PARAMETRIC OPTIMIZATION OF A WING-FUSELAGE SYSTEM USING A VORTICITY-BASED PANEL SOLVER

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Aerospace Engineering

by

Chino Minorikawa Cruz

December 2023
COMMITTEE MEMBERSHIP

TITLE: Parametric Optimization of a Wing-Fuselage System Using a Vorticity-Based Panel Solver

AUTHOR: Chino Minorikawa Cruz

DATE SUBMITTED: December 2023

COMMITTEE CHAIR: Paulo Iscold, Ph.D.
Professor, Aerospace Engineering

COMMITTEE MEMBER: Aaron Drake, Ph.D.
Professor, Aerospace Engineering

COMMITTEE MEMBER: John Fabijanic
Lecturer, Mechanical Engineering

COMMITTEE MEMBER: Russell Westphal, Ph.D.
Professor, Mechanical Engineering
ABSTRACT

Parametric Optimization of a Wing-Fuselage System Using a Vorticity-Based Panel Solver

Chino Minorikawa Cruz

Aerodynamic topology optimization is a useful tool in the aerodynamic design process, especially when looking for marginal gains within a design. One example is a turboprop racer concept aircraft that is designed with the goal of breaking world speed records. An optimization framework was developed with the intention of later being applied to this design. In the early design stages, the optimization framework must focus on quicker methods of drag estimation, such as a panel codes. The large number of design variables in topology optimization can exponentially increase function evaluations and thus computational cost. A vorticity-based panel solver was proven out for this application to reduce the computational cost while keeping the accuracy of the results similar to that of traditional CFD solvers in conditions without prominent flow separation. The framework developed here includes geometry parameterization, function evaluation scripting, and post-processing, which are all run within the optimization algorithm. The designs used to validate the solver are wing-fuselage systems of various sailplane configurations with existing experimental data. These sailplane designs were also used as the initial geometry to demonstrate the framework. A parametric optimum was found to reduce drag by 9%, but it must be noted that this method does have certain trade offs and limitations.
ACKNOWLEDGMENTS

Special thanks to:

• Professor Iscold, for advising me throughout this process. The lessons I learned span greater than the scope of this thesis.

• My advising committee, including Professor Drake, Professor Westphal, and Professor Fabijanic. Your patience and guidance over the years have been greatly appreciated.

• My parents, Junko and Alex Cruz, for their unwavering support throughout my academic journey.

• My friends, for keeping me sane through all the tough times.
TABLE OF CONTENTS

| LIST OF TABLES                      | viii |
| LIST OF FIGURES                    | ix  |

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Aerodynamic Drag</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Inverse Pressure Gradient Matching</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Methods to Calculate Drag</td>
<td>7</td>
</tr>
<tr>
<td>1.3.1 Empirical Equations</td>
<td>8</td>
</tr>
<tr>
<td>1.3.2 Panel Codes</td>
<td>9</td>
</tr>
<tr>
<td>1.3.3 Computational Fluid Dynamics</td>
<td>9</td>
</tr>
<tr>
<td>1.3.4 Wind Tunnel Testing</td>
<td>10</td>
</tr>
<tr>
<td>1.4 FlightStream</td>
<td>11</td>
</tr>
<tr>
<td>1.5 Aerodynamic Topology Optimization</td>
<td>12</td>
</tr>
<tr>
<td>2 Motivation</td>
<td>15</td>
</tr>
<tr>
<td>2.1 Motivation</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Objective</td>
<td>17</td>
</tr>
<tr>
<td>3 Validation Case</td>
<td>18</td>
</tr>
<tr>
<td>3.1 Background</td>
<td>18</td>
</tr>
<tr>
<td>3.2 Methodology</td>
<td>20</td>
</tr>
<tr>
<td>3.3 Results</td>
<td>27</td>
</tr>
<tr>
<td>3.4 Additional Validation Cases</td>
<td>32</td>
</tr>
<tr>
<td>4 Optimization</td>
<td>34</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>FAI Records of Speed Over a 3km Course for Turboprop Engine Aircraft [8]</td>
</tr>
<tr>
<td>3.1</td>
<td>Parameters for Sailplane Wind Tunnel Test</td>
</tr>
<tr>
<td>3.2</td>
<td>Solver Analysis Options</td>
</tr>
<tr>
<td>4.1</td>
<td>Conic Curve Control Points for Fuselage 1 (mm) [5]</td>
</tr>
<tr>
<td>4.2</td>
<td>Inputs for <code>patternsearch</code> Optimization</td>
</tr>
<tr>
<td>4.3</td>
<td>Optimization Options for <code>patternsearch</code> Algorithm</td>
</tr>
<tr>
<td>4.4</td>
<td>Optimization Results from Cycle 12</td>
</tr>
<tr>
<td>4.5</td>
<td>Optimization Results from Cycle 36</td>
</tr>
<tr>
<td>4.6</td>
<td>Optimization Results from Cycle 37</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Breakdown of Drag with Respect to Aircraft Properties [11]</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Flat Plate Perpendicular to the Flow: Pressure Drag [6]</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Flat Plate Parallel to the Flow: Friction Drag [6]</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Wingtip Vortex Responsible for Induced Drag [13]</td>
<td>4</td>
</tr>
<tr>
<td>1.5 Canopy Position with Respect to Wing Position [4]</td>
<td>6</td>
</tr>
<tr>
<td>1.6 Proper Inverse Pressure Gradient Matching vs. Improper Placement of Pressure Gradients [4]</td>
<td>6</td>
</tr>
<tr>
<td>1.7 Key Takeaways from Mike Arnold’s AR-5 Aircraft [4]</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Turboprop Racer Concept</td>
<td>15</td>
</tr>
<tr>
<td>3.1 8 Wing-Fuselage Sailplane Configurations</td>
<td>19</td>
</tr>
<tr>
<td>3.2 Conic Curve Control Points</td>
<td>21</td>
</tr>
<tr>
<td>3.3 Wind Tunnel Walls in FlightStream</td>
<td>21</td>
</tr>
<tr>
<td>3.4 Uniting Wing and Wind Tunnel Box</td>
<td>22</td>
</tr>
<tr>
<td>3.5 Uniting Wing and Fuselage</td>
<td>22</td>
</tr>
<tr>
<td>3.6 Wing Protruding Through the Tunnel Wall</td>
<td>23</td>
</tr>
<tr>
<td>3.7 Deleting Faces for Half-Span Model</td>
<td>24</td>
</tr>
<tr>
<td>3.8 Projecting Edges to Align with Symmetry Plane</td>
<td>24</td>
</tr>
<tr>
<td>3.9 Trailing Edge and Wake Termination Node Boundary Conditions</td>
<td>25</td>
</tr>
<tr>
<td>3.10 Fuselage 1 Configuration 2 $C_L$ vs $\alpha$</td>
<td>27</td>
</tr>
<tr>
<td>3.11 Fuselage 1 Configuration 2 $C_L$ vs $C_D$</td>
<td>28</td>
</tr>
<tr>
<td>3.12 Fuselage 1 Configuration 2 $C_M$ vs $\alpha$</td>
<td>29</td>
</tr>
</tbody>
</table>
4.22 $C_p$ Along Top and Bottom Surfaces of the Fuselage (Cycle 36) . . . 54
4.23 $C_D$ vs. Iteration from Cycle 36 . . . . . . . . . . . . . . . . . . . . 55
4.24 Initial Geometry from Cycle 37, Iteration 01 . . . . . . . . . . . . . 56
4.25 Optimum Geometry from Cycle 37, Iteration 47 . . . . . . . . . . . 56
4.26 Outlines of Initial and Optimum Configurations (Cycle 37) . . . . 57
4.27 Drag Polar of Initial and Optimum Configurations (Cycle 37) . . . 58
4.28 Lift Curve of Initial and Optimum Configurations (Cycle 37) . . . 58
4.29 Drag Curve of Initial and Optimum Configurations (Cycle 37) . . . 59
4.30 Surface $C_p$ and Streamlines from Cycle 37 Initial Configuration . 60
4.31 Surface $C_p$ and Streamlines from Cycle 37 Optimum Configuration 60
4.32 Surface $C_p$ and Streamlines from Cycle 37 Initial Configuration . 61
4.33 Surface $C_p$ and Streamlines from Cycle 37 Optimum Configuration 61
4.34 $C_p$ Along Top and Bottom Surfaces of the Wing (Cycle 37) . . . 62
4.35 $C_p$ Along Top and Bottom Surfaces of the Fuselage (Cycle 37) . . 62
4.36 $C_D$ vs. Iteration from Cycle 37 . . . . . . . . . . . . . . . . . . . . 63

A.1 $u$ and $v$ Mesh Parameters for the Wing . . . . . . . . . . . . . . . 76
A.2 $u$ and $v$ Mesh Parameters for the Fuselage . . . . . . . . . . . . . 76
A.3 Refined Mesh at the Wing-Fuselage Junction . . . . . . . . . . . . . 77
Chapter 1

INTRODUCTION

In the design of an aircraft, minimizing the aerodynamic drag is oftentimes a primary goal. Whether it is for efficiency, speed, or maneuverability, the aircraft will ultimately benefit from a reduction in drag. Section 1.1 covers a breakdown of drag and introduces the topic of interference drag and its significance. Section 1.2 covers a physical understanding of the flow at the wing-fuselage intersection, as well as applied methods to reduce interference drag through a case study of the AR-5 by Mike Arnold. Various methods to calculate the drag force of an aircraft are introduced in Section 1.3, and the method of choice for this thesis is discussed in Section 1.4. A tool for drag minimization, aerodynamic topology optimization, is introduced in Section 1.5.

1.1 Aerodynamic Drag

Figure 1.1 showcases the different mechanisms of drag and how it can be broken down differently for an aircraft. On a broad scale, drag can be split into pressure and friction drag. Pressure drag is the drag created due to the shape of the aircraft and the airflow around it. Pressures are integrated along the surfaces to determine the force acting on the body. Figure 1.2 shows a flat plate perpendicular to the flow. The drag that this plate is experiencing is mostly pressure drag as it is independent of wall shear.
Figure 1.1: Breakdown of Drag with Respect to Aircraft Properties [11]

Figure 1.2: Flat Plate Perpendicular to the Flow: Pressure Drag [6]
On the other hand, Figure 1.3 shows a flat plate parallel to the flow which has a small pressure drag component due to its small frontal area. Most of the drag for this plate is attributed to skin friction drag, which is the drag due to the shear layer created by viscous effects.

![Boundary layer](image)

**Figure 1.3: Flat Plate Parallel to the Flow: Friction Drag [6]**

The drag force of an aircraft can be broken down differently into 3 main components: parasitic (or profile) drag, induced drag, and wave drag. Parasitic drag can be further broken down into form drag and skin friction drag [1].

Induced drag is the drag component that is associated with lift generation. As the main lifting body on an aircraft, the wing produces the most induced drag. The resultant force on a wing generally has a large lift component but also has a component of drag, which is the induced drag. These wingtip vortices shown in Figure 1.4 are a result of induced drag. The empennage and fuselage also have components of induced drag as a function of the lift they produce but at a much smaller scale.

Interference drag is a component of form drag produced by two or more bodies (i.e. the wing and fuselage) that intersect or are near each other. The drag of a combined wing and fuselage tends to be higher than the sum of the individual drag components in freestream. This increase in drag is said to be the interference drag [10]. Interference
drag is attributed to the interaction of boundary layers of the wing and fuselage subjecting the flow to an adverse pressure gradient. If the adverse pressure gradient created by the two bodies is strong enough, vortices may also shed off the body.

As parasitic drag rises with the square of the velocity, every effort must be made to minimize parasitic drag if the main design goal is speed or efficiency. With a decrease in parasitic drag, the minimum drag condition for an aircraft can occur at a higher airspeed. Often neglected during the early stages of a design, the interference drag of an aircraft can be a significant contributor to the parasitic drag of an aircraft.
1.2 Inverse Pressure Gradient Matching

Understanding pressure gradients and how they play into the aerodynamics of an aircraft can help identify areas of the design that are producing excess drag. An adverse pressure gradient (or positive pressure gradient) signifies an increase in static pressure or a decrease in speed. Consequently, a favorable pressure gradient signifies a decrease in static pressure or an increase in speed. Learning how pressure gradients interact with one another is crucial to understanding interference drag.

An aircraft of interest when specifically looking at interference drag reduction is Mike Arnold’s AR-5 [4]. This single-seat turboprop airplane set the world speed record over a 3 km course in 1992. It was known for being very efficient, with the flat plate drag area for this aircraft being 0.88 sq. ft [2]. The first main takeaway from Mike Arnold’s AR-5 design was the “inverse pressure gradient matching” between the canopy and wing. The canopy was positioned with respect to the wing by lining up the canopy’s starting point with the thickest chord-wise section of the wing, shown in Figure 1.5. The canopy’s peak is also positioned to match the trailing edge of the wing. This allowed for the favorable pressure gradient of the wing to be directly under the adverse pressure gradient of the canopy. More importantly, the adverse pressure gradient aft of the wing would be directly under the favorable pressure gradient of the canopy. This inverse matching of pressure gradients would effectively cancel the gradients out as shown in Figure 1.6. This allows for a smooth transition of pressures in areas of high expansion and contraction of the cross-sectional area.

Note that if the canopy position moves slightly fore or aft from this position, the pressure gradients start to add to each other, steepening the adverse pressure gradient over the trailing edge (Figure 1.6). The amount of interference drag for an aircraft is very sensitive to the placement of these pressure gradients with respect to one
another. Oftentimes, the placement for the wing and canopy happen during the early stages of a design when interference drag is not a main consideration. An effort must be made to locate these components in a position where the pressure gradients work well with one another for the design to have a competitive edge over other aircraft.

![Figure 1.5: Canopy Position with Respect to Wing Position [4]](image)

The second takeaway from Arnold’s design was to keep the fuselage sides parallel at the wing intersection. This was done to avoid contracting the fuselage sides at the location of the adverse pressure gradient from the trailing edge of the wing. Reducing the cross-sectional area of the fuselage too early would further decelerate the flow, which would add to the adverse pressure gradient on the aft part of the wing.

![Figure 1.6: Proper Inverse Pressure Gradient Matching vs. Improper Placement of Pressure Gradients [4]](image)
The third takeaway from Arnold’s design is to use expanding radius fairings on the wing-fuselage intersection. Even with parallel fuselage sides, the skin friction will slow the air, adding to the adverse pressure gradient. The adverse gradient from the trailing edge of the wing local to the junction can be further counteracted by these fairings, keeping the interference drag at a minimum. Additionally, they would act as a smooth transition between the nearly perpendicular walls, working to mitigate any large vortex formations. These features from the AR-5 are shown in Figure 1.7 below.

![Figure 1.7: Key Takeaways from Mike Arnold’s AR-5 Aircraft](image)

1.3 Methods to Calculate Drag

To create an efficient design, there first needs to be a method to compute the aerodynamic loads on the design. Depending on the level of accuracy needed at varying
stages of the design, several methods are employed to calculate the aerodynamic performance of the aircraft. This can include empirical equations, computational fluid dynamics, or experimental measurements.

1.3.1 Empirical Equations

The most basic, low-fidelity method of calculating drag comes from the drag equation. This method of drag calculation is most often used early in the design process when many variables are rapidly changing and fast calculations must be made to quickly estimate the drag in a system for other calculations. For an aircraft, the simplest form of the drag equation is

\[ D = \frac{1}{2} \rho V^2 S_{ref} C_D \]  

(1.1)

where \( \rho \) is the density, \( V \) is the freestream velocity, \( S_{ref} \) is the reference area (typically wing planform area), and \( C_D \) is the drag coefficient. The drag coefficient is given by the equation

\[ C_D = C_{D0} + C_{Di} \]  

(1.2)

where

\[ C_{Di} = \frac{C_L^2}{\pi AR e} \]  

(1.3)

\( C_{D0} \) is the parasitic drag component and can be found through empirical methods. \( C_{Di} \) is the induced drag component, which consists of the lift coefficient, \( C_L \), the wing aspect ratio \( AR \), and the Oswald efficiency factor, \( e \). This equation can also be modified to incorporate terms from the wing geometry such as twist and sweep, which increases the fidelity of the drag estimation.

The drag buildup method is another empirical method that takes the drag value from individual components of the aircraft and combines them into a total drag value. The
drag produced from each component (i.e., wing, fuselage, empennage, engine nacelle, etc.) is added, while also taking into consideration interference drag and trim drag. Additionally, drag from gaps, excrescences, landing gear, and high lift devices may also be included. Note that when adding $C_D$ values, the $S_{ref}$ must be the same for all different sources. The drag buildup method allows for the calculation of drag in different aircraft configurations, such as takeoff, cruise, and landing, lending itself useful in the earliest stages of an aircraft’s design when performing trade-off analyses.

1.3.2 Panel Codes

For higher fidelity methods to drag estimation, numerical methods must be employed. These methods are typically implemented in the stages of the design process when the surface geometry is more defined. One of the simplest numerical methods to calculate aerodynamic loads is through panel codes. Panel codes, in a general sense, calculate the pressure distribution around an object using sources, sinks, vortex points, and doublets to model the surface mesh (panels) of a geometry [16]. Different approximations, discretization methods, boundary conditions, and methods to calculate pressure are used to develop a wide array of panel codes that calculate the aerodynamic loads of a vehicle.

1.3.3 Computational Fluid Dynamics

The standard in numerical drag calculation methods is Computational Fluid Dynamics, or CFD. Unlike panel codes, CFD discretizes the volume around a surface mesh and calculates the properties of fluids around an object by solving the governing equations of fluid dynamics. An example of a CFD code is Reynolds-Averaged Navier-Stokes (RANS). RANS codes take the Navier-Stokes equations and provides
a time-averaged solution to model turbulence [17]. Besides RANS, there are multitudes of other CFD codes that employ different assumptions, discretization methods, and turbulence models to come up with a solution. With the wide array of CFD codes, there is a tradeoff between computational cost and accuracy. With this type of method, its computational cost is much higher than the previously mentioned methods. From setting up the model, running the solver, and post-processing the results, CFD software is more costly than a panel code. As a general order of magnitude, CFD will take hours to days for a solution, while panel codes generally take seconds to minutes.

1.3.4 Wind Tunnel Testing

Another method to calculate drag is through wind tunnel testing. Typically, a scale model is used to calculate the loads of a system, but full-scale models may also be tested for later-stage designs. The aerodynamic loads are non-dimensionalized using dynamic pressures and reference dimensions. The scale model loads can be used to correlate to the real system as long as the Reynolds number is matched. This method requires a physical model to be manufactured and then tested at a facility. This method for aerodynamic load calculation can be relatively expensive, especially when many geometric changes must be made. This makes it difficult for wind tunnel testing to be used for parametric studies or topology optimization. However, when considering the costs associated with a CFD simulation, wind tunnel testing receives an edge when having to take measurements in different flowfield conditions. Wind tunnel testing is often performed in the later stages of the design process when the design geometry is mostly solidified. Wind tunnel testing, when employed in conjunction with CFD, can be the gateway to correlating results with the real world. This
helps with adding confidence to the results and validating the other methods utilized during the earlier stages of the design process.

1.4 FlightStream

The panel code of interest for this thesis is FlightStream, which uses a surface vorticity solver instead of the typical pressure solver of a more conventional panel code. Viscous effects are included in this vorticity solver using boundary integral methods. This panel code is developed by Research in Flight and is claimed to produce solutions in a fraction of the time taken by conventional CFD solvers without sacrificing accuracy.

A study from FH Aachen University of Applied Sciences and RMIT University [15] analyzing highly non-planar aircraft configurations compared different low-order methods (VSPAERO and FlightStream) against flight test data and RANS CFD solver STAR-CCM+. It concluded that FlightStream is useful for predicting aerodynamic loads in situations without prominent flow separation effects and could be well-suited for studies where many different configurations must be analyzed.

The primary reason for choosing a panel code for this optimization was computational cost. Especially during the early design stages, there is a lot more room to play within the design space. Although this framework is developed for a parametric model with a handful of design variables, using a panel code leaves the option to quickly scale the scope of the optimization problem if desired. If FlightStream can prove robust enough to be used inside an optimization algorithm, the computational cost saved can be significant when compared to higher-fidelity solvers.

A validation study for FlightStream is performed in Chapter 3 to ensure that the aerodynamic loads calculated are accurate enough to capture subtle differences within
the design space. The ability of the solver to properly calculate the deltas between
the small changes in geometry is of utmost importance when implementing it into an
optimization environment.

1.5 Aerodynamic Topology Optimization

To achieve the most efficient design for a given configuration, optimization methods
must be employed to gain every last bit of performance. In general, an optimization
algorithm aims to minimize or maximize an objective function given a set of con-
straints. In aerodynamic topology optimization, the objective function is usually the
drag coefficient. However, other objective functions such as performance, weight, or
shock wave formations could also be included. While trying to solve for the objective
function, there are a set of constraints such as internal volume or lift coefficient that
help define the design space. The geometry is parameterized and design variables are
used as inputs to the objective function. When considering topology optimization,
these design variables can include points on a mesh, curves, splines, and 3-D surfaces.
The optimization algorithm then works to minimize (or maximize) the objective func-
tion, using different iterative schemes to find an optimum solution.

In some optimization algorithms, a gradient is calculated for the objective function.
This gradient helps guide the fuselage geometry toward a solution that best minimizes
the objective function. Other algorithms, such as the Genetic Algorithm (GA), do not
incorporate gradients and instead involve a process similar to natural selection that
chooses the best solution sets to create the next generation of solutions. Other non-
gradient-based optimization algorithms include a pattern search, where the objective
function is calculated for design variables in a specific pattern and the best solution
is chosen as the starting point of the next pattern. This method iterates through sets of patterns until it converges onto an optimum solution.

In aerodynamic topology optimization, calculating the objective function is relatively costly. Each objective function calculation involves obtaining a full solution to solve for aerodynamic loads. When choosing an optimization algorithm, the total number of objective function calculations must be balanced with the desired accuracy of the optimized solution to ensure that a solution can be found within a feasible time frame.

A study was performed at Linköping University [3] that employed a panel code in an aircraft concept optimization comparing 3 separate optimization algorithms. The panel code was used for performance prediction which would feed into the objective function calculation. The objective function included maximum takeoff weight along with other performance characteristics. The optimization algorithms considered were Fmincon (gradient-based), Complex, and a genetic algorithm. The gradient-based algorithm, Fmincon, is part of the MATLAB Optimization Toolbox and finds the minimum of a constrained nonlinear multivariate function. This algorithm was unable to solve the problem as it seemed to reach a local optimum that performed worse than the designs that the non-gradient-based methods produced. Complex, a variation to the Simplex algorithm, and GA were able to find the global optimum with a hit rate of 17% and 75% respectively. Although the Complex algorithm took less time to come up with a solution, the difference in accuracy made the GA the more favorable algorithm for this specific problem [3].

This thesis employs another algorithm from the MATLAB Optimization Toolbox, \textit{patternsearch}. During the initial stages of creating the optimization framework, the pattern search algorithm was inconsistent in finding the optimum for the design problem. The success of the search algorithm would be heavily dependent on the options chosen for the algorithm beforehand. The non-linear nature of the problem would
confuse the algorithm into choosing a local optimum versus the global optimum for this problem. As a fix, the genetic algorithm was used as a search function before the pattern search kicked in. This was successful as the parametric optimum for this problem was more consistently found by the algorithm. This would allow for the optimization algorithm to use GA to search on a more global scale, while the pattern search would be used to find the optimum on a more local scale relative to the optimum found by the GA.
Chapter 2

MOTIVATION

2.1 Motivation

The initial motivation for this thesis was a next-generation turboprop racer concept design led by Dr. Paulo Iscold. As shown in Figure 2.1, this aircraft is designed with the goal of setting world records for speed. The world governing body for maintaining records of aeronautical activities is the Fédération Aéronautique Internationale (FAI), or World Air Sports Federation. The list of speed records for powered aircraft over a 3 km course is recorded by the FAI, shown in Table 2.1. The sub-class defines the propulsion type, along with the maximum takeoff weight of the aircraft. The current speed records for this sub-class of aircraft ranges from 430-570 km/h or around 269-354 mph.
Table 2.1: FAI Records of Speed Over a 3km Course for Turboprop Engine Aircraft [8]

<table>
<thead>
<tr>
<th>Sub-Class</th>
<th>Performance</th>
<th>Date</th>
<th>Claimant</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-1d</td>
<td>449.71 km/h</td>
<td>08 Oct 2005</td>
<td>Richard L. Kane (USA)</td>
</tr>
<tr>
<td>C-1e</td>
<td>433.19 km/h</td>
<td>08 Oct 2005</td>
<td>Richard L. Kane (USA)</td>
</tr>
<tr>
<td>C-1c</td>
<td>570.33 km/h</td>
<td>11 Sep 2002</td>
<td>Wesley E. Behel Jr. (USA)</td>
</tr>
</tbody>
</table>

To design a world-class aircraft for speed records, minimizing drag is a top priority. Every count of drag saved could be potential performance gains for the acceleration and top speeds within a specific engine class. For this reason, even a relatively small portion of the total drag (such as interference drag) must be minimized. Through proper aerodynamic design fundamentals, an aircraft can be made relatively slippery. The implementation of topology optimization could further push a design to gain every last bit of performance, giving it an edge against competitors. Optimization algorithms are a tool that an aerodynamicist can use to improve a design and should not be solely relied on to produce a design. With the combination of proper aerodynamic design considerations and aerodynamic topology optimization, the best possible design can then be achieved.

During the concept design of an aircraft, many aspects of the design are constantly changing. An optimization algorithm at this stage will require large-scale geometry changes that employ many design variables. Oftentimes, it may not be feasible to use a full CFD solver due to computational cost. A panel code, such as FlightStream, may prove useful given its balance of computational cost and accuracy.
2.2 Objective

Aerodynamic shape optimization has become an indispensable tool for any effective design today. The objective of this thesis is to design a framework for the aerodynamic topology optimization of a wing-fuselage system using a vorticity-based panel solver. The optimization framework will initially be developed for the validation case of a sailplane experiment conducted by Delft University [5]. After validating the solver with experimental results, the same geometry will be used as a starting point to develop the optimization algorithm and produce an optimized result. This thesis will aim to study the feasibility, complexities, and limitations of this optimization strategy and determine if a similar algorithm can be developed using FlightStream for future use on the next-gen turboprop racer concept design.
Chapter 3

VALIDATION CASE

Chapter 3 covers the validation case for this thesis, which will also be used as the initial geometry for the optimization. The solver of interest, FlightStream, will be validated against experimental data to ensure that it can be implemented in an optimization environment to produce accurate results. A mesh convergence study was performed for the sailplane geometry to ensure that the results were mostly independent of the mesh size. In addition to this validation study, the last section of this chapter covers various validation cases from other sources.

3.1 Background

The validation case for this thesis is based on a study published by Boermans and Terleth [5] at the Delft University of Technology. This study was performed with the goal of gaining insight into the interference drag phenomena, as well as gathering experimental data for sailplane performance estimation studies. The experiment was performed in a low-speed, low-turbulence wind tunnel. The models tested consisted of eight different wing-fuselage combinations shown in Figure 3.1. Fuselage 1 is the base configuration, using the fuselage of well-known sailplanes ASW-19 and ASW-20. Fuselage 2 and 3 differ by their contraction ratio behind the location of maximum thickness. The model tested on the wind tunnel was reduced to a 1:3 scale.
Figure 3.1: 8 Wing-Fuselage Sailplane Configurations

The low-speed, low-turbulence wind tunnel is a closed loop with an octagonal test section 1.80 m wide and 1.25 m high. The model was rotated about the 40% chord line, and connected to a six-component balance system to gather load data. Drag measurements were also performed using a wake rake downstream of the model. Flow visualization was performed with oil film to determine the location of the boundary layer transition. The model and test parameters are given in Table 3.1. Lift, drag, and moment data were provided for several configurations and will be used to validate FlighStream for use in the optimization framework.
Table 3.1: Parameters for Sailplane Wind Tunnel Test

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reynolds Number, $Re$</td>
<td>$1.23 \times 10^6$</td>
<td>-</td>
</tr>
<tr>
<td>Airfoil</td>
<td>FX62-K-131/17</td>
<td>-</td>
</tr>
<tr>
<td>Wing Chord, $c$</td>
<td>0.30</td>
<td>$m$</td>
</tr>
<tr>
<td>Wingspan, $b$</td>
<td>1.80</td>
<td>$m$</td>
</tr>
<tr>
<td>Wing Area, $S$</td>
<td>0.54</td>
<td>$m^2$</td>
</tr>
<tr>
<td>Velocity, $U$</td>
<td>60.58</td>
<td>$m/s$</td>
</tr>
<tr>
<td>Turbulence Level</td>
<td>0.4</td>
<td>$%$</td>
</tr>
</tbody>
</table>

3.2 Methodology

The geometry for this validation case is given in the research paper as a list of coordinates for each fuselage. The coordinates correspond to cross sections of the fuselage, defined by conic curve control points. The geometry is defined through an upper and lower conic curve, defined by the parameters given in Figure 3.2. The wing positioning was defined by the leading edge $x$ and $z$ positions given for each configuration. A MATLAB script was created to convert these conic curves into a surface mesh that is imported into the panel code as a table.
Additionally, the presence of the wind tunnel walls is necessary to obtain results that matched the experiment in FlightStream. The wingtip effects of the aircraft in freestream would deviate from the results given in the wind tunnel, which had little to no wingtip effects due to the model spanning the entire test section. The height and width of the tunnel walls are modeled to match the experiment, and the length of the tunnel walls are 2 aircraft lengths upstream and 5 aircraft lengths downstream (shown in Figure 3.3).
The wind tunnel geometry is generated in FlightStream as a box with the front and rear faces removed. Only the walls of the test section are modeled to simulate the wind tunnel. The three separate bodies (fuselage, wing, and box) are united using the functionality provided within FlightStream using OpenVSP, shown in Figures 3.4 and 3.5. When creating the tunnel wall box, the wall must fully intersect with the wingtip to be able to unite the wing with the wind tunnel wall. On the outer surface of the wall, the portion of the wing that was intersecting must be deleted, shown in Figure 3.6. A quick mesh convergence study was performed in Appendix A to determine the appropriate mesh size for the wind tunnel walls.

Figure 3.4: Uniting Wing and Wind Tunnel Box

Figure 3.5: Uniting Wing and Fuselage
To simulate multiple angles of attack in the wind tunnel, the aircraft will be rotated with respect to the wind tunnel walls along the 40% chord line. This means that the freestream angle will not be changed in order to simulate the freestream parallel to the tunnel walls. This disables the use of the angle of attack sweep functionality and a separate simulation must be performed for each angle. The walls will stay at $0^\circ$ axis and the flowfield will also be set to $0^\circ$.

The model is split in half to save computational time by using a symmetry boundary condition. This is done using Topology-Threshold Selection and selecting all $Y$-values above 0 (sometimes a small negative value such as -0.0001 must be used to select all relevant faces to be deleted). Figure 3.7 shows the geometry after deleting for symmetry.
After deletion, some edges from the deleted faces may not be aligned with the symmetry plane. Under Topology - Plane-Based Operations, project all free edges to the XZ plane (symmetry plane). Figure 3.8 shows the edges to be projected onto the symmetry plane, including the wind tunnel walls. This ensures that the symmetry will not cause any errors in the simulation.

In FlightStream, the boundary conditions of interest are the trailing edges of the lifting surfaces and the wake termination nodes. These conditions are shown in red in
Figure 3.9. The auto-detect feature may define the edges of the tunnel as a trailing edge, which should be fixed by removing the physics at that boundary. A wake termination node must appear for both the wing-fuselage intersection and the wing-wall intersection. If the local mesh size at the wing-wall intersection is too large, the proximity sizing of the wing and wall mesh may cause a wake strand to appear outside of the tunnel boundaries. As a fix, a wake disable node (in orange) may be placed on the wake node adjacent to the wake termination node at the wing-wall intersection. A more robust fix would be to refine the mesh size of the tunnel wall around this area.

Figure 3.9: Trailing Edge and Wake Termination Node Boundary Conditions
The last step in the setup is to set the simulation properties and settings. Standard atmospheric air properties at sea level were used for these simulations. The steady solver was set using the values defined in Table 3.1. The solver was initialized using the default calculation for wake termination $x$-location, a symmetry plane with a periodicity of 1, proximity avoidance enabled, stabilization enabled, and fast multipole disabled. The analysis options were set according to Table 3.2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lift Model</td>
<td>Pressure</td>
</tr>
<tr>
<td>Drag Model</td>
<td>Vorticity</td>
</tr>
<tr>
<td>Moment Model</td>
<td>Linear Pressure</td>
</tr>
<tr>
<td>BL Type</td>
<td>Trans. Turbulent</td>
</tr>
<tr>
<td>BL Drag</td>
<td>Momentum Integral</td>
</tr>
<tr>
<td>Surface Roughness Height</td>
<td>0.0</td>
</tr>
<tr>
<td>Flow Separation</td>
<td>Disable</td>
</tr>
</tbody>
</table>

Table 3.2: Solver Analysis Options

The solver is now initialized and the simulation is ready to run. The convergence criteria are set to $10^{-5}$, determined by a point in the simulation with converged lift, drag, and moment coefficients. The case was run for a sweep of angles of attack from $-4^\circ$ to $4^\circ$. This range was chosen based on the linear section of the lift curve since stall effects are not of interest in this thesis.

This validation study is performed for two configurations of the sailplane experiment. The first case is Fuselage 1 Configuration 2 (F1C2) and the second case is Fuselage 3 Configuration 3 (F3C3). This was done to ensure that the panel code could accurately capture the changes in drag with respect to a given fuselage shape and wing configuration. Fuselage 1 acts as the starting point for an optimization case, and Fuselage 3 acts as the optimum case. If the panel code can accurately represent the relative differences in lift and drag for these configurations, it can then be used to
calculate the objective function in the optimization algorithm with enough accuracy
to ensure convergence to an optimum.

### 3.3 Results

Figure 3.10 shows the FlightStream results plotted against experimental data for the
lift coefficient of F1C2. The $\alpha$ value given by the experiment is determined to be the wing angle of attack and is important as there is a wing incidence of 1°. The lift coefficient results show good agreement with the experimental data.

![Figure 3.10: Fuselage 1 Configuration 2 $C_L$ vs $\alpha$](image)

Figure 3.11 shows the FlightStream results plotted against experimental data for the drag coefficient of F1C2. Again, the results from FlightStream show good correspondence with the experiment. However, it must be noted that the $C_D$ considered here is only the $C_{D0}$ component from the FlightStream results. This is because the Flight-
Stream solver is not currently able to properly model the induced drag, $C_{Di}$, with the tunnel wall interactions. Future versions of FlightStream may include the provision to properly model wind tunnel walls and their interactions. For this validation, it is safe to assume that the induced drag component is negligible as the wing spans the entire test section, essentially eliminating any wingtip effects that would normally form in an aircraft in freestream flow.

Figure 3.11: Fuselage 1 Configuration 2 $C_L$ vs $C_D$

Figure 3.12 shows the FlightStream results plotted against the experimental data for the pitching moment coefficient of F1C2. These results show a slight offset from the experimental data. This shift in the moment curve could be due to the addition of the incorrect induced drag component. The center of pressure calculated with this induced drag component may be shifted farther aft than in the experiment, which would account for the larger moment arm. The linear trend in the pitching moment seems to match that of the experiment but at a different magnitude. The pitching
moment will not be used for the optimization algorithm’s objective function so this discrepancy is not of concern.

Figure 3.12: Fuselage 1 Configuration 2 $C_M$ vs $\alpha$

Figure 3.13 shows the FlightStream results plotted against experimental data for the lift coefficient of F3C3. Similar to F1C2, the lift curve has good agreement with the experimental data. This is expected as the lift curve is highly dependent on the wing, which is kept constant for both of these configurations.
Figure 3.13: Fuselage 3 Configuration 3 $C_L$ vs $\alpha$

Figure 3.14 shows the FlightStream results plotted against experimental data for the drag coefficient of F3C3. The simulation results have a fair agreement with the experimental results. The FlightStream results seem to deviate more as the angle of attack increases, which could be explained by the induced drag component being omitted. The induced drag component calculated on FlightStream would be too large to account for this offset, but there may have been some induced drag component in the experiment due to the fuselage shape, or perhaps even the wing-fuselage interference effects for this specific configuration.
In the context of an optimization algorithm, the important parameter is the $C_D$ at a constant $C_L$. Furthermore, the $\Delta C_D$ between two fuselage configurations is what must be validated to ensure that the optimization algorithm can converge onto an optimum. Figure 3.15 shows the drag polar for both Fuselage 1 Configuration 2 and Fuselage 3 Configuration 3 plotted against each other. As shown, the drag deltas between the two cases are properly captured along the linear portion of the polar.
3.4 Additional Validation Cases

The induced drag component for the sailplane case is unable to be validated due to the tunnel wall interactions in the experiment. However, the optimization algorithm will calculate the objective function with the sailplane in freestream without tunnel walls. Additional case studies are provided to ensure that FlightStream can produce accurate results for a wide variety of applications.

Several case studies were performed for this solver, including the NASA Energy Efficient Transport (E.E.T.) AR-12 and the NASA N2A Hybrid Wing Body aircraft. The NASA E.E.T. AR-12 is an independent study [14] performed by NASA Langley to validate FlightStream in the conditions of cruise, takeoff, and landing. The NASA N2A Hybrid Wing Body aircraft is a validation study [7] published by Auburn University and Research in Flight to compare the solver to experimental data and so-

Figure 3.15: $C_D$ for F1C2 vs F3C3
olutions from an Euler solver. Both case studies showed that there was good agreement between the solver and experimental data in the linear portion of the lift curve.
In this thesis, parametric topology optimization will be performed to take the initial geometry and find the optimum that minimizes drag for a given set of conditions and constraints. The geometry parametrization, optimization framework, and results are presented in this chapter.

4.1 Geometry Parametrization

The initial geometry chosen for this optimization problem is Fuselage 1 Configuration 2, shown in Figure 4.1. The geometry is defined as an array of conic curve control points, defined in Table 4.1. $X_f$ is the fuselage conic curve $x$-location (defines fuselage stations), $Y_m$ is the middle conic curve control point $y$-location (defines width), and $Z_u$, $Z_m$, and $Z_l$ are the upper, middle, and lower conic curve control point locations in $z$ (defines height and mid-point). The fuselage curves are generated as ellipses using these conic control points. Figure 3.2 shows how the control points come together to define a fuselage station. The wing position is defined using the leading edge location at $x$ and $z$. The initial wing position is set at $X_f = 665$ and $Z_f = 72$. All units defined are in millimeters.
To reduce the scope of this problem, a smaller section of the fuselage will be modified in the optimization. The fuselage section downstream of the location of maximum thickness will be used for the parameterization. This helps to limit the scope of the optimization problem to the fuselage contraction. The upper conic curve control points, $Z_u$, will be used as the independent design variables for the optimization, further parameterized using a cubic polynomial. To save on computational time for the optimization algorithm, the number of design variables is reduced to the two inner points that define the cubic polynomial, while the two outer points are kept constant and define the slope at the endpoints.

Figure 4.2 shows how four $Z_u$ points (in bold in Table 4.1) are chosen to create the cubic polynomial. The two edge points are held constant in location and slope, while the two inner points work to define the inflection points of the curve. In this thesis,
the \( x \)-locations of the inflection points will stay constant, while the \( z \)-positions of these points will be the design variables. This cubic polynomial is used to define the rest of the \( Z_u \) points, or the roof line of the fuselage. The other conic control points, \( Z_l \) and \( Y_m \), are assumed to be a function of these \( Z_u \) points, using ratios calculated at each station of the original geometry. Figure 4.3 shows how the vertical ratio between \( a \) and \( b \) are defined, as well as the horizontal ratios between \( a \) and \( c \).

![Figure 4.2: Cubic Polynomial for \( Z_u \), Defined by 4 Points](image)

Throughout the optimization, the wing will be kept at a constant \( x \)-location, but the \( z \)-position must be adjusted to ensure that the wing root is constrained within the fuselage geometry. The wing \( z \)-position is set in the geometry parametrization scheme to shift so that the distance from the upper surface of the wing is a defined minimum distance under the \( Z_u \) curve of the fuselage. Figure 4.4 shows a shifted
fuselage position, which in turn shifts the wing position so that the wing is set to be 7 mm lower than the fuselage.

Figure 4.3: Conic Curve Ratio Definition
Although this method of parametrization will not produce the \textit{global optimum} for drag, this would allow the algorithm to calculate the \textit{parametric optimum} of the contraction ratio for the given parameterization method. In future work, the scope of this problem can be expanded upon to optimize more design variables such as the $x$-variable in the cubic inflection points, vertical and horizontal contraction ratios, $Z_m$ locations, and variables for the wing. Furthermore, design constraints can also be added (such as minimum fuselage volume) to ensure the optimization produces a feasible design.
Table 4.1: Conic Curve Control Points for Fuselage 1 (mm) [5]

<table>
<thead>
<tr>
<th>( X_f )</th>
<th>( Y_m )</th>
<th>( Z_u )</th>
<th>( Z_m )</th>
<th>( Z_l )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-53</td>
<td>-53</td>
<td>-53</td>
</tr>
<tr>
<td>60</td>
<td>30</td>
<td>-12</td>
<td>-58</td>
<td>-82</td>
</tr>
<tr>
<td>120</td>
<td>47</td>
<td>20</td>
<td>-60</td>
<td>-102</td>
</tr>
<tr>
<td>220</td>
<td>71</td>
<td>60</td>
<td>-62</td>
<td>-126</td>
</tr>
<tr>
<td>320</td>
<td>89</td>
<td>90</td>
<td>-61</td>
<td>-142</td>
</tr>
<tr>
<td>420</td>
<td>102</td>
<td>111</td>
<td>-59</td>
<td>-152</td>
</tr>
<tr>
<td>520</td>
<td>107</td>
<td>121</td>
<td>-57</td>
<td>-153</td>
</tr>
<tr>
<td><strong>620</strong></td>
<td>106</td>
<td><strong>121</strong></td>
<td>-51</td>
<td>-146</td>
</tr>
<tr>
<td>720</td>
<td>101</td>
<td>115</td>
<td>-45</td>
<td>-135</td>
</tr>
<tr>
<td><strong>820</strong></td>
<td>94</td>
<td><strong>107</strong></td>
<td>-37</td>
<td>-120</td>
</tr>
<tr>
<td>920</td>
<td>85</td>
<td>97</td>
<td>-28</td>
<td>-106</td>
</tr>
<tr>
<td><strong>1020</strong></td>
<td>75</td>
<td><strong>87</strong></td>
<td>-19</td>
<td>-92</td>
</tr>
<tr>
<td>1120</td>
<td>67</td>
<td>77</td>
<td>-11</td>
<td>-80</td>
</tr>
<tr>
<td>1220</td>
<td>60</td>
<td>67</td>
<td>-4</td>
<td>-68</td>
</tr>
<tr>
<td>1320</td>
<td>54</td>
<td>58</td>
<td>-1</td>
<td>-59</td>
</tr>
<tr>
<td><strong>1420</strong></td>
<td>50</td>
<td><strong>51</strong></td>
<td>0</td>
<td>-51</td>
</tr>
<tr>
<td>2000</td>
<td>29</td>
<td>29</td>
<td>0</td>
<td>-29</td>
</tr>
<tr>
<td>2200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

4.2 Optimization Framework

The framework for this optimization problem was developed in MATLAB, a numeric computing environment. Figure 4.5 shows a high-level structure of the code developed for this thesis. The optimization algorithm used in this thesis is *patternsearch*, which is included in MATLAB’s Global Optimization Toolbox. Reference [12] describes in detail how *patternsearch* polling works, along with explanations of the solver options. This algorithm was chosen because of its relative simplicity and speed in finding a solution, especially when constrained to finding the parametric optimum for a small problem such as this one. As the panel code used to obtain the objective function value also needs to be proven out for this application, a relatively simple optimization algorithm was used to speed up the process. If needed, another algorithm (such as the
genetic algorithm) can be used in addition to patternsearch to achieve a more robust representation of the optimum more frequently. This can be done within patternsearch by adding a ”search function” on top of the default patternsearch polling. However, it must be noted that computational resources can be constrained as other algorithms of higher complexity may require more function calculations to achieve an optimum.

The optimization script starts by defining the initial geometry using the conic control points defined previously. The two $Z_u$ values that define the cubic polynomial are used to define the initial design variables for the patternsearch optimization. The design variables, the lower and upper bounds, and the optimization options are used to run the patternsearch algorithm. Additionally, the objective function calculation is performed by 'dragCalcFS', which is the main function that is called by patternsearch. 'dragCalcFS' takes the design variable inputs and outputs a drag value. In summary, this function takes the two design variables, generates the geometry using the parameterization method, writes the script that runs the FlightStream simulation, runs the

Figure 4.5: Optimization Framework in MATLAB Environment
FlightStream simulation, then reads and outputs the function value for that iteration. The *patternsearch* algorithm calls 'dragCalcFS' for each iteration that it performs. Once the stopping criteria set for the algorithm are met, the optimization script stops running.

4.2.1 *patternsearch*

*patternsearch* takes a variety of inputs including design variables, objective functions, constraints, and options for the optimization algorithm. The function inputs used for this problem are shown in Table 4.2 and 4.3. The objective function is 'dragCalcFS', which is described in more detail in Section 4.2.2. The initial values are set to be the conic curve points that define F1C2. The upper and lower bounds are also set for this problem, with the upper bound being the maximum $Z_u$ value of the initial geometry. The lower bound is set as the $Z_u$ value that defines the end of the cubic polynomial. The optimization options are defined for this problem using a systematic approach to produce a consistent result. A linear inequality constraint was set for this algorithm to reduce the likelihood of an odd result. The linear constraint stated that $x_2$ shall be less than or equal to $x_1$ for any given iteration. Nonlinear constraints were not used for this optimization problem.
Table 4.2: Inputs for patternsearch Optimization

<table>
<thead>
<tr>
<th>Input</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fun</td>
<td>@dragCalcFS</td>
<td>Objective Function Handle</td>
</tr>
<tr>
<td>x0</td>
<td>[107,87]</td>
<td>Initial Geometry</td>
</tr>
<tr>
<td>A</td>
<td>[-1,1]</td>
<td>Linear Inequality</td>
</tr>
<tr>
<td>b</td>
<td>[0]</td>
<td>Linear Inequality</td>
</tr>
<tr>
<td>Aeq</td>
<td>[-]</td>
<td>Linear Equality</td>
</tr>
<tr>
<td>beq</td>
<td>[-]</td>
<td>Linear Equality</td>
</tr>
<tr>
<td>lb</td>
<td>[67,52]</td>
<td>Lower Bounds</td>
</tr>
<tr>
<td>ub</td>
<td>[121,107]</td>
<td>Upper Bounds</td>
</tr>
<tr>
<td>nonlcon</td>
<td>[-]</td>
<td>Non-linear Constraints</td>
</tr>
<tr>
<td>options</td>
<td>Table 4.3</td>
<td>Optimization Options</td>
</tr>
</tbody>
</table>

Table 4.3: Optimization Options for patternsearch Algorithm

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>classic</td>
<td>Algorithm used by patternsearch</td>
</tr>
<tr>
<td>Display</td>
<td>diagnose</td>
<td>Information displayed onto Command Line</td>
</tr>
<tr>
<td>MeshTolerance</td>
<td>1</td>
<td>Tolerance on mesh size</td>
</tr>
<tr>
<td>StepTolerance</td>
<td>1</td>
<td>Tolerance on the variable</td>
</tr>
<tr>
<td>UseParallel</td>
<td>false</td>
<td>Compute in parallel</td>
</tr>
<tr>
<td>UseCompletePoll</td>
<td>true</td>
<td>Completely poll around current point</td>
</tr>
<tr>
<td>InitialMeshSize</td>
<td>16</td>
<td>Initial polling mesh size</td>
</tr>
<tr>
<td>MaxMeshSize</td>
<td>32</td>
<td>Maximum polling mesh size</td>
</tr>
<tr>
<td>MaxIterations</td>
<td>100</td>
<td>Maximum number of iterations for optimization</td>
</tr>
<tr>
<td>PlotFcn</td>
<td>@psplotbestf</td>
<td>Output plots from patternsearch</td>
</tr>
<tr>
<td>ScaleMesh</td>
<td>false</td>
<td>Scale the mesh size</td>
</tr>
<tr>
<td>Cache</td>
<td>on</td>
<td>Keeps a history of mesh points</td>
</tr>
<tr>
<td>MeshRotate</td>
<td>on</td>
<td>Rotates the pattern before declaring optimum</td>
</tr>
<tr>
<td>SearchFcn</td>
<td>@searchga</td>
<td>Type of search used by the patternsearch</td>
</tr>
</tbody>
</table>

4.2.2 dragCalcFS

The function ‘dragCalcFS’ was written to automate the process of running a Flight-Stream simulation in order to be implemented into an optimization algorithm within the MATLAB environment. This function takes the two conic control point values
from the optimization as inputs and in turn, outputs the $C_D$ for that given configuration. In order to achieve this, the geometry is generated using the parametrization defined in Section 4.1. A script is written to automate the entire FlightStream simulation. This is achieved by taking a template script with the simulation settings and filling in the details for the current iteration. The FlighStream simulation used for the optimization follows the steps as outlined in Chapter 3, but without the addition of wind tunnel walls. The mesh is also refined in certain areas to better produce robust solutions for the algorithm, covered in Appendix A. Once the script is written, the FlightStream simulation is run from the MATLAB interface through the Windows command line. The results from the simulation are saved onto a text file and read into the MATLAB optimization environment. The simulation is set to run an angle of attack sweep so the aerodynamic loads can be used to create a drag polar. During the optimization, a range of angles from 1-5° in 1° increments is run to create a curve. From this drag polar, the $C_D$ at a constant $C_L$ is taken as the value for optimization. The $C_L$ value used for this optimization is 0.7, which was arbitrarily chosen to be in the linear portion of the lift curve since there are no weight estimates for the design at this stage.

4.3 Results

This section covers the results from the optimization algorithm, including some early results to showcase the hurdles in reaching a parametric optimum within reason. In order to achieve this final result, many of the design variables and optimization options were tweaked along the process to remove any erroneous optimized results.
4.3.1 Initial Results

Figure 4.6 below is an optimized result from one of the early optimization cycles (cycle 12). Note that the portion of the fuselage after the trailing edge of the wing expands without following the general trend of contraction. This is an undesirable result that will likely not correspond to an optimum design in reality.

![Figure 4.6: "Optimum" Geometry from Cycle 12, Iteration 17](image)

Table 4.4 shows the design variables and function values for the initial and optimum iterations in the optimization cycle. Note that $x_2$ is the same at the initial iteration. This phenomenon will be discussed further in Chapter 5.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$C_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>107</td>
<td>87</td>
<td>0.0450</td>
</tr>
<tr>
<td>17</td>
<td>89</td>
<td>87</td>
<td>0.0426</td>
</tr>
</tbody>
</table>

Table 4.4: Optimization Results from Cycle 12

Figure 4.7 shows the differences in geometry between the initial and final configuration from optimization cycle 12. The dashed geometry is the initial configuration, while the solid geometry is the optimum from this cycle. The wing shifts with the fuselage contraction as intended. Figure 4.8 shows the drag polar between the initial and optimized geometry. The surface $C_p$ and trailing edge streamline for the optimized geometry are provided in Figures 4.9 and 4.10.
Figure 4.7: Outlines of Initial and Optimum Configurations (Cycle 12)

Figure 4.8: Drag Polar of Initial and Optimum Configurations (Cycle 12)
Figure 4.9: Surface $C_p$ and Streamlines from Cycle 12 ”Optimum” Configuration

Figure 4.10: Surface $C_p$ and Streamlines from Cycle 12 ”Optimum” Configuration
4.3.2 Final Result 1 (Cycle 36)

This section covers the results from the first of two separate optimization cycles. This result was chosen to be acceptable because the same parametric optimum was repeated among multiple optimization cycles. Cycle 37 starts with different starting points, but shows that it achieves the same optimum configuration as cycle 36. Note that there were many tweaks required to converge on a feasible solution. The two main differences between the initial and final results are the control points and the wing location. The conic control points were changed from indices [8,10,12,16] to [8,9,11,16]. The wing location was changed from $x = 665\, \text{mm}$ to $x = 565\, \text{mm}$ and the previously variable $z$ location is now set at a constant $z = 5\, \text{mm}$. Additionally, the use of a genetic algorithm as the search function before patternsearch polling further improved the hit rate of the actual optimum for this problem. Again, it must be noted that this result is a parametric optimum and the output may vary drastically given different parameters for the optimization problem.

Figure 4.11 shows the initial geometry from cycle 36, while figure 4.12 shows the optimized geometry for this cycle. The optimum was found on iteration 48, according to the stopping criteria set for this cycle. This optimization cycle was started with the initial geometry provided in Chapter 3.

Figure 4.11: Initial Geometry from Cycle 36, Iteration 01
Table 4.5 shows the design variables and function values for the initial and optimum iterations in the optimization cycle. The $C_D$ of the optimum configuration 40 counts lower than the initial configuration, which is an 8.6% improvement.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$C_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>115</td>
<td>97</td>
<td>0.0461</td>
</tr>
<tr>
<td>48</td>
<td>96</td>
<td>72</td>
<td>0.0421</td>
</tr>
</tbody>
</table>

Figure 4.13 shows the differences in geometry between the initial and final configuration from optimization cycle 36. The dashed geometry is the initial configuration, while the solid geometry is the optimum from this cycle. In this cycle, the wing stays at a fixed location to isolate the optimization problem to only the fuselage contraction.
Figure 4.13: Outlines of Initial and Optimum Configurations (Cycle 36)

Figure 4.14 shows the drag polar between the initial and optimized geometry, while figures 4.15 and 4.16 show the lift curve and drag curves plotted against the angle of attack.
Figure 4.14: Drag Polar of Initial and Optimum Configurations (Cycle 36)

Figure 4.15: Lift Curve of Initial and Optimum Configurations (Cycle 36)
Figure 4.16: Drag Curve of Initial and Optimum Configurations (Cycle 36)

The surface $C_p$ and trailing edge streamlines for the initial and optimized geometry are provided in Figures 4.17 to 4.20.
Figure 4.17: Surface $C_p$ and Streamlines from Cycle 36 Initial Configuration

Figure 4.18: Surface $C_p$ and Streamlines from Cycle 36 Optimum Configuration
Figure 4.19: Surface $C_p$ and Streamlines from Cycle 36 Initial Configuration

Figure 4.20: Surface $C_p$ and Streamlines from Cycle 36 Optimum Configuration

Figure 4.21 and 4.22 show the surface $C_p$ values along the top and bottom surfaces of the wing and fuselage. These plots provide a quantitative set of $C_p$ data for the initial and optimum configurations from this optimization. The wing $C_p$ is taken at $y = 0.11$ m and the fuselage $C_p$ is taken at $y = 0$ m.
Figure 4.21: $C_p$ Along Top and Bottom Surfaces of the Wing (Cycle 36)

Figure 4.22: $C_p$ Along Top and Bottom Surfaces of the Fuselage (Cycle 36)
The $C_D$ values for each iteration throughout the optimization cycle are given in Figure 4.23. Note that the 'UseCompletePoll' option is set as 'True' in the optimization options, which leads to spikes in the data. Additionally, the random nature of the genetic algorithm accounts for the sporadic data in the first 40 iterations.

![Figure 4.23: $C_D$ vs. Iteration from Cycle 36](image)

4.3.3 Final Result 2 (Cycle 37)

This section covers the second result from cycle 37. The optimized geometry was consistent with that of cycle 36, which adds confidence that the parametric optimum was found.

Figure 4.24 shows the initial geometry from cycle 37, while figure 4.25 shows the optimized geometry for this cycle. The optimum was found on iteration 47 according to the stopping criteria set for this cycle. This optimization cycle started with the initial geometry expanded closer to the upper bounds for $x_1$ and contracted to the lower bound for $x_2$. 

55
Table 4.6 shows the design variables and function values for the initial and optimum iterations in the optimization cycle. The $C_D$ of the optimum configuration was 45 counts lower than the initial configuration, which is a 9.8% improvement.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$C_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>121</td>
<td>58</td>
<td>0.0459</td>
</tr>
<tr>
<td>48</td>
<td>98</td>
<td>71</td>
<td>0.0414</td>
</tr>
</tbody>
</table>

Figure 4.26 shows the differences in geometry between the initial and final configuration from optimization cycle 37. The dashed geometry is the initial configuration, while the solid geometry is the optimum from this cycle. For this cycle, the wing
also stays at a fixed location to isolate the optimization problem to only the fuselage contraction.

Figure 4.26: Outlines of Initial and Optimum Configurations (Cycle 37)

Figure 4.27 shows the drag polar between the initial and optimized geometry, while figures 4.28 and 4.29 show the lift curve and drag curves plotted against the angle of attack.
Figure 4.27: Drag Polar of Initial and Optimum Configurations (Cycle 37)

Figure 4.28: Lift Curve of Initial and Optimum Configurations (Cycle 37)
Figure 4.29: Drag Curve of Initial and Optimum Configurations (Cycle 37)

The surface $C_p$ and trailing edge streamlines for the initial and optimized geometry are provided in Figures 4.30 to 4.33.
Figure 4.30: Surface $C_p$ and Streamlines from Cycle 37 Initial Configuration

Figure 4.31: Surface $C_p$ and Streamlines from Cycle 37 Optimum Configuration
Figure 4.32: Surface $C_p$ and Streamlines from Cycle 37 Initial Configuration

Figure 4.33: Surface $C_p$ and Streamlines from Cycle 37 Optimum Configuration

Figure 4.34 and 4.35 show the surface $C_p$ values along the top and bottom surfaces of the wing and fuselage. These plots provide a quantitative set of $C_p$ data for the initial and optimum configurations from this optimization. The wing $C_p$ is taken at $y = 0.11$ m and the fuselage $C_p$ is taken at $y = 0$ m.
Figure 4.34: $C_p$ Along Top and Bottom Surfaces of the Wing (Cycle 37)

Figure 4.35: $C_p$ Along Top and Bottom Surfaces of the Fuselage (Cycle 37)
The $C_D$ values for each iteration throughout the optimization cycle are given in Figure 4.36.

Figure 4.36: $C_D$ vs. Iteration from Cycle 37
This chapter includes a detailed discussion of the fine-tuning process for the optimization algorithm, followed by the results. The challenges of producing a feasible design using parametric optimization are highlighted along with areas of improvement.

5.1 Initial Results

The initial results from Section 4.3.1 are of interest because of how frequently it preferred the re-expansion during the optimization. The fuselage had a second expansion of its cross section after the initial contraction, which likely does not correspond with an improved or optimum design. Subjectively, the design also fails to satisfy aesthetics (which is another consideration when designing an aircraft). If a design doesn’t look right, it is oftentimes an early indicator that something is fundamentally wrong.

One reason for the re-expansion of the fuselage was likely the positioning of the wing relative to its contraction. Although the design variables were set to change the contraction of the fuselage, the wing positioning indirectly affected the optimization as it was coupled to the fuselage geometry. The wing shifting up and down with respect to the fuselage makes it harder to determine if the efficiency gains were coming from the fuselage itself or the wing being in a favorable position.

Taking a step back to look at Chapter 1 and the discussion of inverse pressure gradient matching in Mike Arnold’s AR-5 design, the same principles hold true. Having the adverse pressure gradient of the wing be in the same location as the adverse pressure...
gradient of the fuselage is detrimental to the interference drag of the aircraft. The re-
expansion was an attempt to delay the adverse pressure gradient from the contraction
of the fuselage. Furthermore, the expansion begins where the adverse gradient of the
wing is. This is favorable as the pressure gradients work to cancel each other out
(Figure 1.6).

The fuselage sides are coupled with the upper curve, which should ideally stay parallel
in the area of the wing-fuselage intersection to avoid adding to the adverse pressure
gradient. In future work, the parameters can be changed to keep the fuselage sides
constant, which would also help with keeping the effective wing area constant. An-
other potential solution is to add expanding radius fillets to the wing-fuselage inter-
section. However, this would require a different approach to modeling the geometry
using Computer-Aided Design (CAD). As this requires a complete overhaul of the
framework developed here, it is outside the scope of this thesis.

In an attempt to combat the re-expansion of the fuselage, the wing was moved forward
to $x = 565 \text{mm}$ and kept at a constant $z = 5 \text{ mm}$. This was done in an attempt to
de-couple the wing from the effects of the changing fuselage geometry. Additionally,
the indices of the control points used for the optimization were moved forward from
$[8,10,12,16]$ to $[8,9,11,16]$ to help guide a smoother contraction of the fuselage geom-
etry. Lastly, a constraint was set to ensure that the second point, $x_2$, would always
have to be less than the first point, $x_1$. Aside from these changes to the parameter-
ization, the optimization options were changed to include a genetic algorithm as a
search function before the $\text{patternsearch}$ polling begins. This was done to increase
the likelihood of the optimum geometry being found for each optimization cycle.
5.2 Final Results

The final results given in Sections 4.3.2 and 4.3.3 provide an acceptable configuration that is optimum and also aesthetically sound. Between optimization cycles 36 and 37, the geometry provided was within a couple of millimeters. This helps to prove that the optimization algorithm is now robust enough to produce the same geometry, while also giving confidence in the fact that the optimum here is indeed the parametric optimum for this case.

The re-expansion of the fuselage is eliminated due to the updated parameters and constraints. The positioning of the wing with respect to the fuselage contraction is not ideal for inverse pressure gradient matching but provides a middle ground between the ideal "teardrop" shape and holding the contraction until after the trailing edge of the wing. It is likely that the increase in efficiency of the fuselage outweighs the losses that come from the pressure gradients of the fuselage and wing interacting. Overall, the change in geometry is minimal but provides 9% reduction in total drag.

The lift curve plots (figures 4.15 and 4.28) show that there is a slight shift in the lift slopes. This is likely due to two reasons: the fuselage itself creates different amounts of lift and the effective wing area changes due to the fuselage width. The drag curves (figures 4.16 and 4.29) show that both induced and parasitic drag is decreased in the optimization, which is shown by the curves shifting downwards in y. These changes in the lift and drag curves are reflected in the drag polar (figures 4.14 and 4.27), where the entire curve is shifted to the left for the optimized case. This shows that for any given $C_L$, the $C_D$ will be lower in the optimized geometry.

The wing surface $C_p$ plots (figures 4.21 and 4.34) show that the suction peak is increased and shifted forward in $x$ for both cases. This is reflected in the fuselage
surface $C_p$ plots at $y = 0$ (figures 4.22 and 4.35) where the suction peaks also shift forward in $x$, matching the peak of the wing. This forward shift in the local suction of the wing effectively changes resultant force direction more towards the lift and less toward drag. The pressure recovery seems to be smoother in the optimized cases, even if the suction peaks are similar in magnitude. This allows for a smooth adverse pressure gradient in both the optimized cases, whereas the initial geometries have a shallow pressure recovery at the beginning followed by an aggressive adverse gradient towards the end. The optimized geometry provides a more efficient way to produce a certain amount of lift ($C_L = 0.7$). Some of this may also be attributed to the wing area increasing due to the early contraction.

Figure 4.23 and 4.36 provide the $C_D$ values at each iteration. The trend line shows a negative trend between the first and last iteration, which is the expected trend as the optimization is converging to a minimum. During the first 40 iterations, the genetic algorithm causes the search to look random. Afterward, the patternsearch polling takes over from the genetic algorithm’s minimum to find any other local minima. Due to the nonlinear nature of the problem, a genetic algorithm aids in finding the optimum more frequently when compared to using only the patternsearch method.

Once the optimum geometry is output by the optimization algorithm, it must be validated. The initial and final geometry must be analyzed in either higher-fidelity CFD methods or wind tunnel testing to validate FlightStream for use in an optimization environment. Although the raw $C_D$ values may not exactly correlate to higher order methods, as long as the $C_D$ delta and its magnitude agree with FlightStream, it is successfully validated.

As the design matures and starts to converge, higher-fidelity methods for analysis may be used for optimization. The panel code is useful in early design stages to quickly and accurately calculate the aerodynamic loads of a vehicle, but it is not
without its limitations. Although FlightStream includes a boundary integral method to account for viscous effects, there may be smaller details that are not captured within the boundary layer that only more accurate solvers can properly model. After optimizing with a panel code, the solution should be checked with other methods of drag estimation such as higher fidelity models or even wind tunnel testing.

As the scope of the optimization problem broadens, the *parametric* optimum can begin to look like the *global* optimum. This requires the use of more independent variables in the optimization, which scales the computational cost in orders of magnitude depending on the type of optimization algorithm used. For this problem, two independent design variables are coupled with other parameters (such as the fuselage side and bottom conic points) to reduce computational cost. We can start to remove this coupling to increase the number of independent variables by using all the conic curve control points as design variables or adding separate variables to control wing geometry and location.

Depending on the scope of the problem, design variables should be thoughtfully chosen to provide an optimum in the desired design space. The limiting factor is computational cost and time, one solution of which is using parallel processing on high-performance computing clusters. In many cases, these resources are limited and it may not be feasible to develop and incorporate complex optimization algorithms to come up with a design. Zooming out, the results presented in this thesis prove that topology optimization is only a tool in the aerodynamic design process. The design must consider not only aerodynamics but also aesthetics and specific design requirements. A good aerodynamicist should balance these design criteria while using fundamental principles to create the best possible design. From here, the design can be used as a starting point for an optimization algorithm to fine-tune the geometry.
This helps simplify the scope of the optimization problem, which is a large save on computational cost.
6.1 Conclusion

With the potential application of a turboprop racer concept developed to break world speed records, an optimization framework was developed in MATLAB which handles the geometry parameterization, objective function scripting, and optimization of this problem. This framework interfaces with a vorticity-based panel solver, FlightStream, which was validated against wind tunnel data. The framework was developed for a sailplane based on the ASW-19 and ASW-20 which was studied at Delft University. A case study of the AR5 by Mike Arnold provides valuable insight into inverse pressure gradient matching, which helps with the understanding of pressure gradients and how they interact with one another. This is especially important for the wing-fuselage junction, which is the main contributor to interference drag in this system. The optimization framework was successfully able to provide optimized results, providing a feasible design with a potential 9% improvement in total drag. The optimum found here was parametric and not the global optimum. This means that the parameters chosen for this problem will in some way produce slightly different results. In order to achieve a global optimum, the parameterization must take into account many other independent variables which shall not be coupled with each other. Oftentimes, finding the global optimum is not feasible and the computational cost of this optimization problem must be balanced with proper design considerations to arrive at an optimum.
6.2 Future Work

Although the framework was built to provide a parametric optimum for this problem, it is necessary to fine-tune the parameterization of the model. Coupling design variables to reduce the computational time comes with its own trade-offs and must be balanced with the desire for a parametric or global optimum. Over time, a more complex parameterization of the model shall be used to fine-tune the geometry. This will allow for a more robust optimization code that will reliably produce an optimized result. Adapting this framework for a high-performance computing environment will become necessary with the increasing number of design variables. Although FlightStream does run efficiently on a local setup, the growing number of function evaluations will eventually require more computational resources.

The next area of development would be to fine-tune the mesh generation. Especially with a changing wing and more complex fuselage geometries, the mesh refinements must be able to automatically adapt to the changing design. Throughout the development of this framework, simulations would occasionally not converge onto a solution creating a hole in the optimization. Checks are now in place to recognize these outlier points and re-run the simulation with a new mesh, but a more robust meshing strategy will certainly be helpful. The incorporation of a CAD-based model and mesher is recommended as the parameterization is being scripted. If using a .stl (Standard Tessellation Language) surface geometry, Blender [9], an open-source 3D computer graphics software, can also be used to morph the geometry using scripted parameters.

The optimization algorithm used, patternsearch, was chosen due to its simplicity and low computational cost for this thesis. A more robust genetic algorithm was used in combination to increase the hit rate for the optimum. Doing so also required more computational resources and only a couple of course generations were used.
in the meantime. As more design variables get introduced into the problem, the optimization algorithm and its options must be tweaked again to properly find a solution.

Lastly, the design problem that this framework was developed for is the Turboprop Racer concept led by Dr. Iscold. Although the geometry parameterization will need an overhaul in order to meet the specific needs of the design, this thesis acts as a proof-of-concept for using a low-order panel code such as FlightStream while providing a framework for the optimization. The aim was to cover enough groundwork as a starting point in getting an optimization algorithm going for this next-generation aircraft design.
BIBLIOGRAPHY


[8] fai.org. Records. https://www.fai.org/records/f%5B0%5D=field_record_sport%3A2020&f%5B1%5D=field_type_of_record%3A158&f%5B2%5D=field_group%3A177&f%5B3%5D=field_status%3A146.


This section covers the initial mesh convergence study performed to form the baseline mesh, as well as the extra refinements that were used to increase the robustness of the simulation required by the changing geometry throughout the optimization cycles.

When importing the geometry using a series of cross-sectional curves, the mesh density can be defined using two variables, \( u \) and \( v \). The \( u \) mesh parameter defines the chordwise refinement for lifting components and the azimuth refinement for non-lifting components. The \( v \) mesh parameter defines the spanwise refinement for lifting surfaces and axial refinement for non-lifting surfaces. A growth rate with a specified periodicity can also be defined to add more refinement in areas such as the leading and trailing edge of a wing.

To get the initial mesh, a simple mesh convergence study was performed. This was done by isolating the wing and fuselage, focusing on the convergence of the drag value for each component in freestream. The \( u \) and \( v \) values were each held constant while the other was swept through values increasing in increments of 10. This method was chosen to isolate and ensure that the drag value would not be sensitive to each mesh parameter. Figure A.1 shows the convergence for the wing mesh and Figure A.2 shows the convergence for the fuselage mesh. The minimum mesh values for this geometry were \( u = 100 \) and \( v = 80 \) for the wing and \( u = 50 \) and \( v = 100 \) for the fuselage. The wing had trouble converging on a mesh size, so a growth rate of 1.05 was used favoring the leading and trailing edges.
Figure A.1: $u$ and $v$ Mesh Parameters for the Wing

Figure A.2: $u$ and $v$ Mesh Parameters for the Fuselage
When joining the two systems together, the wing-fuselage interface requires a slightly more refined mesh to ensure that the Unite tool can properly capture the junction. Additionally, this specific area is where there are large pressure gradients interacting with one another. The wing-fuselage junction and its interference drag can be captured with better accuracy using a finer mesh in this area. Mesh zones can be defined in the cross-section import file, which specifies the number of mesh points for a defined area along the geometry. For this problem, the wing and fuselage mesh was refined in the location of the wing intersection. Figure A.3 shows the mesh refinements for both bodies.

![Figure A.3: Refined Mesh at the Wing-Fuselage Junction](image)