parameter Data Types
Each input parameter shall use a primitive data type such as float, integer, string, or boolean.

All input variables shall employ simple, well-defined data types to maintain clarity and consistency in the model structure. This simplifies parameter manipulation, data exchange, and debugging across modules.

E.2.2. System Requirements 2

The diagram, containing all Requirements of the System Requirements 2 package are presented in figure E.7.

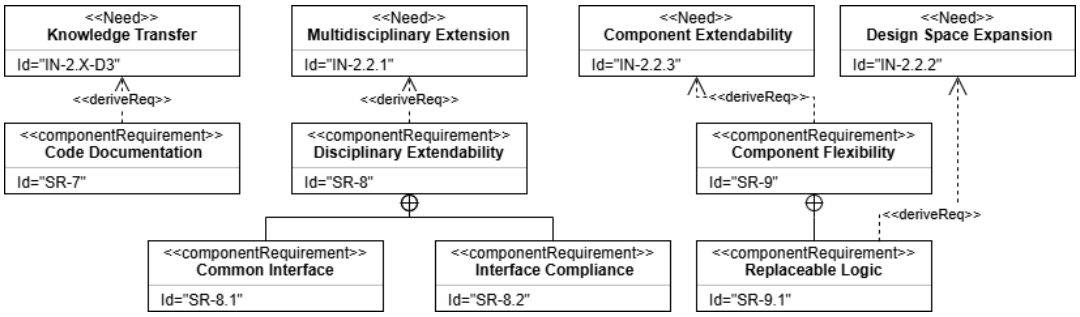


Figure E.7: System Requirements 2 package

SR-7 - Code Documentation
The model code shall include clear documentation explaining its structure and use.

To ensure transparency and knowledge transfer, the model's codebase shall include thorough inline documentation and a structured user guide via doc-strings. This documentation shall explain the tool's internal logic, data flow, and extension mechanisms, facilitating long-term maintainability and collaboration.

SR-8 - Disciplinary Extendability
The tool shall be easily extendable with new disciplinary capabilities.

The tool shall support straightforward integration of new analysis disciplines without requiring architectural redesign. This ensures scalability of the framework and enables continuous development as new research areas or methods emerge.

SR-8.1 - Common Interface
All disciplinary modules within the model shall implement a common interface.

Each discipline shall conform to a shared interface definition, ensuring consistent communication and data exchange across modules. This approach enables new disciplines to be added with minimal integration effort while maintaining functional consistency.

SR-8.2 - Interface Compliance
The functionality of each discipline shall operate using a shared interface.

All disciplinary functionality shall operate through the common interface rather than using discipline-specific implementations. This guarantees interoperability between modules and ensures that newly developed disciplines automatically inherit shared capabilities when implemented correctly.

SR-9 - Component Flexibility
Each top-level component shall be defined by a simple class containing the fundamental design rules, unconstrained by lower-level details.

Each major subsystem shall be implemented as a lightweight class encapsulating only the essential design logic. This simplifies component internals and allows developers to add new behaviour or functionality without structural dependencies on lower-level details, leading to modular and uncoupled design.

SR-9.1 - Replaceable Logic
Each top-level component shall allow replacement or injection of new rules or parametrisations without modifying the top-level class.

Parametrisations and rules shall be modularized into independent units that can replace or extend existing functionality without sub-classing. This modular approach promotes flexibility, reduces coupling, and supports iterative tool enhancement as new design insights are developed.

E.3. Assessment of System Requirements

This section presents the assessment of the defined system requirements to verify that the developed tool fulfills its intended functionality. Each system requirement is evaluated based on whether the implemented features and resulting capabilities satisfy the specified criteria. The tool has been designed and developed with these system requirements in mind as some features are direct implementations of particular requirements, while others have been guided by them to ensure coherent and goal-oriented development.

The successful confirmation of each system requirement demonstrates that the corresponding Needs are satisfied. Since these Needs directly originate from the thesis hypotheses, their fulfillment collectively supports the confirmation of these Consequently, verifying the satisfaction of all system requirements thus confirms that the research objective has been achieved.

E.3.1. Requirement Satisfaction Scale

Requirement satisfaction is evaluated using a five-level qualitative scale that reflects the degree to which the requirement criteria are fulfilled. Where applicable, the assessment also indicates the subset of designs for which the stated level of satisfaction applies.

- **Fully satisfied (100%)**: All aspects of the requirement are met without limitations, fulfilling the intended functionality in full.
- **Largely satisfied (75%)**: The requirement is met to a high degree, with only minor deviations that do not significantly affect the intended functionality.
- **Partially satisfied (50%)**: The requirement is met in part, with clear limitations or dependencies that restrict full compliance.
- **Marginally satisfied (25%)**: Only limited aspects of the requirement are fulfilled, providing restricted or conditional compliance.
- **Not satisfied (0%)**: The requirement is not met, or compliance cannot be demonstrated.
- **Not Assessed (NA)**: The requirement could not be evaluated within the scope of this work.

This means that compliance is evaluated based on the degree to which the requirement is satisfied and, where applicable, the subset of designs for which this level of satisfaction holds.

SR-1 - Geometry Generation
The tool shall automatically generate valid geometries that are quickly usable in downstream software.
Since this requirement is composed fully by SR-1.1 - Rapid Geometry Generation and SR-1.2 - Downstream Compatibility, which are largely and fully satisfied respectively, this requirement is *largely satisfied*.

SR-1.1 - Rapid Geometry Generation
The tool shall generate valid geometry rapidly.

This requirement is decomposed into SR-1.1-D10 - Computation Time and SR-1.1-D1 - Geometry Validity, as indicated by the derivation relationships in figure E.4. Consequently, the assessment of these derived requirements directly determines compliance with SR-1.1 - Rapid Geometry Generation.

Requirement SR-1.1-D10 - Computation Time is fully satisfied, as all successfully generated manual designs are produced within the prescribed time threshold. Requirement SR-1.1-D1 - Geometry Validity is satisfied conditionally, with compliance depending on the selected input parameters provided to WINGen. Taken together, SR-1.1 - Rapid Geometry Generation is therefore satisfied in an input-dependent manner: it is fully satisfied for manually generated designs and partially satisfied for automatically generated designs leading to it being *largely satisfied*.

SR-1.1-D10 - Computation Time
The tool shall compute a single design in under 30 seconds.

The computation time required to generate the final surface geometry is evaluated for the designs in the manual design feasibility study (section 8.1.2). The *baseline design* is generated in 11.5–13.5 s, the *Havilland Dash 8 design* in 18.5–19.5 s, and the *Elysian design* in 7.5–10 s.

Execution times show limited run-to-run variability due to system-level effects such as hardware characteristics, background processes, and cache behaviour, and are therefore interpreted as an order-of-magnitude measure. Differences between designs are primarily governed by their geometric–numerical complexity[5]. All evaluated designs remain well below the 30 s threshold and is in the correct order of magnitude, *fully satisfying* requirement SR-1.1-D10 - Computation Time.

SR-1.1-D1 - Geometry Validity
Generated geometry shall be manifold, complete, valid, simple, and accurate.

Geometric validity of the generated designs is ensured through the comprehensive geometry assessment procedure described in section 8.1.1. All validity criteria (completeness, absence of self-intersections, manifoldness, simplicity, and accuracy) are explicitly evaluated by the `verify_model()` function, which is executed via the `verification_result` slot for the complete system assembly.

Results from the model robustness study (section 8.1.3) indicate that not all automatically generated designs satisfy this requirement, demonstrating that compliance is strongly dependent on the selected input parameters. Nevertheless, all manually generated designs meet the geometry validity criteria, as does a substantial fraction (30–35%) of the automatically generated design set. To conclude SR-1.1-D1 - Geometry Validity is *largely satisfied*.

SR-1.2 - Downstream Compatibility
The tool shall generate geometry compatible with downstream software.

This requirement is supported by two derived requirements: SR-1.1-D1 - Geometry Validity and SR-1.2-D2 - STEP Export. Compliance with SR-1.2 - Downstream Compatibility therefore follows directly from the assessment of geometric validity and the ability to export geometry in a suitable exchange format.

As the satisfaction of SR-1.1-D1 - Geometry Validity is dependent on the selected input parameters, while SR-1.2-D2 - STEP Export is fully satisfied, the downstream compatibility requirement is satisfied in an input-dependent manner. Consequently, SR-1.2 - Downstream Compatibility is *fully satisfied* for valid designs generated by the tool.

SR-1.2-D2 - STEP Export
The tool shall export geometry in STEP format including all relevant data.

The inclusion of a dedicated STEP export capability module (section 4.3.2) satisfies the functional aspect of this requirement by enabling geometry export in STEP format. The requirement regarding

[5]Geometric–numerical complexity includes but is not limited to curve interpolation, optimisation during geometry import and boolean operations..

the inclusion of relevant data is further satisfied through the complete compliance with both SR-1.2-D2.1 - Geometry Export and SR-1.2-D2.2 - Hierarchy Export, leading to *full satisfaction*.

SR-1.2-D2.1 - Geometry Export
The tool shall export all external and helper geometry to STEP.

Through the introduction of a dedicated model traversal algorithm (chapter C), a user-defined separation between *root* and *helper* geometry is applied during STEP export. This approach ensures that all relevant geometric entities are included in the exported files, thereby *fully satisfying* this requirement.

SR-1.2-D2.2 - Hierarchy Export
The tool shall export the complete model hierarchy to STEP.

The same model traversal algorithm (chapter C) enables the export of the complete model hierarchy for user-specified components, defined via *child references*. As a result, the full hierarchical structure is preserved in the STEP output, leading to *full satisfaction* of this requirement.

SR-2 - Realistic WNI Representation
The tool shall generate realistic geometries of WNI configurations representative of FC-powered aircraft.

SR-2 - Realistic WNI Representation is supported by both SR-2-D3 - Geometry Validation and SR-2-D4 - Knowledge Integration. Consequently, the satisfaction of SR-2 - Realistic WNI Representation follows directly from the assessment of these derived requirements and is therefore *partially satisfied*.

SR-2-D3 - Geometry Validation
The geometry shall be validated against three representative WNI configurations, where the tool-generated geometry must differ by less than 1% in terms of RMS and maximum Euclidean distance.

The geometry of a wing–nacelle configuration is successfully validated against a reference model, yielding a geometric deviation RMS of 0.13% and a maximum deviation of 0.49%. However, as only a single validation case is considered-and this case does not include a scoop intake or a non-straight wing-this requirement is only *marginally satisfied*.

SR-2-D4 - Knowledge Integration
The model shall incorporate all relevant knowledge on WNI configurations for FC-powered aircraft as identified in the literature study.

The modelling of all system components is directly informed by the preliminary parametrisations presented in chapter 2, thereby embedding a substantial portion of the reviewed knowledge within WINGen. However, these parametrisations are limited to the scope of the preliminary aerodynamic design studies and do not incorporate the research findings related to FC and thermal management system (TMS) sizing or integration methods. Furthermore, no dedicated parametric investigation of WNI aerodynamic behaviour has been conducted beyond the preliminary aerodynamic validation presented in section 8.2.3.

As a result, while relevant literature-based knowledge is partially embedded in the model, the overall requirement is only *partially satisfied*.

SR-10 - Geometry Fidelity
The generated geometry shall maintain a fidelity level such that the smallest features correspond to at least 1% of the relative model size.

The size of geometric features in the generated models is strongly dependent on the selected input parameters, as low values for geometric size inputs directly result in correspondingly small features or components. The relative model size is defined as the mean aerodynamic chord (MAC) where available, or otherwise the nacelle length.

Inspection of the manually generated designs (section 8.1.2) shows that the majority of geometric features exceed 1% of the relative model size, with MAC values ranging from 2.04 m for the *baseline*

design to 3.35 m for the *Havilland Dash 8 design*. This is reflected in the selected input parameter values, of which almost none fall below 1% of the corresponding reference length. An exception is formed by the scoop lip radii, which for all designs reach values as low as 0.005 m, placing this specific geometric feature below the 1% threshold.

Furthermore, the parameter specifications (chapter D) applied in the design space coverage study (section 8.1.3) allow lower bounds for the diverter height (0.5% of nacelle length) and scoop lip radii (0.1% of nacelle length), both of which lie well below the 1% fidelity criterion. Taken together, these observations lead to a *largely satisfied* assessment of requirement SR-10 - Geometry Fidelity.

SR-3 - Parameter Definition

The tool shall include numerous input parameters with large value ranges that are impactful and non-constraining.

Requirement SR-3 is composed of SR-3.1 - Parameter Quantity, SR-3.2 - Parameter Impact, SR-3.3 - Parameter Range, and SR-3.4 - Parameter Independence. As these derived requirements are assessed as largely satisfied, not assessed (NA), largely satisfied, and partially satisfied, respectively, the overall requirement is considered *partially satisfied*.

SR-3.1 - Parameter Quantity

The tool shall provide a sufficient number of input parameters, such that highly complex components include at least 10 parameters, moderately complex components at least 5 parameters, and low-complexity components at least 2 parameters.

The number of input parameters per component is summarised in table D.1. The only low-complexity component (spinner) meets the minimum requirement with two input parameters. The moderately complex components-wing–nacelle matching, nacelle–scoop integration, and wing–nacelle integration-contain fewer than the required five parameters, indicating that their parametrisations should be extended to increase design freedom and flexibility. In contrast, all highly complex components (wing, nacelle, and scoop) exceed the minimum requirement of ten parameters. Overall, WIN-Gen provides a sufficient number of parameters for low- and high-complexity components but an insufficient number for moderately complex components, leading to a *largely satisfied* assessment of this requirement.

SR-3.2 - Parameter Impact

Each input parameter shall have a significant impact on the resulting design, such that a variation of 1% of the feasible parameter range leads to a geometric change exceeding the model fidelity threshold.

Although a parameter impact study could be conducted by defining the feasible parameter ranges (chapter D), perturbing each parameter by 1%, and comparing the resulting geometries, such an analysis was not performed within the scope of this thesis due to time constraints. Consequently, this requirement is *not assessed (NA)*.

SR-3.3 - Parameter Range

Each input parameter shall have a large, continuous range of possible values of at least 0.75–1.25 times its default value.

The default value of each input parameter is defined as its value in the *baseline design*. The feasible continuous ranges are specified in table D.1. However, sampling within these full ranges results in a design convergence rate of only 30–35%, indicating that the effective ranges must be reduced by approximately a factor of two[6].

Two parameters do not meet the specified range requirement: the nacelle length (feasible range: [1.785 m, 2.55 m] versus required range: [1.875 m, 3.125 m]) and the scoop height (feasible range: [0.2343 m, 0.3281 m] versus required range: [0.1875 m, 0.3125 m]). As these exceptions represent only a small subset of the total parameter set, this requirement is considered *largely satisfied*.

[6] Determining the exact reduction factor would require a dedicated design space exploration achieving 100% convergence, which is beyond the scope of this work.

SR-3.4 - Parameter Independence
Input parameters shall not constrain or conflict with each other.

Parameter independence is assessed based on the number of imposed dependencies listed in table D.1. A significant portion of the input parameters (nine in total) require dependencies to ensure geometric convergence, indicating that full independence is not achieved. As these dependencies are necessary to maintain robustness of the geometry generation process, this requirement is assessed as *partially satisfied*.

SR-4 - Component Parametrisation
The tool shall include parametrisations for all components presented in section 2.1.1, section 2.2.1 and section 2.3.1.

This requirement consists of SR-4.1 - Component Inclusion, SR-4.2 - Parameter Coverage, and SR-4.3 - Component Control, which are respectively largely, largely, and fully satisfied. Consequently, SR-4 is assessed as being *largely satisfied*.

SR-4.1 - Component Inclusion
The tool shall include all components presented in section 2.1.1, section 2.2.1 and section 2.3.1.

In its current form, WIN-Gen includes full modelling of the wing, nacelle, spinner, internal subsystem geometries, and a scoop intake. All minimum components defined in the scope are therefore included. However, not all intake concepts described in section 2.2.1 are implemented: exits, leading-edge, ring, underslung, and flush intakes are omitted. These components were explicitly excluded due to time constraints and the already extensive modelling scope of the thesis. As this limitation was anticipated, the requirement is considered *largely satisfied*.

SR-4.2 - Parameter Coverage
Each component shall be parametrised so that all parameters defined in section 2.1.1, section 2.2.1 and section 2.3.1 are included in at least one parametrisation.

The internal subsystem geometry is implemented exactly as defined in section 2.1.1 (section 6.3). Furthermore, the nacelle and scoop parametrisations include at least all parameters specified in figure 2.3 and figure 2.1, respectively. However, several aerodynamically relevant scoop parameters introduced in figure 2.2 are not implemented in the current parametrisation (chapter 7), including the peripheral extent (φ_p), intake width (w_c), and diverter length (l_{div}). Additionally, area-ruling dimples are not included in the spinner parametrisation (section 6.2). As a result, this requirement is assessed as *largely satisfied*.

SR-4.3 - Component Control
Each modelled component shall be individually turned on and off.

By design, WIN-Gen allows each component (wing, nacelle, and intake) to be activated or deactivated independently (section 4.1). The geometry generation flow diagram (figure 4.3) demonstrates that the final surface generation and integration steps are independent of component activation, with all component combinations supported. This requirement is therefore *fully satisfied*.

SR-5 - Aerodynamic Assessment
The tool shall enable preliminary aerodynamic assessment of a design without manual preprocessing, except for specifying flow conditions and solver settings.

This requirement is primarily addressed through SR-5.1 - Flight Conditions Module, which is *fully satisfied* as WIN-Gen allows explicit specification of flow conditions and solver settings. In addition, SR-5-D5 - Aerodynamic Solver is derived from this requirement and is assessed as only *partially satisfied*. While WIN-Gen can automatically perform aerodynamic analyses, the current results do not yet achieve sufficient accuracy and confidence, as demonstrated by the preliminary validation presented in Aerodynamic Validation.

Consequently, SR-5 is assessed as *partially satisfied*. A more extensive investigation into mesh quality, solver settings, and aerodynamic validation is required to fully satisfy this requirement.

SR-5.1 - Flight Conditions Module
The tool shall include a flight-conditions module with computed and user-overrideable flow and flight parameters.

WIN-Gen includes a comprehensive flight-conditions module implemented within the `AppConfig` object, which is accessible throughout the modelling and simulation pipeline, including the aerodynamic solver interface. This results in *full satisfaction* of the requirement.

SR-5-D5 - Aerodynamic Solver
The tool shall include a low- to mid-fidelity aerodynamic solver capable of estimating lift, drag, and other performance metrics while accounting for the complete geometry.

WIN-Gen incorporates an aerodynamics computational module (CM) (section 4.3.4) with an automatic interface to the surface-vorticity-based solver FlightStream (FS). FS is capable of analysing the complete geometry generated by WIN-Gen at a fidelity consistent with the geometric fidelity defined in section E.2.1. However, the preliminary aerodynamic validation (section 8.2.3) indicates that the predicted aerodynamic coefficients are not yet sufficiently accurate, primarily due to limitations in surface meshing and solver parameter selection. As a result, this requirement is *partially satisfied*.

SR-5-X-D6 - Solver Compatibility
Generated geometries shall be compatible with the integrated aerodynamic solver and therefore be surface-meshable.

As part of the geometric verification process, an unstructured surface mesh is generated on the outer mould line of the final geometry (section 8.1.1). All manually generated designs are fully surface-meshable (section 8.1.3). For automatically generated designs, approximately 22.5% of failure cases are attributed to unsuccessful surface meshing. In these instances, the geometry itself is successfully generated and passes all geometric validity checks, but the meshing algorithm fails to converge. This leads to the requirement being *largely satisfied*.

SR-5-X-D6.1 - Automatic Meshing
The tool shall include an automatic surface meshing module capable of exporting meshes.

WIN-Gen includes a dedicated meshing computational module (section 4.3.3) capable of automatically generating and exporting unstructured surface meshes for all designs. The convergence behaviour and robustness of the meshing process are discussed in detail in section 8.1.3. This requirement is therefore *fully satisfied*.

SR-6 - Design Interaction
Design generation and modification shall be intuitive and straightforward for the user.

This requirement is defined by SR-6.1 - Initialization Simplicity and SR-6.2 - User Simplicity, which are assessed as *marginally* and *fully* satisfied, respectively. As a result, SR-6 is considered *largely satisfied*, making design generation and modification intuitive and straightforward.

SR-6.1 - Initialization Simplicity
Model initialization shall be easy and intuitive.

This requirement is addressed through the derived requirement section E.3.1 and is therefore assessed accordingly. In its current form, model initialization is not yet easy or intuitive, as it relies on Python scripting with manually specified inputs and limited automated data import. Consequently, this requirement is *marginally satisfied*.

SR-6.1-D7 - Initialization Module
The model shall include initialization modules capable of creating a design from aircraft design data or reference geometry.

This requirement is composed of SR-6.1-D7.1 - CPACS Input, SR-6.1-D7.2 - Reference Geometry Input, and SR-6.1-D7.3 - Coordinate Data Input. The first two are not satisfied, while the latter is fully satisfied, resulting in an overall assessment of *marginal satisfaction*.

SR-6.1-D7.1 - CPACS Input
The tool shall initialize a design from CPACS-formatted data.

WIN-Gen does not currently include a CPACS import capability. As a result, model initialization from CPACS-formatted data is not possible, leading to *no satisfaction* of this requirement.

SR-6.1-D7.2 - Reference Geometry Input
The tool shall initialize a design from reference STEP geometry.

WIN-Gen does not include a geometry import and parsing module for reference STEP files. As evidenced by the manual geometry reconstruction performed during the validation study (section 8.2.1 under Rebuilding the Geometry in the Tool), the absence of such functionality represents a significant limitation. Therefore, this requirement is currently *not satisfied*.

SR-6.1-D7.3 - Coordinate Data Input
The tool shall initialize airfoils and other cross-sections and curves from coordinate data.

WIN-Gen includes functionality for importing coordinate data for airfoils and cross-section shape parametrisations (section 5.2.4 and section 6.4.2). These data are automatically transformed into fully parametric CST-based definitions, resulting in *full satisfaction* of this requirement.

SR-6.2 - User Simplicity
Design changes shall be made easily and intuitively.

This requirement is defined by SR-6.2-D8 - Parameter Collection and SR-6.2-D9 - Parameter Data Types, both of which are fully satisfied. Consequently, SR-6.2 is assessed as *fully satisfied*, indicating that design modification within WIN-Gen is straightforward and intuitive once a model has been initialized.

SR-6.2-D8 - Parameter Collection
All relevant model input parameters shall be organized into a single data collection.

By design, all input parameters in WIN-Gen are centralized within the `data` attribute of the model architecture (section 4.1), where the `Descriptors` of all components are aggregated (section 4.2.1). This structure enables straightforward modification of input parameters and clear separation between input data and generated geometry, leading to *full satisfaction* of this requirement.

SR-6.2-D9 - Parameter Data Types
Each input parameter shall use a primitive data type such as float, integer, string, or boolean.

The parametrisation in WIN-Gen is implemented such that all input parameters use primitive data types, predominantly numerical values with a limited number of booleans and strings. This is evident from the parameter definitions listed in table D.1, resulting in *full satisfaction* of this requirement.

SR-7 - Code Documentation
The model code shall include clear documentation explaining its structure and use.

The WIN-Gen source code includes mandatory documentation for all modules, classes, functions, and attributes. This is enforced through version-control practices with mandatory commit checks, ensuring consistent documentation coverage throughout the codebase. This requirement is therefore *fully satisfied*.

SR-8 - Disciplinary Extendability
The tool shall be easily extendable with new disciplinary capabilities.

This requirement is defined by SR-8.1 - Common Interface and SR-8.2 - Interface Compliance, which are assessed as *fully* and *partially* satisfied, respectively. As a result, SR-8 is considered *largely satisfied*. While extending WIN-Gen with new disciplinary models requires some development effort, the framework is explicitly designed to support such extensions.

SR-8.1 - Common Interface
All disciplinary modules within the model shall implement a common interface.

WIN-Gen includes a dedicated `Simulation` interface (section 4.3.4) that defines shared functionality across all disciplinary modules. Any new discipline can be integrated by adhering to this interface, enabling consistent execution, data exchange, and post-processing. This facilitates systematic extension of the framework and leads to *full satisfaction* of this requirement.

SR-8.2 - Interface Compliance
The functionality of each discipline shall operate using a shared interface.

Although the `Simulation` interface provides common functionality across disciplines, substantial portions of the computational logic remain discipline-specific and cannot be fully abstracted into a single shared interface. Consequently, extending WIN-Gen with a new discipline still requires discipline-specific implementation effort, such as data preparation and simulation execution. For this reason, this requirement is *partially satisfied*.

SR-9 - Component Flexibility
Each top-level component shall be defined by a simple class containing the fundamental design rules, unconstrained by lower-level details.

This requirement is addressed through SR-9.1 - Replaceable Logic and is therefore assessed accordingly. Overall, SR-9 is considered *largely satisfied*.

SR-9.1 - Replaceable Logic
Each top-level component shall allow replacement or injection of new rules or parametrisations without modifying the top-level class. - WIN-Gen clearly distinguishes between *abstract*, *configurable parametric*, and *specialised* classes. The relationship between abstract and configurable parametric classes fully adheres to the intent of this requirement, enabled by the introduced *parametrisation-as-composition* concept (section 4.2).

However, the current balance between configurable parametric and specialised classes, approximately 15 to 7, as illustrated in figure 5.1, figure 6.1 and figure 7.1, indicates potential for further improvement. Increasing the proportion of configurable parametric classes would enhance modularity and extensibility. Consequently, this requirement is assessed as *largely satisfied*.

E.3.2. Requirement Satisfaction Matrix

A summary of the assessment of all requirements is provided in the requirement satisfaction matrix in table E.1. Overall, WIN-Gen demonstrates strong compliance with the defined system requirements, with approximately 34% of requirements being *fully satisfied* and a further 34% being *largely satisfied*. An additional 17% of the requirements are *partially satisfied*, while only a small fraction are *marginally satisfied* (7%), *not satisfied* (5%), or *not assessed* (2%).

The most notable limitations are observed in SR-2, where aerodynamic realism due to limited validation remain limited, and SR-6, where model initialization lacks automated and accurate workflows. Furthermore, SR-5 (aerodynamic assessment fidelity) and SR-3 (parameter definition robustness) are identified as key areas for further improvement. Overall, the results indicate a mature and extensible design framework with clearly defined directions for future enhancement.

Table E.1: Requirement satisfaction matrix. NA = not assessed; NS = not satisfied; MS = marginally satisfied; PS = partially satisfied; LS = largely satisfied; FS = fully satisfied.

ID	NA	NS	MS	PS	LS	FS
SR-1					×	
SR-1.1					×	
SR-1.1-D10						×

Continued on next page

ID	NA	NS	MS	PS	LS	FS
SR-1.1-D1					×	
SR-1.2						×
SR-1.2-D2						×
SR-1.2-D2.1						×
SR-1.2-D2.2						×
SR-2				×		
SR-2-D3			×			
SR-2-D4				×		
SR-3				×		
SR-3.1					×	
SR-3.2	×					
SR-3.3					×	
SR-3.4				×		
SR-4					×	
SR-4.1					×	
SR-4.2					×	
SR-4.3						×
SR-5				×		
SR-5.1						×
SR-5-D5				×		
SR-5-X-D6					×	
SR-5-X-D6.1						×
SR-6					×	
SR-6.1			×			
SR-6.1-D7			×			
SR-6.1-D7.1		×				
SR-6.1-D7.2		×				
SR-6.1-D7.3						×
SR-6.2						×
SR-6.2-D8						×
SR-6.2-D9						×
SR-7						×
SR-8					×	
SR-8.1						×
SR-8.2				×		
SR-9					×	
SR-9.1					×	
SR-10					×	