



2015-05-01

A Collaborative Conceptual Aircraft Design Environment for the Design of Small-Scale UAVs in a Multi-University Setting

Joseph Samuel Becar

Brigham Young University - Provo

Follow this and additional works at: <https://scholarsarchive.byu.edu/etd>



Part of the [Mechanical Engineering Commons](#)

BYU ScholarsArchive Citation

Becar, Joseph Samuel, "A Collaborative Conceptual Aircraft Design Environment for the Design of Small-Scale UAVs in a Multi-University Setting" (2015). *All Theses and Dissertations*. 5857.

<https://scholarsarchive.byu.edu/etd/5857>

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

A Collaborative Conceptual Aircraft Design Environment for the Design of
Small-Scale UAVs in a Multi-University Setting

Joseph Samuel Bécar

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of
Master of Science

Steven E. Gorrell, Chair
C. Greg Jensen
W. Jerry Bowman

Department of Mechanical Engineering
Brigham Young University
May 2015

Copyright © 2015 Joseph Samuel Bécar
All Rights Reserved

ABSTRACT

A Collaborative Conceptual Aircraft Design Environment for the Design of Small-Scale UAVs in a Multi-University Setting

Joseph Samuel Bécar
Department of Mechanical Engineering, BYU
Master of Science

In today's competitive global market, there is an ever-increasing demand for highly skilled engineers equipped to perform in teams dispersed over several time-zones by geography. Aerospace Partners for the Advancement of Collaborative Engineering (AerosPACE) is a senior design capstone program co-developed by academia and industry to help students develop the necessary skills to excel in the aerospace industry by challenging them to design, build, and fly an unique unmanned aerial vehicle (UAV). Students with little to no experience designing UAVs are put together in teams with their peers from geographically dispersed universities. This presents a significant challenge for the students in assimilating and applying aircraft design principles, using and interpreting output from analysis tools in multiple disciplines, and communicating their findings with their team members in an effective way.

This thesis documents the development of a collaborative design tool for the generation and evaluation of small-scale electric-powered UAV concepts in AerosPACE. The integrated design and optimization software CCADE (Collaborative Conceptual Aircraft Design Environment) enables the immersion of team members from different universities in a software environment which shares design information and analysis results in a central database. Input files for use by open-source analysis tools are automatically generated, and output files read in and displayed in a user-friendly graphical interface. Analysis codes for initial sizing, geometry, airfoil selection, aerodynamics, propulsion, stability and control, and structures are included in the software. Optimization methods are proposed for implementation in future versions of CCADE to explore the breadth of the design space and help students understand the sensitivity of their design to certain key parameters. Testing of CCADE by students during the 2014-2015 AerosPACE course showed an increased volume of explored concepts and prompted questions from students to fill gaps in understanding of fundamental principles. Suggestions for increased student acceptance and use of the software are given. Through its unique architecture and application, CCADE aims to increase productivity and teamwork among AerosPACE participants by increasing the number of concepts which can be fully analyzed, enabling broader exploration of the feasible design space to produce unique and innovative aircraft configurations, and allowing teammates to share thoughts and learning via a shared design and analysis work-space.

Keywords: UAV, aircraft, conceptual, design, analysis, CAD, CAE, collaboration

ACKNOWLEDGMENTS

I would like to thank Dr. Michael Richey, Marc Nance, and Mike Vander Wel of The Boeing Company for their sponsorship and support of AerosPACE and this research.

Thanks to Dr. John Sullivan of Purdue University, who provided significant technical guidance and Excel worksheets used as the foundation for the methods employed by CCADE, as well as valuable perspective concerning the use of CCADE by students. Also, to Dr. Crisler of Embry Riddle Aeronautical University and Carl Johnson of Georgia Institute of Technology gave valuable feedback on the tool and ideas for future work.

A large debt of gratitude owed to Dave French, Josh Coburn, Jordan Johnson, Tim Bright, Jeff Nuss, and Mark Bennett, who, beside looking over code and helping to fix the plethora of bugs encountered during development of CCADE, provided the foundation for my understanding of the programming required for the collaborative features of CCADE.

Special thanks to my mentor, Dr. Steven Gorrell, who provided support and encouragement throughout my research, as well as sound advise, direction, and critique. Thanks for believing in me.

Thanks to my parents and family for their constant, unabated support.

I am forever grateful to my lovely wife Lara, for her contagious positive attitude and unyielding faith in me. Her patience and encouragement when I was discouraged made all this possible. I couldn't have done it without you.

Most of all, I thank my God for the inspiration through the Holy Ghost that supplied me with ideas and helped me conquer every obstacle, and also for the strength and resolve to see this through to the end.

TABLE OF CONTENTS

LIST OF TABLES	vii
LIST OF FIGURES	viii
NOMENCLATURE	xii
Chapter 1 Introduction	1
1.1 Problem Statement	2
1.2 Research Objective	5
Chapter 2 Background	9
2.1 Aircraft Design Tools	9
2.1.1 Parametric Aircraft CAD Tools	10
2.1.2 Aerodynamic	12
2.1.3 Propulsion	14
2.1.4 Structures	14
2.2 Automation and Multi-Disciplinary Optimization	15
2.3 Collaborative Engineering	17
2.4 Benefits of CCADE	19
Chapter 3 Architecture	20
3.1 Database Model Architecture	20
3.1.1 Database Structure	21
3.1.2 Database Management System	23
3.1.3 Data Model Framework	24
3.2 GUI Architecture	25
3.3 Client-Server Architecture	28
3.3.1 WCF Services	28
3.3.2 N-Tier Entity Framework	32
Chapter 4 Workflow	34
4.1 Mission Profile	36
4.2 Initial Sizing and Constraints	37
4.2.1 Constraint Sizing	38
4.2.2 Weight Sizing	45
4.3 Geometry	47
4.4 Airfoil Selection	52
4.5 Aerodynamics	56
4.6 Stability and Control	57
4.7 Propulsion	60
4.7.1 Motor Performance	61
4.7.2 Propeller Performance	62

4.7.3	Motor-Propeller Matching	64
4.7.4	Flight Analysis	66
4.8	Structures	66
Chapter 5	Design Exploration	74
5.1	Problem Statement	74
5.2	Methodology	75
5.3	Proof of Concept	80
5.3.1	Design Variables, Constraints, Objectives	81
5.3.2	Setup	83
5.3.3	Results and Discussion	85
Chapter 6	Implementation in AerosPACE	96
6.1	Course Implementation	96
6.2	Student Feedback	97
6.3	Suggestions for Future Implementation	99
Chapter 7	Conclusion	101
7.1	Summary	101
7.2	Impact of CCADE	102
7.2.1	Workflow Automation	102
7.2.2	Collaborative Design	103
7.2.3	Design Exploration and Optimization	104
7.3	Future Work	104
REFERENCES	107
Appendix A	Database Structure	112
A.1	Independent Database Entities	112
A.2	User Management Entities	113
A.3	Mission Profile Entities	114
A.4	Sizing Entities	115
A.5	Geometry Entities	116
A.6	Analysis Storage Entities	117
A.7	Propulsion Entities	118
A.8	Structures Entities	119
Appendix B	CCADE User Manual	120
Appendix C	Design Exploration Code	150
C.1	Optimus Configuration File	150
C.2	Parametric Workflow Construction Script	151
Appendix D	Optimization Airfoils	155
D.1	CJ 25209	156

D.2	Eppler 186	157
D.3	Eppler 325	158
D.4	Eppler 330	159
D.5	Eppler 340	160
D.6	EH 1.5/9.0	161
D.7	EH 2.0/10.0	162
D.8	MH 61	163
D.9	MH 78	164
D.10	Phoenix	165
D.11	RS-001 T10	166
D.12	RS-004 A	167
D.13	Selig 5010	168
D.14	Selig 5020	169
D.15	SD 7003	170

LIST OF TABLES

4.1	Definition of the types of mission legs available in CCADE, including the parameters necessary to fully describe them.	37
4.2	Description of the inputs to the constraint analysis.	39
4.3	Description of the inputs to the weight analysis. The weight analysis also requires the same inputs as the constraint analysis, given in Table 4.2	45
4.4	Weight sizing outputs derived from the inputs from both sizing operations and the mission profile.	48
4.5	Description of additional inputs entered in the airfoil selection module of CCADE.	54
4.6	Description of additional inputs entered in the airfoil selection module of CCADE.	54
4.7	Types of control surfaces available in CCADE and the properties relevant to their modeling in AVL.	59
4.8	Output derived from raw AVL data in the stability and control module.	60
4.9	Description of inputs used to evaluate the performance of the motor.	61
4.10	Description of inputs used to evaluate the performance of the propeller.	63
4.11	Types of outputs available to be plotted from the propulsion module for assessing the power system and matching motor to propeller.	65
4.12	Description of the structural module inputs to describe each cross-section for a single spar in a wing component.	69
4.13	Definitions of the loads and spar output obtained from the structures module.	72
5.1	Specifications of the design variables, constraints, and objectives in the proof-of-concept DOE and optimization.	84
5.2	The starting point and optimum point of the differential evolutionary algorithm. The optimum was in the 556th experiment.	89

LIST OF FIGURES

1.1	A comparison of relative time spent on conceptual (blue), preliminary (orange), and detailed (gold) design phases in A) the 2013-2014 AerosPACE course and B) the typical large-scale aerospace company in industry. The expected benefits of CCADE may be able to shift the schedule of the 2014-2015 course similar to the relative times shown in C.	3
1.2	CAD models of the final UAV configurations for the three teams of the 2013-2014 AerosPACE course.	4
1.3	In the traditional concept generation and evaluation process, designers meet together to talk about their designs abstractly, then break to perform actual design activities individually. They must then summarize their findings and the team makes a decision dependent on the quality of these reports.	6
1.4	In the proposed process enabled by CCADE, team members can simultaneously access the design data and rapidly execute analyses that previously were done individually outside the meeting. This allows the team to learn together from the design activities and make better collectively informed decisions.	7
2.1	The conceptual design workflow presented by Raymer. The workflow is iterative in nature and is focused on full-scale fueled aircraft. Image courtesy of [1].	11
3.1	The structure for the collaborative design cloud. Information flows upward through the cloud to be stored in a centralized database. Users receive continuous updates on values changed via queries to the team database, allowing them to work collaboratively on the designs.	21
3.2	Diagram of partial object-oriented data structure.	22
3.3	Diagrams of the MVC and MVVM GUI architecture models.	26
3.4	A screenshot of the graphical user interface of CCADE. The graphs shown can offer zooming and navigation capabilities, and all data is linked directly to information located on the database.	27
3.5	The dialog of IIS for hosting a WCF service. The default web site is configured is configured with open ports for communication, and can be accessed from outside the network with a IP address on the public domain.	29
3.6	A schematic of N-Tier architecture, adopted by CCADE, with the role of N-Tier Entity Framework highlighted. No infrastructure or integration services were necessary for CCADE. Image courtesy of [2].	33
4.1	The workflow for CCADE, adapted from Raymer. Concepts are generated, sized, sketched, analyzed, and optimized automatically according to user input of mission requirements and mission profile. The concepts are compared according to selected criteria and the best design selected.	35
4.2	An example chart plotting the constraints on power loading as a function of wing loading. Feasible design space is the area above each constraint curve. The selected design point is marked with the red diamond.	40
4.3	A single-section wing component example using the XML language of OpenVSP.	49

4.4	A screenshot of the geometry viewport in CCADE displaying the generated STL file for all team members.	50
4.5	Plot of the Eppler 325 airfoil as displayed in the CCADE GUI. Plots of the airfoil help students identify airfoils such as this that have reflex.	53
4.6	Examples of output produced by XFOIL and graphed by CCADE to display to all team members. Figure 4.6f shows the upper surface in green and the lower surface in red.	55
4.7	A screenshot of the panel geometry automatically generated in CCADE for use in calculating aerodynamics with AVL.	56
4.8	Examples of output produced by AVL and graphed by CCADE to display to all team members.	58
4.9	Examples of output produced by the propulsion module and graphed by CCADE to display to all team members. Operating line determined by the matching equations graphed in green.	67
4.10	Examples of output produced by the propulsion module to analyze the propeller at the operating point during acceleration and deceleration.	68
4.11	Plot of lift versus spanwise position from the actual AVL output (black) and the 2nd-order Taylor Series Approximation (green). The approximation effectively interpolates the output data by following the trend of the data.	71
4.12	Examples of output produced by the structures module and graphed by CCADE used for sizing the main wing spar, assuming that the spar is the only load-carrying member under the applied loads.	73
5.1	The proposed architecture for the design exploration module. The thin-client sends a job request to the server, where it waits in a queue until it reaches the front of the line. After the job is run by the server, the results are stored in the database and the client can then query the database to display them.	76
5.2	A screenshot of the design exploration module GUI interface. The module currently only handles setup for a trade-study including aerodynamic and stability analyses.	77
5.3	A visualization of the template Optimus workflow. Inputs are shown as blue ovals, outputs as red ovals, hexagons, or stack of rectangles, files are shown as green cylinders or block containing “XML”, and batch scripts are shown as orange squares with a lightning bolt.	80
5.4	A diagram of how the available geometry input parameters in CCADE are organized. Only three of the driving parameters, shown in light-blue, are shown, depending on the original geometry definition.	82
5.5	A plot from the DOE comparing change in pitching moment coefficient with change in elevon percent chord. There is a clear decrease in pitching moment negative slope with increasing elevon percent chord, even after considering variation from other parameters.	87
5.6	A table showing the sensitivity of a subset of the parameters to each other. The parameters with the highest sensitivity to each other have higher values (warmer colors). The diagonal should be neglected since it represents the sensitivity of a parameter to itself.	88

5.7	Visualization of the UAV design geometry used for the proof-of-concept optimization study.	90
5.8	Three-dimensional scatter plots showing the progression of the genetic algorithm through the design space. All experiments are shown in blue. Generations are highlighted in turquoise.	91
5.9	The optimization progress by generation (iteration). Each parameter is shown as a separate plot, with braces showing the diversity of the population, and the upper/lower bounds and target values shown. Parameters shown are, from top to bottom, the Optimus-defined objective function, elevon percent chord, section 1 sweep, section 2 aspect ratio, section 3 dihedral, angle of attack at cruise, total drag coefficient, and elevator deflection.	95
A.1	Entities which do not rely on relationships to other entities. UAV_Designs is the top-level entity for all designs, HelpUrls is a list of HTML links which make course lectures immediately available to students, and Materials is a source for all user- or administrator-defined materials.	112
A.2	Entities for users, including students, faculty, and coaches, and the teams they belong to.	113
A.3	Entities which define the mission profile. All mission leg types inherit from the entity MissionLeg to enable polymorphism.	114
A.4	Sizing parameters are stored in under a common entity set using a name and value.	115
A.5	Entities which completely define the geometry for the UAV. Geometry is broken down into Components, Sections, Airfoils and Controls.	116
A.6	All analysis storage entities inherit from AnalysisOutputs. The disciplines defined here include Airfoil Selection, Stability and Control, and Aerodynamics.	117
A.7	Entities for the propulsion components including Motor, Battery, and Propeller.	118
A.8	Entities for storing spar design data and output. Spars are defined by their individual cross-sections.	119
D.1	CJ 25209 airfoil profile.	156
D.2	CJ 25209 airfoil performance at a Reynolds number of 500,000.	156
D.3	Eppler 186 airfoil profile.	157
D.4	Eppler 186 airfoil performance at a Reynolds number of 500,000.	157
D.5	Eppler 325 airfoil profile.	158
D.6	Eppler 325 airfoil performance at a Reynolds number of 500,000.	158
D.7	Eppler 330 airfoil profile.	159
D.8	Eppler 330 airfoil performance at a Reynolds number of 500,000.	159
D.9	Eppler 340 airfoil profile.	160
D.10	Eppler 340 airfoil performance at a Reynolds number of 500,000.	160
D.11	EH 1.5/9.0 airfoil profile.	161
D.12	EH 1.5/9.0 airfoil performance at a Reynolds number of 500,000.	161
D.13	EH 2.0/10.0 airfoil profile.	162
D.14	EH 2.0/10.0 airfoil performance at a Reynolds number of 500,000.	162
D.15	MH 61 airfoil profile.	163
D.16	MH 61 airfoil performance at a Reynolds number of 500,000.	163

D.17 MH 78 airfoil profile. 164
D.18 MH 78 airfoil performance at a Reynolds number of 500,000. 164
D.19 Phoenix airfoil profile. 165
D.20 Phoenix airfoil performance at a Reynolds number of 500,000. 165
D.21 RS-001 T10 airfoil profile. 166
D.22 RS-001 T10 airfoil performance at a Reynolds number of 500,000. 166
D.23 RS-004 A airfoil profile. 167
D.24 RS-004 A airfoil performance at a Reynolds number of 500,000. 167
D.25 Selig 5010 airfoil profile. 168
D.26 Selig 5010 airfoil performance at a Reynolds number of 500,000. 168
D.27 Selig 5020 airfoil profile. 169
D.28 Selig 5020 airfoil performance at a Reynolds number of 500,000. 169
D.29 SD 7003 airfoil profile. 170
D.30 SD 7003 airfoil performance at a Reynolds number of 500,000. 170

NOMENCLATURE

A	Cross-sectional area of the spar
α	Angle of attack, or thrust multiplier for non-sea-level take-offs
AR	Aspect Ratio, the ratio of span to chord
B	Number of blades a propeller has
b	Span, distance from root to tip of wing or section or for non-symmetric wings, it is twice the physical span
β	Weight multiplier for partially filled fuel tanks
c	Chord, distance from leading edge to trailing edge
c_{ref}	Mean geometric chord of the main lifting component
C_L	Coefficient of lift
C_P	Power coefficient of the propeller
C_T	Thrust coefficient of the propeller
C_{D_0}	Parasitic drag, consisting of viscous and pressure drag sources
C_{D_i}	Induced drag, drag due to lift
$C_{L_{max}}$	Maximum lift coefficient of the aircraft
$C_{L_{mp}}$	Coefficient of lift at minimum power
$C_{f_{turb}}$	Skin friction coefficient for a flat plate in turbulence
D	Drag or propeller diameter
e	Oswald efficiency or span efficiency
η	Efficiency
η_m	Motor efficiency
η_p	Propeller efficiency
g	Gravitational acceleration constant
GR	Gear ratio, the ratio of angular velocities between input and output
I	Moment of inertia of the spar
I_0	No-load current, the current draw by the motor when it is unloaded
I_{in}	Motor load or current draw from the battery by the motor to drive the propeller
J	Advance ratio of the propeller
J_0	Advance ratio at propeller stall
J_{match}	Matched advance ratio of the propeller to the motor
k	Induced drag coefficient
$k_{battery}$	Energy density of the batteries, in units of energy per unit weight
K_t	Thrust coefficient of the motor
K_v	RPM constant, the number of RPMs the motor will spin at per volt applied
L	Lift
$\frac{L}{D}$	Lift-to-drag ratio
M	Bending moment on the spar
μ	Viscosity of air, or coefficient of friction of the runway
n	Load factor, the amount of gravitational acceleration per total weight applied to the aircraft during a turn, or rotational speed of the propeller, $\frac{RPM}{60}$
ω	Bending stress in the spar
P	Power, either provided by the propeller for thrust, from the motor to drive the propeller, or from the battery
p	Propeller pitch
P_m	Power provided by the motor

P_{prop}	Power required to spin the propeller
$\frac{P}{W_{TO}}$	Power loading
q	Dynamic pressure of air at flight altitude and velocity
R	Propeller radius, half of the propeller diameter
ρ	Air density at flight altitude
ρ_{SL}	Air density at sea level
ROC	Rate of climb
RPM	Revolutions-per-minute of a motor or propeller
R_m	Electrical resistance of the motor
R_{bc}	Electrical resistance of the battery
Re	Reynolds number
S	Planform area, the projected area of the wing or section as viewed from overhead
S_{ref}	Projected planform area of the main lifting component
S_{wet}	Wetted area of the aircraft
s	Spanwise distance from the wing tip
s_{TO}	Take-off or landing distance
T	Thrust provided by the propeller, or air temperature at the flight altitude
t_{leg}	Duration of a particular mission leg
T_m	Torque of the motor
T_{prop}	Thrust provided by the propeller
τ	Shear stress in the spar
θ	Number of turns in a particular mission leg, with one turn equal to 360^{deg}
V	Velocity of the aircraft
V_0	Stall Velocity
V_b	Nominal voltage of the battery
V_s	Shear loading on the spar
$V_{BestRange}$	The velocity at which the aircraft will achieve the maximum possible range
V_{TO}	Take-off or landing velocity, defined as 20% greater than the stall velocity
$V_{ceiling}$	Velocity at the ceiling constraint
V_{in}	Input voltage of the battery to the motor
W_b	Weight of the batteries used to power the aircraft for some mission leg
W_e	Empty weight of the aircraft, without fuel (batteries) or payload
W_p	Weight of the payload the aircraft is intended to carry
W_{TO}	Take-off weight of the aircraft
$\frac{W_{TO}}{S}$	Wing loading
x	Distance traveled during a mission leg, or a discretized section of spar set along the span
Δx	Distance between panels in a panel method CFD analysis
y_{alt}	The altitude above sea-level
y_{na}	Distance to the neutral axis of the spar

CHAPTER 1. INTRODUCTION

In today's competitive global market, there is an ever-increasing demand for highly skilled engineers equipped to perform in teams dispersed over several time-zones by geography. Large companies in the aerospace industry also face an aging workforce, with the percentage of employees eligible for retirement approaching 25% [3]. In order to address both of these concerns, companies like Boeing have increasingly reached out to universities through capstone programs to motivate engineering students to join their profession and bridge the gaps in traditional engineering curricula [4]. Capstone programs in mechanical and aerospace engineering programs expose students to the entire design process from problem identification and conceptual design to testing and manufacturing [5]. Aerospace Partners for the Advancement of Collaborative Engineering (AerosPACE) is such a capstone program co-developed by academia and industry as a pipeline to equip students with the skills they need to succeed in the aerospace industry [6].

The ASEE Vision 2030 Task Force identified "working collaboratively and in virtual design teams" as one of the keys to success for mechanical engineers in dominant engineering organizations [7]. Another key they identified for promoting better college graduate preparation was a partnership between academia and industry, molding the curriculum to the needs of both. Following their recommendations, AerosPACE puts students in teams with their peers from multiple geographically dispersed universities and assigns coaches from Boeing to provide an industry perspective. The teams are multi-disciplinary, consisting of mechanical, aerospace, and manufacturing engineering students. Over the course of two semesters, student teams design, build, test, and fly a small-scale, electric-powered unmanned aerial vehicle (UAV) that fulfills requirements documented in a Request for Proposals (RFP). In this way, AerosPACE builds "core competencies for the next generation of aerospace innovators in a sociotechnical, collaborative environment founded in the learning sciences" [8].

One defining aspect of the project is its collaborative nature, challenging students to solve problems in design, project management, data sharing, and manufacturing in geographically dispersed teams. When teams are dispersed, there are often gaps and discrepancies in communication, and sharing files and information becomes a serious issue. The time spent sifting through disorganized information to locate a specific piece of required data can greatly hamper the students' progress. These challenges simulate the distributed nature of large aerospace companies, and students are encouraged to discover the best methods to overcome such challenges. However, there is still much that can be done to facilitate collaboration in these dispersed teams, especially when it comes to information sharing and collaborative concept generation.

1.1 Problem Statement

Students from the 2013-2014 AerosPACE course provided valuable feedback which motivated the work of this thesis. During the 2013-2014 course, 36 students were divided into three teams of 12 with students from each university included in each team. The participating universities were Brigham Young University, Purdue University, Georgia Institute of Technology, and Embry-Riddle Aeronautical University. The type of questions asked by the students during the conceptual design phase of the project revealed a lack of understanding of the fundamental design process, as well as obvious communication struggles. Such questions included:

- “What am I supposed to do with these analysis results?”
- “What tools can I use to get this information?”
- “What do these results tell me about my design?”
- “Where do I get the information to run this analysis?”

The schedule of the 2013-2014 AerosPACE course is illustrated in Figure 1.1A, divided into the design phases of Conceptual, Preliminary, and Detailed Design. Figure 1.1B shows the comparison to the relative time spent in industry on these design phases. The 2013-2014 course was obviously front-heavy on the conceptual design phase, which resulted in students having difficulty conducting the detailed analyses, tests and manufacturing the later phases of the design in the time allotted.

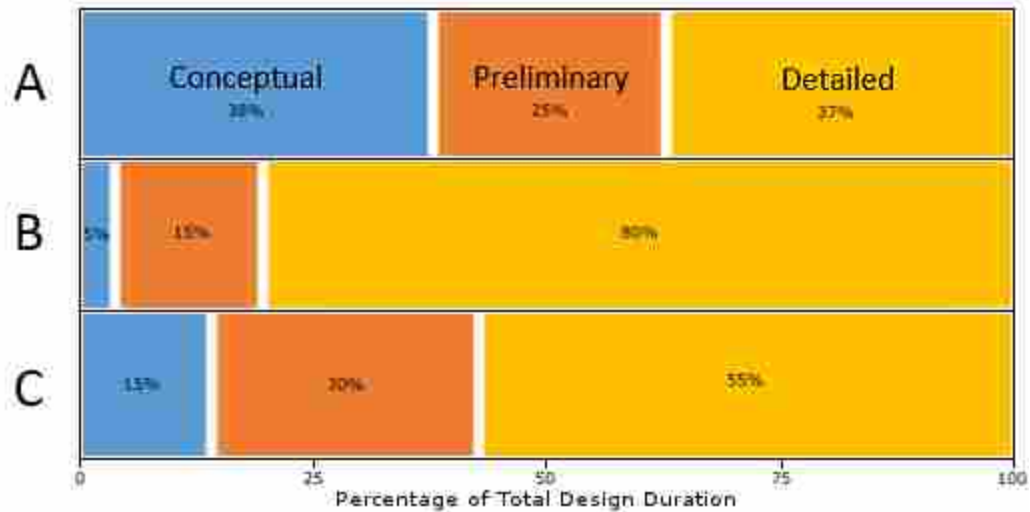


Figure 1.1: A comparison of relative time spent on conceptual (blue), preliminary (orange), and detailed (gold) design phases in A) the 2013-2014 AerosPACE course and B) the typical large-scale aerospace company in industry. The expected benefits of CCADE may be able to shift the schedule of the 2014-2015 course similar to the relative times shown in C.

For the 2014-2015 course, when the work of this thesis was presented, Tuskegee University was added to the list of academic partners, and the number of students was increased to 44. This required that the total number of teams be increased to four, and teams were formed making sure at least three universities were represented on each team. To give students more time for these necessary and critical processes, the schedule for the 2014-2015 course was revised to reflect the distribution shown in Figure 1.1C. In order to justify shorting the conceptual design phase, something needed to be done to help students grasp the fundamentals more quickly and allow them to explore the design space in a shorter amount of time. In addition to making more time for detailed design and manufacturing, there was a need recognized during the 2013-2014 course to help students analyze a more diverse set of aircraft concepts and stimulate greater creativity. The RFP for that year required students to design the aircraft with a blended wing-body configuration. As illustrated by Figure 1.2, this requirement, combined with the difficulty of rapidly iterating through the analysis of competing UAV concepts, resulted in final UAV designs that were very similar.

Flow of information and data sharing between integrated product teams (IPTs) was not ideal in the 2013-2014 course. Based on student feedback, a significant volume of analytical results

went unused simply because there was a lack of understanding about who actually needed the data and what that data contributed to the overall design. Not only was this wasteful with relation to the time students spent generating the results, but it was an obstacle for IPTs relying on the timely delivery of information for their own analyses. Teams did not understand how to communicate with other IPT disciplines what types of information they required, nor could they actively identify which IPTs might need the information they were generating. While team meetings included reports from IPTs on the progress they were making, students often needed to meet outside of these scheduled meetings to obtain the information that was left out of the presentation. This resulted in delays which might have been avoided had the students understood the system-level design process.

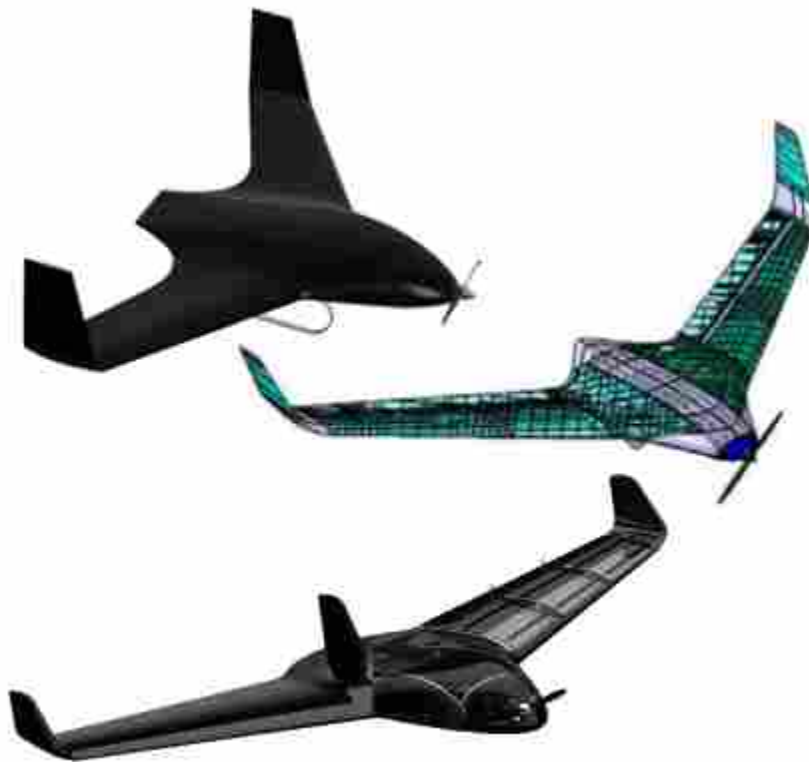


Figure 1.2: CAD models of the final UAV configurations for the three teams of the 2013-2014 AerosPACE course.

Even when the design process is understood, there is a learning curve when attempting to use the UAV conceptual design tools available to students. Vortex lattice methods and panel

methods, for example, are powerful to the experienced designer, but are certainly not intuitive to a student exposed to aircraft design for the first time. During the 2013-2014 course, students related spending significant time wrestling with the user interfaces and controls of these programs, only to learn after presenting their results that they had one or more incorrect inputs. Such tasks can detract from students' learning and postpone correct understanding of fundamental principles until their work is reviewed by a faculty member or teaching assistant, if at all.

The problems addressed by this thesis include the slow speed of generating concepts and evaluating them based on established analytical models, as well as faulty communication between team members due to divided work-spaces, synchronous workflows, and lack of understanding about the design process. There is also a need to enhance creativity and both enable and encourage students to explore greater portions of the design space defined by the RFP. By alleviating some of the more tedious and time-consuming tasks for student designers, more time is available to brainstorm more innovative solutions satisfying the mission requirements.

1.2 Research Objective

The main objective of this thesis is to present the architecture, development, and benefits of a software tool created in response to the problems described in Section 1.1. The tool, a Collaborative Conceptual Aircraft Design Environment (CCADE), assists students during the conceptual design phase of the AerosPACE course by providing the framework for collaboratively generating and analyzing UAV concepts which fulfill mission requirements as defined in the RFP. The program is flexible enough to analyze a wide range of UAV concepts, making it applicable for a variety of RFPs during more than one academic year. It captures the entire workflow for electric-powered UAV design and allows the user the freedom to explore non-traditional configurations in addition to conventional ones. CCADE significantly reduces the time required to evaluate the strength of a design configuration, allowing students to make better educated decisions earlier in the conceptual design. Furthermore, to connect students across multiple universities and allow uninhibited sharing of information among team members, a collaborative framework which contains all design and analysis information in the cloud is presented as an integral feature of CCADE.

As a design tool, CCADE integrates analysis software from multiple disciplines in a predefined workflow and is flexible enough to handle a variety of configurations, allowing full mobility

of the students through the design space. As a collaboration tool, CCADE facilitates information sharing by storing all pertinent design information in a centralized database server, and provides web services to synchronize data between team members working on the design simultaneously. By making the analysis results immediately available to all team members, an online round-table meeting can be much more interactive and engage all team members. This change of paradigm is illustrated in Figures 1.3 and 1.4.

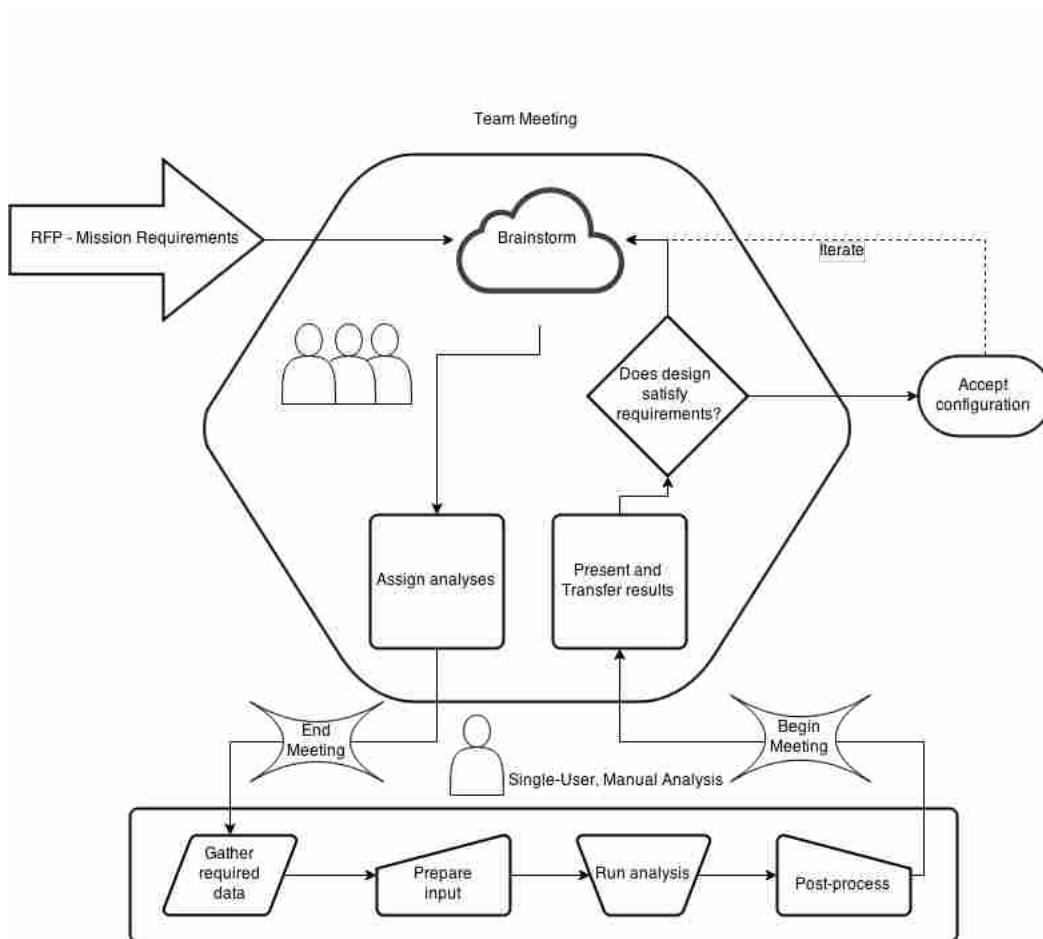


Figure 1.3: In the traditional concept generation and evaluation process, designers meet together to talk about their designs abstractly, then break to perform actual design activities individually. They must then summarize their findings and the team makes a decision dependent on the quality of these reports.

In the typical process for student design teams shown in Figure 1.3, meetings consist of collective brainstorming and student progress reports of work done since the last meeting. This

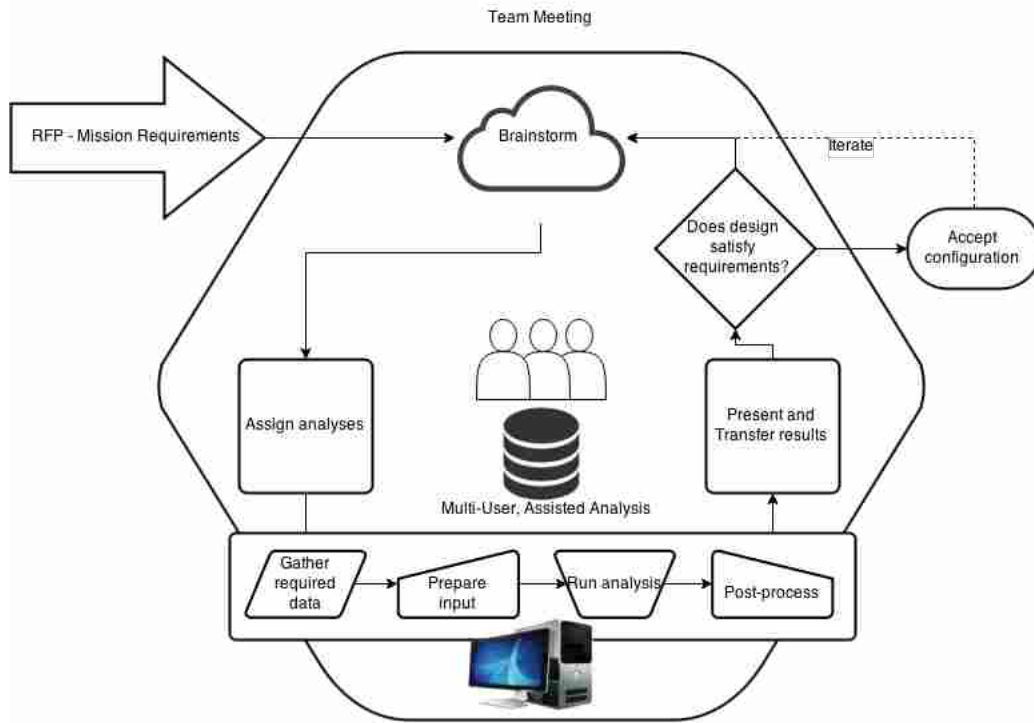


Figure 1.4: In the proposed process enabled by CCADE, team members can simultaneously access the design data and rapidly execute analyses that previously were done individually outside the meeting. This allows the team to learn together from the design activities and make better collectively informed decisions.

method is inefficient for transferring information between IPTs because verbally or visually communicated summaries of the data do not always contain the level of detail required, or do so in a format unacceptable for immediate use. This results in more scheduled team meetings, and potentially requires multiple follow-up meetings between IPT leads to transfer data and negotiate common data formats.

The improved process enabled by CCADE is show in Figure 1.4. Meetings are consolidated, and the multi-user architecture allows either multiple students to enter a design and review the data while it is generated, or subgroups to work on multiple design concepts simultaneously. Within minutes they are able to review their results with the team as a whole. Since the raw analysis data is accessible to all team members, data sharing is immediate and intuitive. Students quickly learn the steps for completing a full analysis of the design, and what information each discipline

needs to complete their respective analyses. This equips them for later design stages where analyses are more detailed and programs which encapsulate the entire workflow do not exist or are not practical.

CCADE is unique in its focus on student education, the conceptual design workflow for small-scale electric-powered UAVs, and collaboration. There exist a number of programs similar in purpose and scope to CCADE, which are described in detail in Chapter 2. Generally, however, the majority of these programs focus on the design of full-scale, fueled passenger aircraft and are unsuitable for the electric UAVs designed in AerosPACE. Many of them are much too detailed or require a much greater experience base than can be expected from university students in their senior year. Furthermore, none of these programs attempts to bring designers together in the same work-space over the web. This research is unique and innovative in that CCADE provides an environment for students to learn from each other in a design work-space that is built for collaboration. Students are guided through the design process by the definition of the workflow, and assisted at every step of analysis. This interactive, collaborative design environment has the capacity to increase the productivity and learning of students, while providing them the tools needed to excel in geographically dispersed teams.

Chapter 2 of this thesis reviews the related literature and published works associated with projects similar to CCADE, including the analysis tools that are utilized by the program for each of its modules. Chapter 3 presents the architecture of CCADE, including the model for data storage, framework for the graphical user interface, and methods for creating a shared workspace among students on a design team via network services. Chapter 4 presents the analytical methods involved in each of the disciplines addressed by CCADE, including inputs and outputs for each tool, and examples of how they might be used. Chapter 5 delves deeper into the specifics of a design exploration module in CCADE, which employs optimization methods to parametrically generate a workflow and run a number of experiments from user-defined design variables, constraints, and objectives. Chapter 6 introduces how CCADE was implemented in the AerosPACE course, presents lessons learned and offers suggestions for future implementation. Finally, Chapter 7 gives the conclusions of the research and suggestions for future work and development of CCADE.

CHAPTER 2. BACKGROUND

This chapter contains a review of the government, academic, and commercial research and activity relating to the design methods, architecture, and applications of this thesis. There has been a number of projects having very similar applications to this one in terms of combining a host of analysis tools to consolidate the design workflow, but few which concentrate on the educational value of such a tool in a capstone course such as AerosPACE, and none which incorporate multi-user functionality such as is attempted here. Luckily, collaborative engineering itself has become an important area of research since the advent of the emerging global marketplace. The foundation of CCADE is established through a synthesis of classical design methods, automation and optimization techniques, and collaboration infrastructures.

A brief description of some of the tools and methods involved in aircraft design is given which have applications to small-scale, electric-powered UAV design. Then a review of multi-disciplinary optimization tools is presented, including tools tailored specifically to aircraft design. Then, advances in collaborative engineering and especially architectures and languages which enable coordination and collaboration between geographically dispersed teams are discussed. Finally, a summary of how this background justifies the analytical and collaborative features of CCADE is given.

2.1 Aircraft Design Tools

The conceptual design analysis workflow for aircraft is well established in literature [1, 9]. Raymer's conceptual design workflow, on which the CCADE workflow is based, is shown in Figure 2.1. To summarize the process shown, after concepts have been generated based on the design requirements, they are realized in the concept sketch. This sketch may be on paper or with a parametric Computer-Aided Design (CAD) tool. This sketch is used to size the aircraft. Sizing is an iterative process, especially for fueled aircraft where the weight of the aircraft changes over

the mission profile. The sized aircraft concept is then analyzed in greater detail for aerodynamics, weight, propulsion, structures, and other relevant disciplines. Once this is done for many different aircraft concepts, a selection for which design to take into the preliminary design can be made.

Many commercial, open-source, and government analysis tools have been developed to reduce the time spent creating spreadsheets and conducting tedious hand-calculations, as well as enable higher-fidelity results and more complex geometry configurations that significantly deviate from traditional ones. Because the aircraft configuration is not defined in detail during the conceptual design stage, suitable analysis tools require relatively short execution times. Thus, the tools addressed in this chapter, which were considered for integration with CCADE, focus on methods which allow definitions of aircraft geometry based on familiar design parameters and require relatively fewer computational resources than finite volume methods for computational fluid dynamics (CFD) and finite-element analysis (FEA), for example. The trade-offs for shorter execution time generally include limiting assumptions in the methods and reduced accuracy. The object of conceptual design is simply to define a configuration that meets the requirements of the RFP, so reduced accuracy at this stage is acceptable. Assessing trends and calculating a rough estimate of performance is sufficient to be able to down-select to one particular configuration to carry over into preliminary design. In this section, various CAD, aerodynamic, propulsion, and structural analysis tools available for use during the conceptual design are introduced.

2.1.1 Parametric Aircraft CAD Tools

For the conceptual design, it is helpful to have a solid or surface model of the design configuration that allows calculation and querying of specific geometric properties. A hand-sketch is useful for brainstorming and visualization, but is limited as far as geometric measurements are concerned. Modern detailed CAD programs are very powerful, but require a significant upfront time investment to model. Therefore, there is a niche in parametric aircraft geometry generation that many software tools have attempted to fill.

Perhaps the most popular of these is Open Vehicle SketchPad (OpenVSP), first developed by NASA in the 1990s and then released as an open-source package in 2012 [10]. The advantages of OpenVSP over detailed CAD include reduced modeling time and ease-of-use. A user may add one or more basic components such as a wing, fuselage, propeller, etc. which are defined using

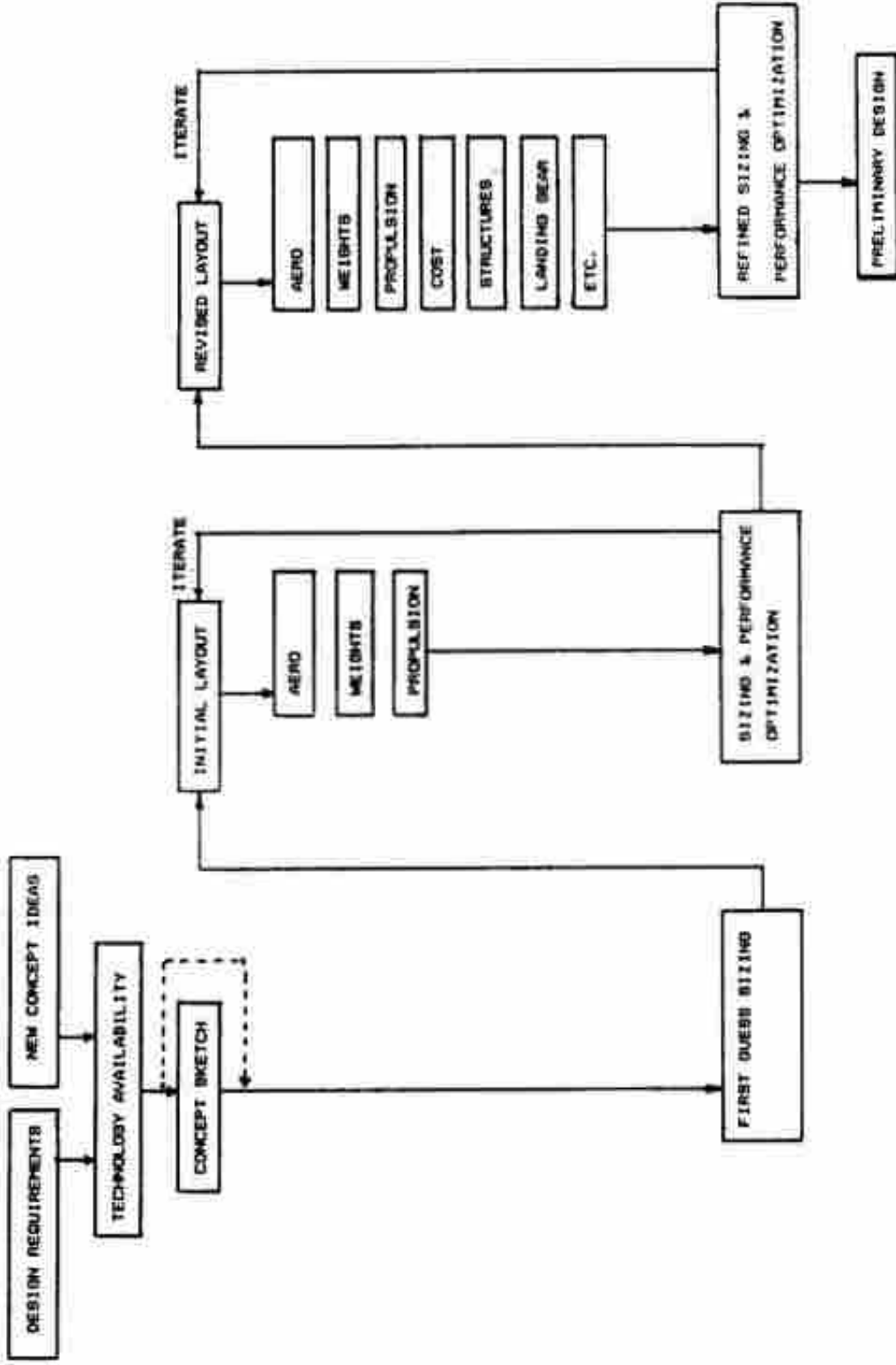


Figure 2.1: The conceptual design workflow presented by Raymer. The workflow is iterative in nature and is focused on full-scale fueled aircraft. Image courtesy of [1].

a set of parametric values familiar to an aircraft designer. For example, the wing may be defined using the span, root chord and tip chord, and then assigned an airfoil shape to interpolate along the span. This is much simpler than the steps involved in creating this geometry in CAD, which would involve importing the set of points defining the airfoil, scaling them with respect to the root and tip chords, connecting them with a spline and then generating a swept surface between the two cross-sections. Furthermore, any change to parameters in OpenVSP yields an almost instantaneous response in the geometry visualization, whereas modifying the CAD definition as described is much more involved, and may even necessitate remodeling the surface from scratch.

The learning curve of using OpenVSP as compared with detailed CAD programs is negligible, as shown by Fredericks et. al. [11]. As of January 2015, the software is also equipped with a powerful application programming interface (API), which offers programmatic control over any of the actions available via the graphical user interface (GUI). Previous versions offered a limited set of commands via a rudimentary scripting engine, which is also taken advantage of in CCADE. The addition of OpenVSP's API has enabled interfacing with multi-disciplinary optimization (MDO) programs and other external code.

Other tools similar to OpenVSP in function include AVID's PAGE [12], Larostera's SUMO [13], and RAGE [14]. PAGE is commercially available, SUMO is an open-source code, and RAGE is a Stanford in-house project. Of the software mentioned, OpenVSP is currently experiencing the most active development [15], and has the most extensive freely available documentation. The code transparency, open license, and familiarity among students and faculty made it the number one choice for use in CCADE.

2.1.2 Aerodynamic

Many of the most popular and reliable aerodynamic tools developed for conceptual designers have been around for decades. Drela and Youngren created two codes in the 1980s for analyzing the aerodynamics of airfoils, and full three-dimensional lifting surfaces. XFOIL is equipped for powerful analysis of airfoils, and has been used extensively in airfoil development and optimization. It's implementation is a combination of computationally fast and relatively accurate high-order panel methods and a fully-coupled viscous-inviscid interaction method [16]. Athena Vortex Lattice (AVL) implements, as the name suggests, vortex lattice methods to analyze the aerodynam-

ics of finite wings [17]. By degrading the real three-dimensional geometry to a set of flat panels for each lifting surface, and applying to each panel a horseshoe vortex, the flow conditions for the entire surface can be solved using a finite-difference approach. One of the main advantages to AVL is the ability to model control surfaces and assess the stability of the aircraft. Perhaps the main limitations of the vortex-lattice method implemented by AVL is the assumption of potential flow around the wing, which ignores viscous effects and compressibility.

In order to overcome this limitation, XFOIL was combined with AVL in the graphical interface of XFLR5 [18]. By running a vortex-lattice method such as AVL and then analyzing the airfoil sections at the appropriate Reynolds number with XFOIL, stall can be accurately predicted and greater understanding of the aerodynamics and stability of the aircraft obtained. Unfortunately, at the time of writing, there is no way to programmatically control XFLR5; all input is expected to come through the graphical interface via human interaction. This drawback makes XFLR5 unsuitable for CCADE.

There are a host of other programs accomplishing a similar purpose which implement the vortex-lattice method, lifting line theory, or the three-dimensional panel method. Tornado, for example, is a Matlab program which implements the vortex-lattice method [19]. APAME, a three-dimensional panel-method, offers the ability to represent fuselages and other complicated geometry in great detail, but otherwise has the same limitations as AVL in that it solves only for induced drag, ignoring parasitic sources such as friction and pressure [20]. AVID VorView is a commercial vortex-lattice tool with roughly the same capabilities as AVL [21]. The advantage of AVL is that by redirecting the standard input, one may quite easily control the program without any human intervention.

AVL and XFOIL are both tried and tested, and although they have limitations, these limitations are well-known and documented. One disadvantage of both XFOIL and AVL is their antiquated command languages, which are unfamiliar and non-intuitive to younger students. The time necessary to learn how to create valid input files and then decipher the resulting output can quickly become restrictive in the context of a comprehensive design class. Automation of these tools can ease this burden, providing students with the ability to utilize key features of these powerful aerodynamic analysis tools without delving into lengthy documentation.

2.1.3 Propulsion

Many numerical tools for analyzing aircraft propulsion exist, which were ultimately passed over in favor of theoretical methods. For propulsion, Capable Computing's MotoCalc™ is perhaps the most popular commercial tool for hobby aircraft designers [22]. Using MotoCalc™, one can input the manufacturer specifications of a motor, electronic speed controller, battery, and propeller, and very quickly assess the predicted power, thrust, and efficiency of the system. It even offers some automatic motor-propeller matching methods, which utilize an internal database of motors and propellers to suggest feasible combinations. However, the methods it uses to calculate performance of the propulsion system are not apparent from the documentation, and the program can only be controlled manually via human interaction. DriveCalc is a tool very similar in capability to MotoCalc, but open-source [23]. While these programs are useful, they typically require a great deal of detailed information about the propulsion system that a student at the outset of the AerosPACE project might not be prepared to give. Simple calculations with generic assumptions such as those utilized by McCormick [24] are sufficient for the conceptual design, and cut down on required processing power. These methods are described in Chapter 4.

2.1.4 Structures

There are many structural design tools available during the detailed design, including FEA. For the conceptual designer, however, the main objective is to size the structural components for the predicted loads and determine the weight of the structures that will influence the center of gravity. Until the design of the outer mold line of the aircraft is frozen, any detailed structural arrangement might drastically be altered. OpenVSP Structural Analysis Module (VSP SAM) [25] is a tool which prepares a structural arrangement developed in OpenVSP for processing by Calculix [26], an open-source FEA code. Due to the complexity of integrating with VSP SAM, and the computational load associated with FEA, it was ultimately passed over in favor of simple hand-calculations for a single spar in bending. This method provides valuable stress and deflection data for sizing the structures, while keeping the design process simple enough to be accomplished during the very early stages of conceptual design. The method is described in detail in Chapter 4.

2.2 Automation and Multi-Disciplinary Optimization

Because of the massive economic benefits of even marginal increases in commercial aircraft performance, there has been no shortage of research done in automating the aircraft conceptual design workflow to support hands-off computer-driven optimization routines. Commercial and distributed open-source software have flooded the market, and much research has been done to increase the capability of such tools. Understandably, the main focus of developers has been full-scale aircraft, mainly airliners. Roskam's Advanced Aircraft Analysis (AAA) is geared toward both students and preliminary design engineers of small subsonic civil, military, and transport airplanes [27]. Aircraft Design Software (ADS) includes students, professionals, and home-built designers in its target audience and includes the capability to export the designed model as an XPlane Flight Simulator file to test the design [28]. PIANO focuses not only on the automation of these analyses for preliminary design, but also for evaluating competitors where little detailed information is available [29]. Like ADS, it also offers environmental emissions assessments, which is a key factor in airliner design. Each of these software generally follow the same workflow shown in Figure 2.1, accepting design input and providing automatic analysis output from separate modules representing each analysis discipline.

CEASIOM is a similar package automating the aircraft design workflow, developed by the German Aerospace Center (DLR) and the Royal Institute of Technology (KTH) as open-source software [30]. CEASIOM is a physics-based analysis framework which integrates tools including SUMO for CAD, TORNADO for low-Reynolds aerodynamics, a Reynolds Averaged Navier Stokes flow simulator for high-fidelity CFD, a six degrees of freedom flight simulator, and Neo-CASS for aero-elastic solutions. CEASIOM is very powerful in the sense that it encapsulates all the design data in a single file format, CPACS (Common Parametric Aircraft Configuration Schema). This file format allows distributed design of the aircraft by allowing all tool-specific data to be captured in a single file, allowing splitting and merging of CPACS data sets to facilitate concurrent design, and keeping track of convergence criteria and even discretization parameters to eliminate duplication of work [31]. While CEASIOM is geared more towards the detailed design phases, including controls and aero-elastics, VAMPZero is a companion software also developed by DLR utilizing the same CPACS framework for the conceptual design. By sharing the CPACS format with CEASIOM, VAMPZero is able to compute a bridge function between identical analy-

ses performed by its own low-fidelity codes and CEASIOM's high-fidelity codes [32]. This bridge function, periodically updated during an optimization routine, allows variations of the design to use the low-fidelity results combined with the bridge function to approximate the high-fidelity solution. This variable fidelity approach has been shown to converge to the high-fidelity solution using significantly less executions of the high-fidelity code.

While the development of CPACS came from the top-down approach of taking aircraft design and breaking it down into single data objects in the form of point lists, an independent effort was made to define an XML markup language for aircraft design (ADML) to store design and analysis data. ADML started with a bottom-up approach by constructing mathematical objects and functions to enable unconventional aircraft configurations beyond the tradition ones which governed the development of CPACS [33]. Thus ADML serves as a simple yet very powerful method for representing the entire aircraft in a portable format. Another aircraft design package called MADE (Multidisciplinary Aircraft Design and Evaluation), utilized a database structure to store design and analysis data [34]. Using the Oracle database management system (DBMS), the entire design configuration and all design parameters could be saved and subsequently accessed from anywhere. This architecture, while supporting distributed computing, upheld the same serial single-user paradigm as the other tools mentioned.

Iqbal and Sullivan have created a spreadsheet-based program which integrates high-fidelity analysis tools into the conceptual design of small unmanned aerial vehicles. [35,36] This program was used by students participating in the AIAA design-build-fly competition which is similar to AerosPACE in terms of the required design work, scale, and methods of propulsion. The program attempts to close the gap between detailed and preliminary design by integrating high-fidelity tools in a way that is practically applied to design methods such as Pugh's Method of Concept Generation and Selection. Examples of the tools included are Catia for CAD, Fluent for CFD, and ANSYS for FEA. While higher fidelity tools are excellent for analyzing a single design or a small number of related designs, they are inefficient at evaluating a large volume of concepts, especially when lower-fidelity tools are sufficient to make informed design decisions. The increased time gap between input of design parameters and the desired output stifles the designer's creativity. Al-Shamma and Ali recognized the need for students to be connected to the design process while using such software so that they understand and are taught necessary principles of aircraft design [37].

Almost immediate feedback from the design software is required in order for students to register the influence of the change in input with the change in the response. Raymer's RDS addresses this need by focusing primarily on the education of students, relying almost solely on hand-calculations contained in classical aircraft design textbooks [38]. Yet again, however, his software is targeted towards full-scale civil and military aircraft, and is unsuitable for the design of small-scale, electric-powered UAVs.

Apart from automation and optimization frameworks specifically geared toward aircraft analysis such as those already mentioned, there are a number of more general commercially available frameworks which provide a more flexible and customizable route toward automating portions of the conceptual design. Dassault Systems' Isight is one popular solution [39], as are PHX Model-Center [40] and Noesis Solutions' Optimus [41]. Each of these are capable of capturing a workflow and iterating over that workflow during an optimization routine. Optimus was chosen for use with CCADE due to their interest in the research of this thesis, willingness to donate licenses, and offering personal assistance in using the software. The workflow template provides the information to access data, store it in a persistent location, execute various computer-aided engineering (CAE) tools, and post-process the output [42]. The learning that results from sensitivity analyses performed by such software are extremely valuable, not to mention time-savings by automating this processes rather than conducting them manually. Sensitivity studies and design explorations run by Optimus in CCADE are discussed in Chapter 5 of this thesis.

The advantage of these software is in eliminating the need for the designer to prepare input files for several different software tools, transfer relevant files to a central place, and decipher the various output files each software produces. These often tedious tasks may consume a significant percentage of the designer's time during the conceptual design phase. In a collaborative environment, repetition of these tasks by many members in a design team can significantly slow down productivity.

2.3 Collaborative Engineering

Engineering has always been a collaborative science and collaborative engineering itself the subject of much research. Lu et. al. suggest that mathematical or numerical models alone are inadequate to deal with the "dynamic socio-technical subjects" in engineering, which are often

subject to group prejudices or individual preference [43]. There is a need for a "shared reality," where engineers on a design team come together with stakeholders to co-create. This is poorly accomplished when PowerPoint slides are the only medium of collaboration. Hammond et. al. point out a number of weaknesses of design teams, especially distributed or so-called "virtual" teams [44]. One of these weaknesses involves the common misperception in virtual teams that communication is a "channel or medium" for decision-making rather than the absolute core of such. Teams must formulate the design problem, brainstorm alternatives, analyze and evaluate the ideas, and then select one of them to move on to preliminary design. While most of these activities are highly collaborative, analysis has typically remained a task completed independently. This gap in collaboration during the analysis step can have detrimental effects on the teams creativity and momentum.

Since the advent of computers and engineering software, engineers have moved from the drafting table in an open floor plan to individual cubicles, making collaboration much more difficult. The dispersed nature of many large aerospace companies makes collaboration even more difficult. There are a host of freely available communication tools that utilize the internet to make collaboration easier, but engineering software itself has been slow to adapt to the global collaborative paradigm and create a "shared reality" among design teams. NX-Connect, an extension of CAD software developed by the v-CAX Lab at Brigham Young University, is one attempt to address this emerging paradigm that allows synchronous, multi-user edits to a single CAD model or assembly. [45, 46] The use of NX-Connect during preliminary and detailed design in the AerosPACE course is the subject of ongoing research [47, 48], and many of the lessons learned have been applied to the development of CCADE.

Because distributed teams can potentially combine the best technical expertise and unlock the creativity that stems from diversity, it is important to create an environment that facilitates collaboration. Johansson et al. linked analysis modules together in a Service Oriented Architecture (SOA) to allow distributed execution through a web-based message service along with integration of airplane design disciplines [49, 50]. Distribution of computational resources is quite easy with today's high speed data transfer rates, and a workflow may be divided up and passed along to dispersed technical experts, but work must still be done to truly bring multi-disciplinary teams together and take advantage of their diversity from a system-level during conceptual design.

Product Lifecycle Management (PLM) software is typically used in large aerospace companies as a base for their collaborative efforts, but actual use is often limited to the product data management (PDM) functionalities such as data storage, version control, notification, and tracking [51]. It has been unable to fully take advantage of the sophisticated knowledge models of multi-disciplinary engineering. PLM is also much too complex for students unlikely to generate enough data to make full use of the software. Rather than saving time or making things easier, it becomes just another tool to learn.

2.4 Benefits of CCADE

While programs mentioned in this chapter are helpful for professionals and students learning to design full-scale commercial and military aircraft, students also benefit greatly from active learning courses such as AerosPACE where they are exposed to manufacturing methods and required to actually construct a UAV [5, 52]. This, along with the fact that UAVs are becoming much more popular and useful with recent advents in electrical and communications technology, suggests a need for an automated conceptual design program tailored to these types of vehicles. A method for storing and accessing this data among multiple clients is also required. CCADE is presented in this thesis as filling these gaps in the current market.

CCADE allows the designers to access the results of multiple analysis software without requiring coding expertise or experience with the specific software interfaces. By providing quick and easy access to the most relevant output data, students are taught which outputs to look for and how key variables influence each output. The emphasis on design rather than coding different analysis tools. CCADE attempts not only to create a common repository for design information, but to construct a “shared reality” between AerosPACE student engineers where they can discover themselves the best practices for distributed virtual teaming.

CHAPTER 3. ARCHITECTURE

This chapter is devoted to explaining the architecture that enables CCADE to function as a collaborative design software. The architecture was chosen to maximize portability and ease of coding. CCADE is designed as a three-tier client-server system consisting of database, server, and client. The database contains the critical model data, the server provides read and write access to the model with data validation, and the client displays this information to the user in an intuitive way. This architecture can be broken down to the back-end, the front-end, and the bridge between the two. These components, and their interaction, are shown in Figure 3.1. The back-end refers to the database model architecture, at the top of the diagram, which receives and stores model specific data in a predetermined format with built-in dependencies and logic necessary for defining both the design data for the UAV, and the user data which assists in collaboration. The front-end is the client graphical user interface (GUI), at the bottom of the diagram, which is the portal for the user to both view data contained in the database, as well as manipulate the design and interact with team members. The bridge between the two consists of services, contracts, and both client and server code which enables communication on secure lines, ensures that data adheres to predetermined rules, and is successfully transmitted over the web. This is shown in Figure 3.1 as the “Server”, and includes the methods by which information travels via the arrows from each block. This particular architecture, though unique in its application, is related to and derived from proven methods in software engineering and information technology [53].

3.1 Database Model Architecture

In order to store UAV design information in a distributed environment, and to facilitate the collaboration between multiple clients, a centralized location for the persistent data was required. In this section, the actual database structure used by CCADE is presented. Then the database type

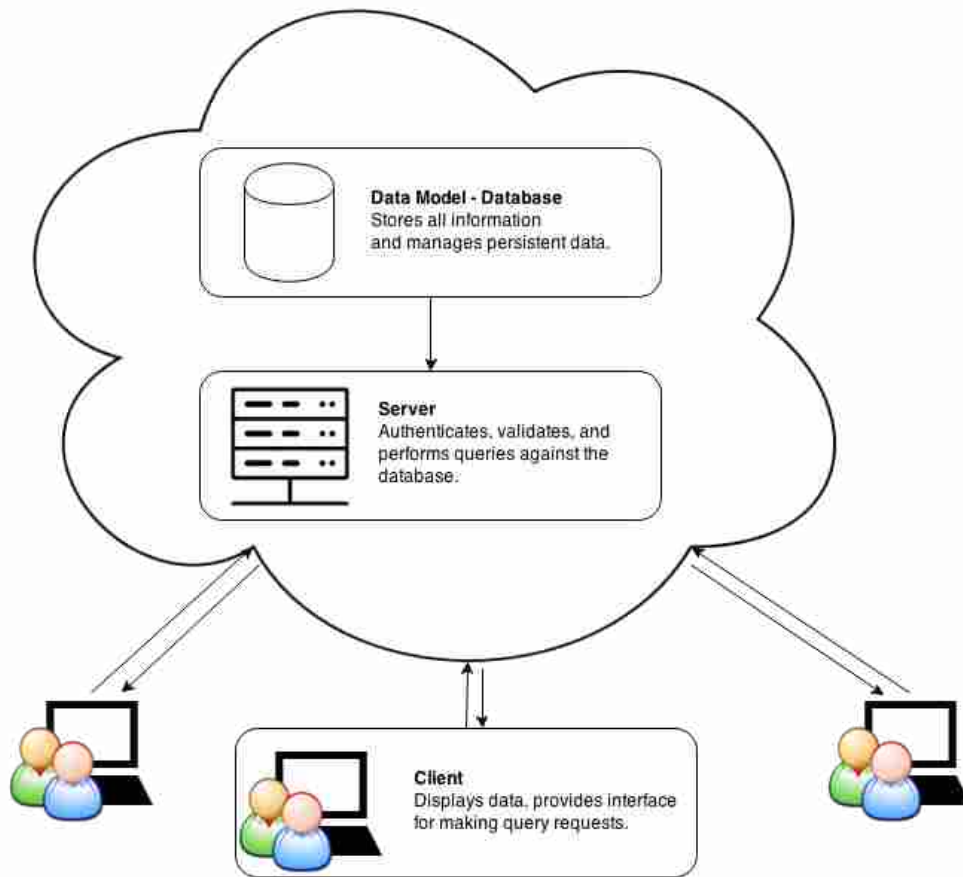


Figure 3.1: The structure for the collaborative design cloud. Information flows upward through the cloud to be stored in a centralized database. Users receive continuous updates on values changed via queries to the team database, allowing them to work collaboratively on the designs.

and database management system (DBMS) are explained. Finally, the framework used to create the database is introduced.

3.1.1 Database Structure

The data model of CCADE is object-oriented. A certain product (a UAV) has properties and components. Its components also have properties, and may have subcomponents. The collection of the subcomponents, components, and their properties constitute the complete definition of the UAV design. Each of these components is referred to as an entity. Figure 3.2 shows a non-exclusive set of objects belonging to a UAV design. Beside physical objects such as the geometry, motor,

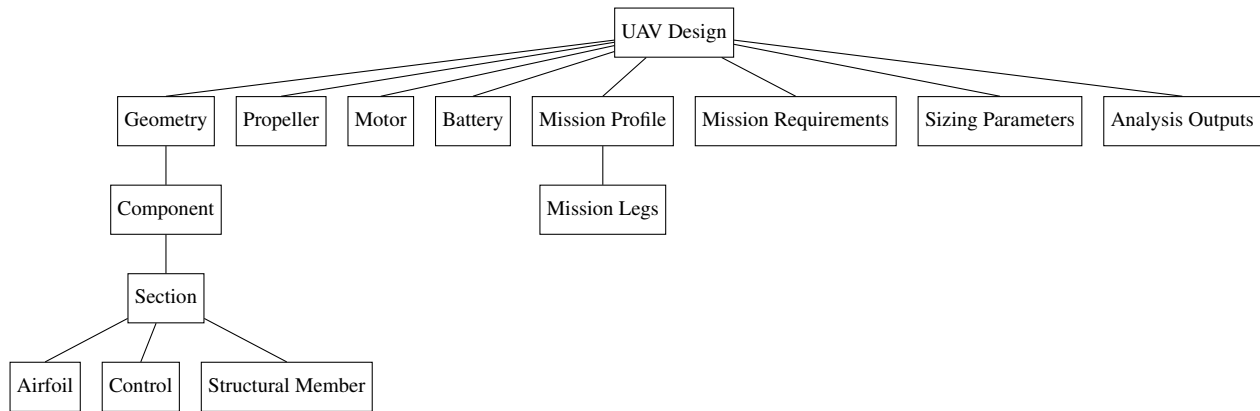


Figure 3.2: Diagram of partial object-oriented data structure.

and propeller, concepts and analytical data can also be represented as objects. Each instance of an object is given an identifier to distinguish it uniquely from all other instances, known as a primary key. Different types of relationships exist between entities. Such relationships include:

- *One-to-many* (1..∞) - The most common type of relationship. For example, a wing may have any number of sections, and a UAV any number of wings.
- *One-to-one* (1..1)- This is not as common. Each entity has one and only one of the other entity, similar to an additional property. For example, a wing section may only have a single control surface assigned to it, and each control surface belongs to only one wing surface.
- *Many-to-many* (∞..∞)- Helpful when an instance of an entity is non-uniquely related to multiple others. CCADE does not include these types of relationships.

Each relationship in Figure 3.2 is either *one-to-many* or *one-to-one*. The complete listing of entities, their properties, and their relationships is given in Appendix A. The relationships are contained within each entity by use of foreign keys. A foreign key would be a field in an entity which refers to the primary key of another entity. In this way, one can define parent-child or belonging relationships.

Along with showing primary keys for identification and foreign keys for relationships, Appendix A also lists each entity's properties. Each property or field of an entity has a specific data type which gives the format for the data that may be stored. The data types used for the CCADE structure include:

- Integer - whole numbers
- Float/Real - numbers with a decimal value
- NVarChar - a set of alpha-numeric characters in a certain order
- Bit - a boolean value, “True” or “False”
- VarBinary - a set of binary code, used to store files directly in the database

Using these data types, all information pertaining to a certain object can be represented. Within the geometry of the aircraft, for example, each wing would have an identification number, stored as an integer. Each wing would have a planform area, span, and possibly dihedral associated with it, which might all be stored as real numbers. Each wing might also have a name, stored as an NVarChar or string value. There might also be a control surface on the wing, but one would only know if there were a flag answering “true” or “false”, stored as a bit. Once an aerodynamic analysis had been run analyzing the wing, one would need to upload the output file in binary to the database, where it would be stored as a varBinary. The data type ensures that all data written to the database conforms to the model established in Appendix A. There is enough generality and flexibility built in to the model, however, to allow a great range of creativity for the designer.

3.1.2 Database Management System

A database management system (DBMS) is a tool for creating and managing databases. It includes the actual data, the database engine or methods for accessing, locking, and modifying the data, and the schema, which defines the relational structure of the data. There are numerous DBMS freely and commercially available. The one utilized by CCADE is Microsoft SQL (Structured Query Language) Server, which was already established on a university server with a public IP address. This meant that much of the overhead of obtaining hardware for the database, installing the DBMS, and obtaining a public address on the web (for students outside the university network domain) had already been taken care of. Microsoft SQL Server uses a simple extension of structured query language (SQL) called Transact SQL (T-SQL). Using this language, one may easily create or delete databases, entities (as SQL “tables”), procedures, entries within a table, and

a host of other data manipulation tasks. The command is constructed in SQL language, and then “committed” to the database. Using a transaction technique, the DBMS will execute the command, and if the command fails, can attempt to “roll-back” the transaction by undoing any changes made. This is especially helpful if, for example, a student enters an unacceptable value for a property which will cause another operation to fail. Because of its simple language and optimized SQL commands, the DBMS can execute operations with incredible speed. Even queries from multiple users at the same time in a scaled system can be easily and rapidly handled. Overall, this DBMS offers excellent efficiency in terms of serving requests as well as storing data. Because all of the data is stored in primitive forms, as was shown in the previous section, the DBMS can transfer the data very quickly over even relatively slow connections. This is important so that there is no lag on the client side when requesting data.

3.1.3 Data Model Framework

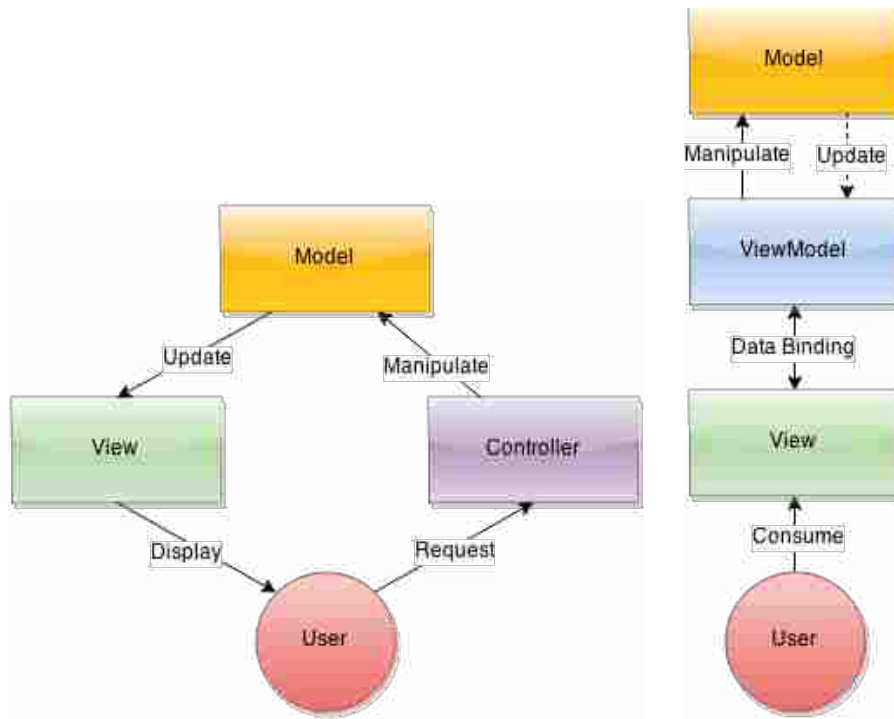
Unfortunately, the efficiency gained by storing information in a database comes at a cost. The DBMS commands, which must be written in SQL, are cumbersome to include in a separate coding language. Since the entire client program cannot be written in the same way the database is, there must be a way to translate between the client program code and the database code in order to perform queries, receive information, and store it locally in memory. Just as an object-oriented approach was taken in structuring the database, a similar approach must be taken to store these objects in memory on the computer. Entity Framework is a data access layer (DAL) from Microsoft to map conceptual data models in a Microsoft .NET coding language to the physical data stored in the database. It automatically generates what are called Plain Old CLR Objects (POCOs) for each entity in the database model. POCOs are very simple storage objects which are easily serialized. This is imperative for transferring the information over different layers via a network connection. Entity Framework will map database fields to properties of the POCOs, as well as handle inheritance and other relational dependencies. It automatically generates code for securely accessing the database, performing queries against the database model, and serializing data for transmission (or deserializing received data). These queries may involve creation of an entity, editing a field, deleting an entity, or otherwise requesting information about the entity or multiple entities.

One of the major advantages of using Entity Framework is the ability to graphically build a database model and define entities, their properties, and relationships between them, and then automatically have a SQL script generated which can be run to create that database. This is a major time-savings in development, when the database model is frequently augmented to accommodate unforeseen entities or properties. Another advantage is the automatic incorporation of inheritance in the database model. Inheritance is a principle whereby objects may be treated as a type of a super-class, possessing all of its properties and functions, while also including additional properties or functions specific to its own designation. For example, when classifying legs of a mission profile, several of the flight segments have the same properties, ie. duration of flight and altitude. They can be classified as a generic air segment, and then define their own specific properties such as flight velocity, or number of turns in a maneuver. This reduces repetition of code, and allows for polymorphism. Polymorphism allows a function to be called on any class that inherits the abstract function from the parent and implements it uniquely. Each class may behave differently due to the unique implementation of the function, despite the fact that the function call was identical for each instance. Inheritance is a behavior of the coding language, and is not inherently a part of a relational database. Entity Framework manages the relations in the database for the developer, so that objects are placed correctly into tables defining their properties and inheritance relationships.

3.2 GUI Architecture

In order to provide an intuitive, user-friendly interface to the design tools and data available through CCADE, the GUI architecture must support binding to the data model. It is desirable that the GUI be agnostic with respect to the model. The assumption is that both the model and GUI (also referred to as the “View”) are free to change, without a subsequent requirement that one be modified in order to accommodate changes in the other. This thinking has led to several different philosophies regarding how to achieve this separation. Two popular philosophies are the Model-View-Controller architecture and the Model-View-ViewModel architecture. Diagrams of each of these architectures are shown respectively in Figures 3.3a and 3.3b.

In both architectures, the job of the View is to display information to the user and provide an access point to manipulate the model. Any changes made, and all requests to access data, pass through a distinct layer. In Model-View-Controller (MVC), all requests are passed through the



(a) The Model-View-Controller architecture. (b) The Model-View-ViewModel architecture.

Figure 3.3: Diagrams of the MVC and MVVM GUI architecture models.

Controller; the Controller has predefined protocols and potentially additional logic to validate the input to ensure that the database does not encounter any errors during the transaction. It also is equipped with authentication to make the changes, so that this authentication information is also independent of the View. The Model then registers the change, and updates the View.

The Model-View-ViewModel (MVVM) is similar to MVC, but the Controller is replaced with a ViewModel. Rather than directly connecting the Model data to the View, all information passes through the ViewModel. The ViewModel can contain additional code that formats the data for consumption by the View. In addition, rather than exposing the View to Model-specific actions, the ViewModel contains the function definition for any actions the user may wish to take, which are implemented in the View. The ViewModel has the special property of being able to detect when changes have been made, and can react accordingly; this property is referred to as data binding. By binding a field on the View to a property in the ViewModel, there is no need to actively execute the update; the update is made as soon as that change is made and validated. This is accomplished by making the ViewModel implement the interface *INotifyPropertyChanged*.

Then, in the definition of properties, an additional line is added when a value is set to trigger the *OnPropertyChanged* event, which sends a notification to any control registered to the event. The CCADE GUI implements the MVVM approach, mainly to exploit the benefits of data binding in connecting the View to the Model data. For speed of development and easy implementation of the object-oriented structure, C# in the Microsoft .NET Framework was chosen as the language for coding the ViewModels.

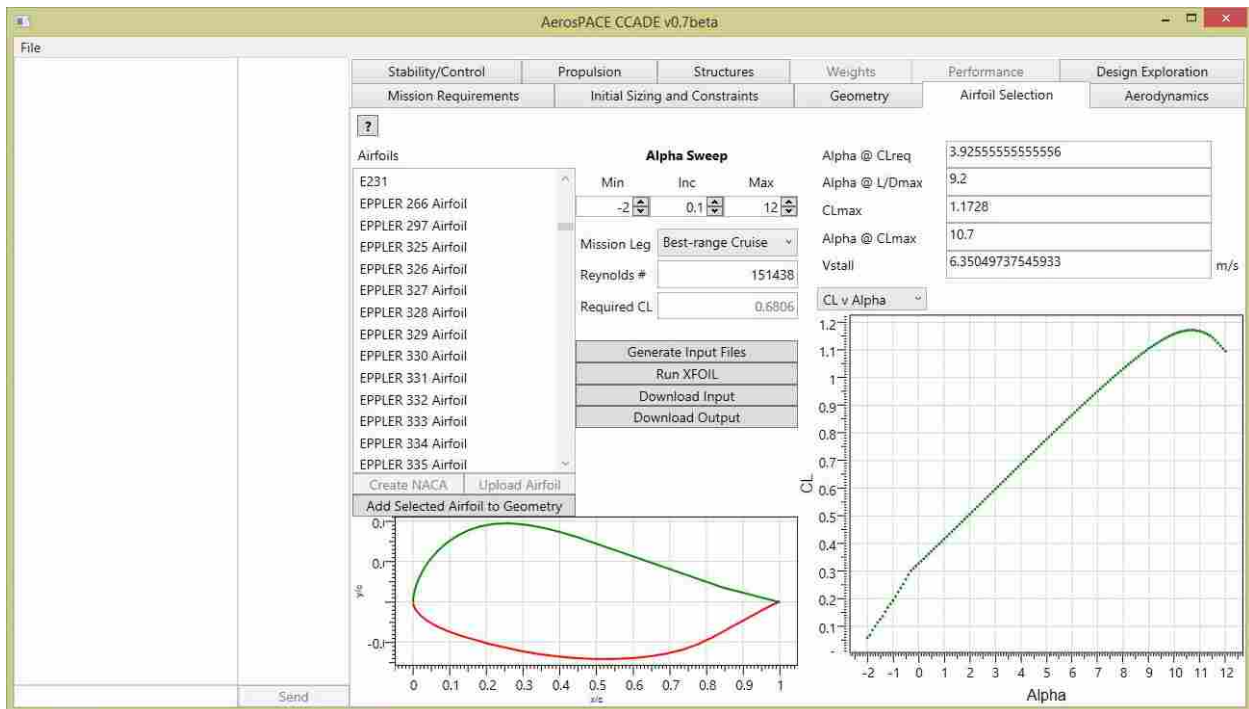


Figure 3.4: A screenshot of the graphical user interface of CCADE. The graphs shown can offer zooming and navigation capabilities, and all data is linked directly to information located on the database.

The MVVM pattern of CCADE is enabled by the capabilities of Windows Presentation Foundation (WPF). WPF is written in Extensible Application Markup Language (XAML), which can interface seamlessly with the C# code of the ViewModel. This is done by assigning the ViewModel as the DataContext of the View. The XAML View will create binding relationships with the ViewModel, and then send and receive notifications when objects are changed. In this way, the user is consistently connected via the ViewModel to the Model data. WPF also easily provides a modern look and native support for three-dimensional view-ports which can be appreciated by

students expecting a clean, finished appearance of the GUI. An example of the graphical user interface is shown in Figure 3.4. A chat box (inactive in the current version of CCADE to reduce errors) for users to communicate with each other is located on the left, with the main design area broken into tabs according to each analysis discipline. The module shown in Figure 3.4 is the Airfoil Selection module, providing several graphs for output and visualizing the airfoil, and areas for design information and analysis specifications to be input.

3.3 Client-Server Architecture

In order to bridge the front-end code with the back-end database, there must be a server program located on the same hardware as the database to interpret requests, perform actions, and return results. The client must likewise be equipped to send messages to the server in a format that can be serialized and are sufficiently annotated so that the server can interpret them after transmission over the network and then execute correctly. This section will describe how this is accomplished with CCADE first by addressing the Windows Communication Foundation (WCF) services which run on the server-end and their function, and then describing the framework which was utilized to create these services. This framework also assisted in the creation of contracts used by both client and server for transfer of information over the network. These two topics encompass the three-tier client-server architecture of CCADE, which enables students using installations of the client software on their individual machines to collectively access the same design and collaborate more effectively.

3.3.1 WCF Services

WCF is a framework for programming an application which accepts asynchronous messages and responds as a service to these messages. Simply put, the application waits and listens for the messages without knowing when they might come, and is designed to respond in a certain way. There are many different ways of hosting WCF services, but for simplicity all of CCADE's services were hosted via Windows Internet Information Services (IIS). A dynamic-link library (DLL) for the service application is created, and referred to by an SVC file. When located in the same folder along with all necessary dependencies, this is all that is needed for IIS to begin hosting the service.

Figure 3.5 shows a screenshot of IIS with the service application (Aerospace.CCADEServices) listed in the tree on the left, and the configuration of the service in the center. The configuration receives as input a path on the machine where the service is located, and provides a virtual path for client machines who connect via the public IP address of the server and appropriate port. The protocols enable different levels of security and other features for transmitting information. Setting up the service in IIS creates an endpoint, which can then be referred to by a client to establish a connection. The client must also have a defined endpoint in order to receive information from the server. If the client is outside of the server's local network, the server must also have an IP address that is registered on the public domain. Two services for CCADE are addressed in this section: the data service which handles all of the queries to the database, and the change notification service which informs clients of changes made by other users within the same design.

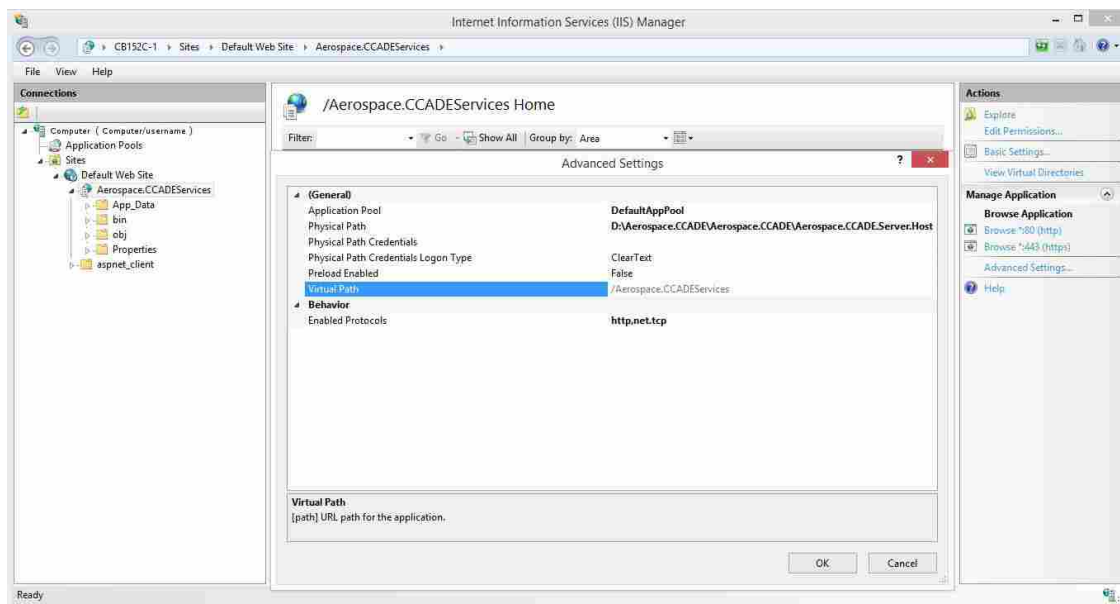


Figure 3.5: The dialog of IIS for hosting a WCF service. The default web site is configured is configured with open ports for communication, and can be accessed from outside the network with a IP address on the public domain.

Data Service

The purpose of the CCADE data service is to accept requests from clients for querying the design data in the database and making changes. This service includes the libraries for Entity

Framework, which will communicate with the database via the generated POCOs. It has functions defined for executing SQL procedures on the database, and repositories which store the data gathered from the database in the server memory for transfer over the network. The server endpoint is configured with WSHTTP binding to accept requests using the HTTP protocol and does not require authentication by the client. This allows free distribution of the software to students and coaches without additional network authentication configuration. When using CCADE, the client connects to the server via HTTP to request a read or write action of the server. The server checks whether the user is authenticated, and in the case that the action is to write to the database, validates that the type and value of the input meets database requirements. If this succeeds, the server performs the action against the database and updates the client. If not, an error message is displayed for the user indicating the failure.

CCADE is run on the user's PC as a thick client, conducting the computationally expensive analysis tasks on the client machine. The data service fulfills the task of storing results on the server, which in comparison is almost instantaneous. This scheme makes information almost immediately available for all the users within the design without redundant processing which could also potentially cause consistency issues. All data that has been queried from the database gets stored in the client's local memory in the form of a database *DataContext*. The client only uses data stored in the DataContext, so it is imperative that the DataContext contains the most up-to-date values stored in the database. In the case of data that the user is not currently viewing and editing within their analysis module, the DataContext is disposed of after every query, forcing a new DataContext and a new query every time the data is used. The case of multiple users viewing and editing within the same analysis module is addressed in the next section. The data service is designed to handle messages synchronously, gathering multiple requests in a queue and handling each one by one. For simplicity, the last edit is allowed overwrite all previous edits if they are made approximately at the same time.

This method of data transmission requires that the data sent across the wire to the database and back to the client machines be relatively small. If this were not so, it would be possible for bits of data to be lost over the connection and corrupt the model. Furthermore, users on slow connections would have their programs locked for unreasonable times waiting for an upload or download to complete. Since the number of query events is large, the size of these downloads

must be minimized. Thus, the responsibility for generating images, plots, and other complex data is delegated to the client machines. For example, each time the geometry module is opened, the client queries the database for the geometric information, then executes OpenVSP to generate a three-dimensional geometry file that can be viewed and manipulated on-screen. This file would be rather large and take far too much time to download from the database each time, but the relatively small OpenVSP file, a simple text XML document, is able to be quickly interpreted and converted client-side to provide instant updates for the team each time a design change is made.

Change Notification Service

CCADE is structured so that only visible data is locally stored on the client. When any new data must be displayed, or when data outside what the user can visibly see and manipulate is required to conduct analyses, it must be queried from the database. This eliminates many issues of inconsistencies between users working in parallel on the same design. However, it creates a problem when two users are viewing and editing the same design data simultaneously. The CCADE change notification service is intended to solve this problem by providing instant updates to each client's display with changes made by any other client.

Each user is required to register with the change notification service upon opening CCADE and after switching to a new analysis module. The server keeps a list of which analysis module each of the logged-in users are actively viewing. Since there is no need to update one client if he or she is the only one viewing the data in question, the service only sends an update notification to other users currently in the same analysis module. The service receives a message every time a client changes a value that is stored persistently on the database. The message consists of the type of entity changed (which indicates what table corresponds to the entity), the identification (ID) number of the entity, the type of action performed (whether it is an added value, deleted value, or modified value), and the user ID that made the change. The latter is intended for future versions which may track modifications and evolution of the design including how teams work together using CCADE. Once the message is received, the server broadcasts that same message to the other users in the analysis module as an argument to a callback function executed on the client machines. These broadcast messages are by necessity not solicited by the clients, and thus require a special protocol in order for the clients to accept the message. CCADE uses .NET TCP protocol for the

change notification service. The callback function, which is defined in the client code, instructs the client to detach the entity from the DataContext if it exists, and then query the database once again for the new value (unless the entity has been deleted). This way, the old data is orphaned (and subsequently collected by the program's garbage collector) and the new entity replaces it with a new value from the database. Once the client's DataContext is updated, a property-changed event is fired to the user interface, which updates the display with the new value.

This service currently works in CCADE for computers within the same network as the server, but not for out-of-network machines. The service currently omits sufficient addressing of the client machine to direct the message out-of-network. A solution to this issue was unable to be found in time for the students to make use of it during the 2014-2015 AerosPACE course, but is recommended for future work. The current workaround requires users to exit the analysis module they are using and re-enter it for the changes made by their teammates to be reflected.

3.3.2 N-Tier Entity Framework

Section 3.1.3 introduced Entity Framework as a mapping strategy for connecting the physical data stored in the database to the conceptual data model of an object-oriented coding language like C#, used in CCADE. While Entity Framework is useful when the application and database are located together, a great deal of the simplicity and ease-of-use is lost when attempting to transfer entities over tiers. As previously mentioned, CCADE operates with three tiers: the database, the server, and the client. Normally, Entity Framework is only located on the server, to keep the client from directly accessing the database and keeping business logic isolated from the client application. N-Tier Entity Framework, an open-source framework distributed under the Apache license, provides an API similar to Entity Framework on the client side to provide the ease-of-use, as well as generates service contracts, repositories, and other necessary service code including querying functions [2]. As illustrated in Figure 3.6, it fits in well with the MVVM GUI architecture explained in Section 3.2.

N-Tier Entity Framework comes with scripts which automatically arrange the code into projects which are distributed singly to the server and client, and others which are distributed to both server and client. Once the Entity Framework data model has been defined, it generates fully annotated entities which contain in a serializable form all the necessary metadata to easily use

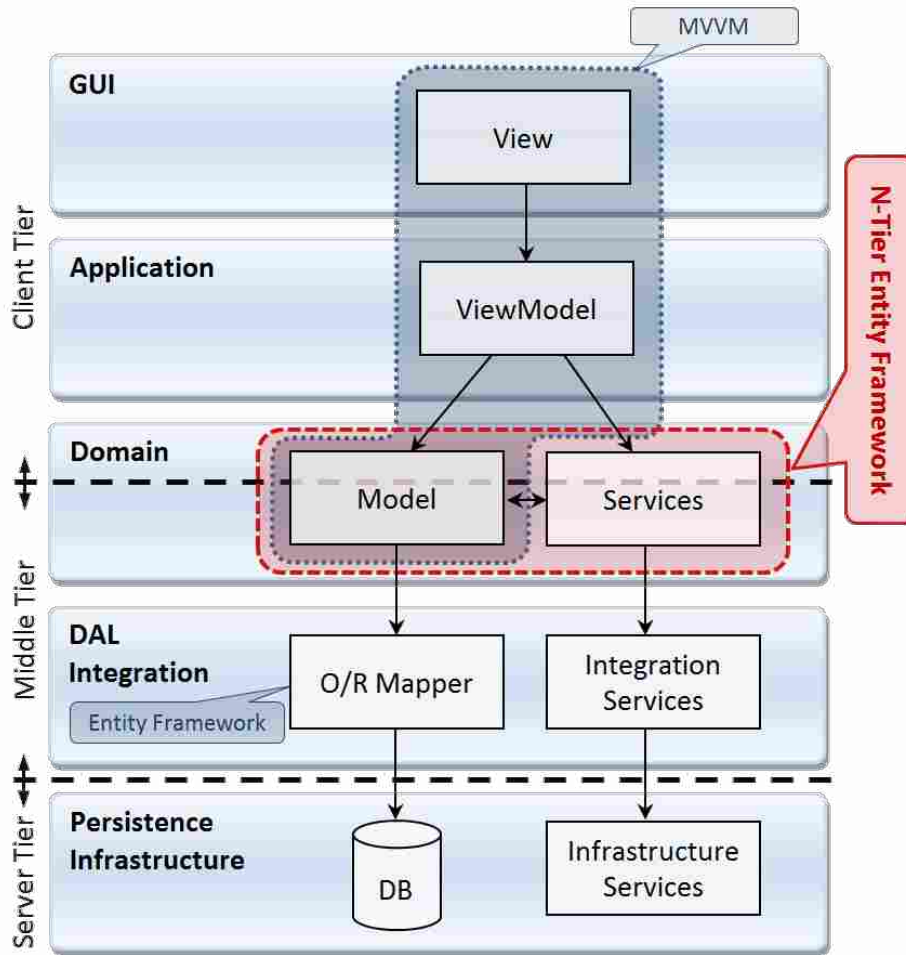


Figure 3.6: A schematic of N-Tier architecture, adopted by CCADE, with the role of N-Tier Entity Framework highlighted. No infrastructure or integration services were necessary for CCADE. Image courtesy of [2].

on the client-side. It also generates a WCF service implementation for the CCADE data service, WCF service contracts to ensure proper communication between client and server, and client DataContext definitions which provide functionality similar to having Entity Framework located on the client machine. This helped define the architecture for CCADE, and increased the speed of development of each of the tiers.

CHAPTER 4. WORKFLOW

An in-depth explanation of the methods used by CCADE will be given in this chapter. CCADE is considered both a design and analysis tool. It acts as a facilitator for completing design activities, stores design decisions, and also provides access to methods for analyzing the design and sharing the results with the team. This chapter will review the design features available in CCADE, as well as the analysis methods corresponding to each discipline. The organization of the chapter follows the ordering of the analysis modules within CCADE, roughly following the direction of conceptual design as presented by Raymer in [1]. The CCADE workflow, presented visually in Figure 4.1, is a modified version of Raymer's, shown in Figure 2.1, eliminating elements specific to fueled and manned aircraft, as the unmanned vehicles designed in the AerosPACE course are battery-powered. Analysis tools to accomplish the calculations were chosen for their low computational load and fast execute speed, so that feedback to the students is rapid. In order to allow students to run CCADE on a personal computer and deploy analysis tools in a single software package along with the collaborative design environment, all of the tools chosen are released under licenses allowing open distribution and modification of the original source code (including the GNU General Public License and the Apache License), or otherwise commonly available to students running a Windows 7 or higher operating system.

As long as command languages and file formats are not changed, CCADE is fully compatible with new versions of XFOIL, AVL and OpenVSP. AVL and XFOIL are no longer under active development, so no drastic changes are expected for any future versions which may be released in the future. OpenVSP version 3 made changes in file format which rendered its models incompatible with CCADE. This incompatibility was rectified during the development of the design exploration proof-of-concept described in Chapter 5, and integration with the geometry module is a point of future work.

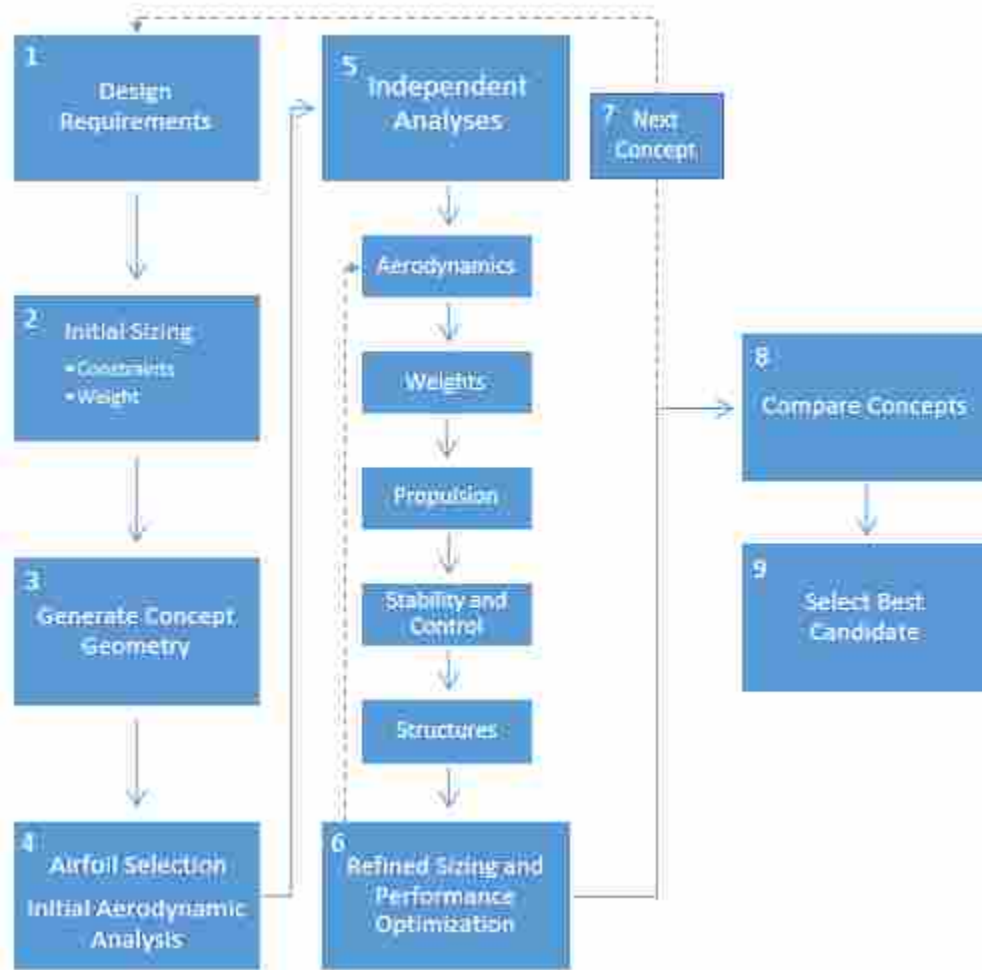


Figure 4.1: The workflow for CCADE, adapted from Raymer. Concepts are generated, sized, sketched, analyzed, and optimized automatically according to user input of mission requirements and mission profile. The concepts are compared according to selected criteria and the best design selected.

Following the illustrated CCADE workflow in Figure 4.1, the analysis modules in this chapter are presented in the following order: Mission Profile and Requirements, Initial Sizing and Constraints, Geometry, Airfoil Selection, Aerodynamics, Stability and Control, Propulsion, and Structures. Analysis modules for Weights Analysis and Overall Performance are planned for future versions of CCADE. The Design Exploration module is discussed separately in Chapter 5. As a brief summary of the process, the user first selects certain mission requirements by defining a mission profile in CCADE. Once these are defined, and an initial concept of the vehicle is already in the mind of the designers, the UAV may be sized according to estimates of key parameters based

on historical values of similar aircraft, and a knowledge of the payload. Once the sizing operation has yielded a wing area and take-off weight, the concept geometry can be lofted in OpenVSP, which will provide a complete definition of the geometry and reference values for future analyses. Airfoils are then analyzed for best application to the mission, and applied to the wing geometry. Aerodynamic coefficients are calculated with AVL, giving the first glimpse into whether the assumptions made during the sizing analysis are realistic to the configuration. Once these first tasks are completed, the designers may move freely and without regard to the order of execution through the analysis modules, for all the requisite data is present, and any new analysis will have sufficient information to execute properly. Stability and control analyses can be performed to size control surfaces and ensure adequate static stability. The propulsion module allows students to input actual propulsion system components and match them together to determine their combined efficiency. The structures module serves as an introduction to sizing the structural members of the UAV, analyzing a single spar in the wing components of the aircraft. The weight and balance module ensures that the center of gravity is in the appropriate position for stability. The final performance analysis either assures that the design synthesized to this point meets all mission requirements, or suggests that a relaxation of certain mission requirements could provide significant improvement to the design.

CCADE walks the user through each of these steps, and provides the environment to accomplish these design activities collectively as a team. As previously mentioned, the process follows step by step through the workflow shown in Figure 4.1 until box 5. After this, analyses can be conducted in parallel. This chapter focuses on the theory and methods involved in the analyses; a guide for using CCADE and explaining the graphical interface in detail with screenshots of the program is given in Appendix B. Once a set of concepts has been generated using this pattern, they can be compared with reference to selected evaluation criteria, and the best candidate chosen to move on to preliminary design.

4.1 Mission Profile

The mission profile, the top block of Figure 4.1, serves as a foundation for all other analysis tools, providing a description of flight conditions and design constraints. It is necessary to allow a wide range of mission profiles to be constructed to provide the flexibility for the tool to be used

for a variety of requirements. The approach adopted by CCADE is to provide a set of mission legs which, when put together with other mission legs, constitute the complete mission profile. This approach allows for the introduction of new mission legs as the requirements demand, so long as they are properly defined. As soon as a user creates a new design within CCADE, he is prompted to construct a mission profile by adding to a set of mission legs and specifying key parameters to describe each leg. The types of mission legs available in CCADE, along with their descriptions and the required input parameters to fully define them, are listed in Table 4.1. Because climb after take-off of small electric-powered vehicles does not require large power draw, this segment is omitted. However, a separate rate-of-climb constraint is taken into account during the initial sizing. The profile can be rearranged and modified at any point during the design, with changes propagating downstream without any action required on the part of the user. In this module, no calculations are made, but mission profile data is stored in the database for future use.

Table 4.1: Definition of the types of mission legs available in CCADE, including the parameters necessary to fully describe them.

Mission Leg Type	Description	Required Input Parameters
Take-Off	Acceleration to lift-off via a runway	Altitude, CLmax, Runway Distance, Runway Friction Coefficient
Cruise (at velocity)	Steady flight	Altitude, Duration, Velocity
Loiter (at velocity)	Steady flight, typically at a lower velocity than cruise	Altitude, Duration, Velocity
Best Range Cruise	Optimizing velocity to maximize the lift-to-drag ratio, minimizing drag and maximizing the total range	Altitude, Duration
Turns	Turning with load-factor determined by velocity	Altitude, Velocity, Number of Turns
Landing	Final landing of the vehicle	Altitude, CLmax, Runway Distance, Runway Friction Coefficient

4.2 Initial Sizing and Constraints

Once the mission profile has been entered, the designers may conduct the initial sizing of the UAV (block 2 in Figure 4.1). It is expected that before the team begins using CCADE, they have

held a brainstorming session and created rough concept sketches, which can be used as a baseline for estimates of key aircraft parameters. From these estimates and historical data, the required inputs to the sizing equations such as aspect ratio, friction drag, empty weight fraction, etc. are approximated. These initial guesses and the mission requirements enable spreadsheet calculations for power loading, wing loading, and take-off weight to be performed using worksheets based on classical sizing equations. The equations used in CCADE and presented in this section are taken from works by Mattingly [54], Brandt [9], and McClamroch [55]. A graph of the required power loading versus the wing loading of the vehicle according to the defined constraints is continuously updated for the user, and weight sizing outputs also constantly updating values as inputs change. This section will introduce the equations used for the constraint sizing to build the graph, the equations used in weight sizing, and give examples of the respective output of both.

Multiple Microsoft Excel worksheets with these equations are synchronized with the CCADE GUI in the initial sizing module. By taking advantage of the Microsoft .NET Excel Interop library, a class library was written for CCADE to funnel inputs from the GUI interface to the correct cells in the appropriate worksheet. This relieves the user of building relationships themselves between different worksheets, or keeping them separately updated with the most recent mission profile and concept data. The key outputs, including wing loading, power loading, take-off weight, and battery weight, may then be extracted from the worksheets and displayed immediately to the viewer. The constraints graph is automatically generated in Excel in the background, but CCADE is able to size it appropriately for viewing in CCADE, turn series on or off according to the relevance of constraints (in the case of rate-of-climb and load factor constraints), and export the image from Excel. Outputs in this module automatically update when a change is made to the mission profile data.

4.2.1 Constraint Sizing

The main goal of the constraint sizing is to determine the required wing loading, $\frac{W_{TO}}{S}$, and power loading, $\frac{P}{W_{TO}}$, for the UAV. Wing loading is the ratio of take-off weight to the planform area of the main lifting surface. This is a measure of the relative size of the aircraft; a low wing loading means it is very large for its weight, a high wing loading on the other hand can handle tighter maneuvers. Power loading is the ratio of power provided by the propeller to take-off weight. It

is analogous to thrust loading, for aircraft driven by gas turbine engines which are designed to produce a certain thrust. Since propeller motors are designed to a certain horsepower, it makes more sense to speak in terms of a power loading on the aircraft; a higher required power loading means a larger motor is needed.

Table 4.2: Description of the inputs to the constraint analysis.

Input Parameter	Description
Aspect Ratio (AR)	Span to constant chord ratio, $\frac{b}{c}$ or more generally, using wing area, $\frac{b^2}{S}$
C_{D_0}	Parasite drag, mainly due to skin friction. This value can be approximated with low-fidelity in the Geometry tab. It may be updated when a better approximation exists.
Oswald Efficiency (e)	Represents increase in drag due to lift (induced drag, C_{D_i} , as compared to an ideal wing). It is used to derive the induced drag from a known lift coefficient using the relationship $C_{D_i} = \frac{C_L^2}{\pi e AR}$
Propeller Efficiency	The efficiency of the propeller, defined as the ratio between power out and power in.
Motor Efficiency	The efficiency of the motor, defined as the ratio between power out and power in.
Load Factor	Ratio of lift-to-weight. A higher load factor comes from more demanding maneuvers.
Rate of Climb	The rate of change in altitude, proportional to the excess power available. Specified in $\frac{ft}{s}$

A summary of the inputs used in the constraint sizing analysis is given in Table 4.2. Once the inputs are all present, the calculations can be executed using the wing loading as the independent variable, determining the power loading as a function of the wing loading. The result is an image of the constraints graphed on the same plot, which shows the feasible design space of the UAV. An example chart is shown in Figure 4.2. In this chart, the feasible design space is the area above all constraint series. The current selected operating point (a coordinate containing values for both wing loading and power loading) is shown as a red diamond on the graph. Each constraint series is labeled, so that the user may see clearly what the limiting constraints on the UAV are.

In the CCADE GUI, the user may with the click of a button automatically adjust the wing and power loading to the intersection of constraints to estimate an ideal operating point for wing and power loading. From there, the user may more intelligently select an operating point that is sufficiently conservative and satisfies all requirements, including requirements that are not tracked by CCADE.

The parasite drag used for the constraint sizing is utilized by many subsequent analyses, but can be refined and updated later on once the geometry has been finalized. Initial guesses for

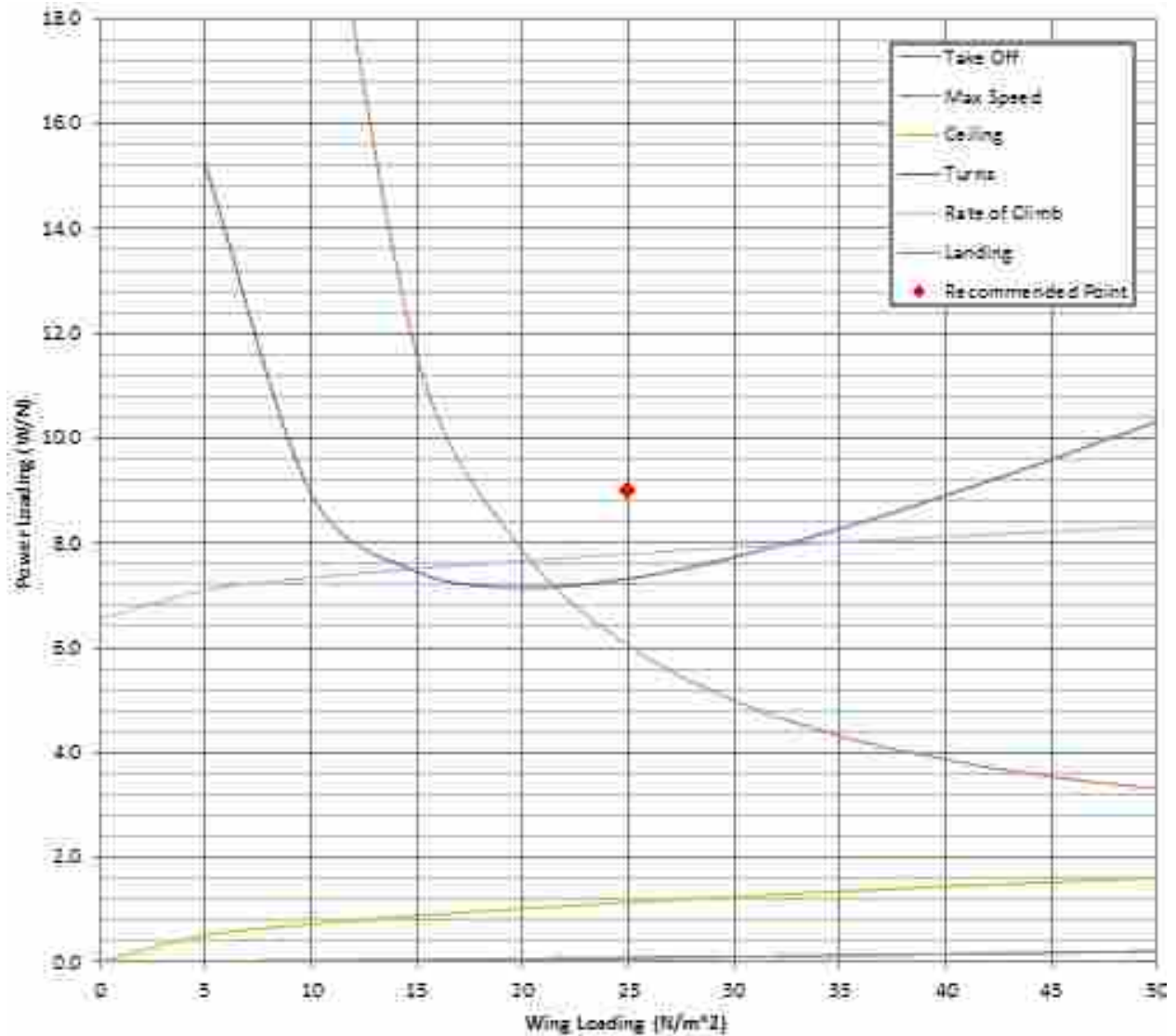


Figure 4.2: An example chart plotting the constraints on power loading as a function of wing loading. Feasible design space is the area above each constraint curve. The selected design point is marked with the red diamond.

motor, propeller, and Oswald efficiency are used until they can be calculated more exactly. Most of the inputs in the sizing module serve as a guide for lofting a preliminary outer mold line for the UAV in OpenVSP, but do not directly influence the final performance calculations. They are superseded by values calculated in later stages of the workflow.

Since the classical equations for constraint sizing are given in terms of thrust loading, the relationship between propeller power and thrust in Equation 4.1 is utilized.

$$\frac{P}{W_{TO}} = \frac{T}{W_{TO}} * V \quad (4.1)$$

It should be noted that this relationship calculates propeller power, but use of motor and propeller efficiencies can yield the corresponding battery power required, which is useful for selecting an appropriate battery. This approach, with power loading defined as the battery power required divided by the take-off weight of the aircraft, is used in all following sections.

For each constraint, the relationship between wing loading ($\frac{W_{TO}}{S}$) and power loading ($\frac{P}{W_{TO}}$) must be found. The available constraints in CCADE are:

- Take-off
- Landing
- Maximum Speed
- Ceiling
- Rate-of-climb (ROC)
- Turns/Load Factor

The derivation of the relationship between wing loading and power loading for each of these constraints is now presented.

Take-off and Landing

Take-off and landing constraints are the required power to generate enough lift to take-off within the required distance, or fly at a slow enough speed to land on the required runway length. They are analyzed identically. The only difference between the two solutions comes from differences in the input values of the mission legs defined in the mission profile, most notably the required runway lengths. This section will refer to take-off for simplicity, but landing can be substituted for take-off at any point.

It is assumed that the velocity at take-off is at the minimum allowed velocity, 20% greater than the stall-speed of the aircraft. The take-off velocity is defined by Equation 4.2.

$$V_{TO} = 1.2V_0 = 1.2\sqrt{\frac{2}{\rho C_{L_{max}}} \frac{W_{TO}}{S}} \quad (4.2)$$

This depends on the density of the air at the altitude of the launch strip and the maximum lift coefficient of the UAV, which must be estimated from historical data of similar configurations. From Equation 4.2, it is apparent that the take-off velocity can be directly calculated for a given wing loading.

The complete form of the thrust loading required at take-off is given by Equation 5.77 in Brandt [9] as shown in Equation 4.3.

$$\frac{T}{W_{TO}} = \frac{1.44\beta^2}{\alpha\rho C_{L_{max}}gS_{TO}} \left(\frac{W_{TO}}{S} \right) + \frac{0.7C_{D_0}}{\beta C_{L_{max}}} + \mu \quad (4.3)$$

In the context of small-scale electric-powered UAVs, the thrust required to overcome parasitic drag and the friction of the runway is assumed to be negligible compared to the thrust required to generate the required lift. α and β are assumed to be unity. For the final form of power loading, only the first term of Equation 4.3 is included, assuming that the other two are negligible (or accounting for them in a safety factor). The efficiency of the motor, and the efficiency of the propeller are added to the equation by including them in the denominator. The definition of C_L is then substituted for $C_{L_{max}}$, and the take-off velocity multiplied on both sides to yield the battery power to weight ratio in Equation 4.4.

$$\frac{P}{W_{TO}} = \frac{1.44V_{TO}}{\rho gS_{TO}C_{L_{max}}\eta_m\eta_p} \left(\frac{W_{TO}}{S} \right) \quad (4.4)$$

Maximum Speed

The maximum speed constraint relates the power loading to wing loading based on the desired or anticipated maximum velocity of the aircraft. The thrust to weight ratio is found simply by relating the lift to drag ratio at this maximum speed as shown in Equation 4.5.

$$\frac{T}{W_{TO}} = \frac{1}{L/D} \quad (4.5)$$

The required lift is assumed to be the same as the take-off weight. The dynamic pressure, found with Equation 4.6, and the lift-induced drag coefficient, found by Equation 4.7, are then used to calculate the total drag on the aircraft by Equations 4.8 to 4.10,

$$q = \frac{1}{2}\rho V^2 \quad (4.6)$$

$$k = \frac{1}{\pi e AR} \quad (4.7)$$

$$D = D_{parasite} + D_{induced} \quad (4.8)$$

$$D_{parasite} = qSC_{D_0} \quad (4.9)$$

$$D_{induced} = \frac{kW_{TO}^2}{qS} \quad (4.10)$$

where V is the maximum velocity desired by the design team, taken as the maximum of all velocities in the mission profile. Substituting back into Equation 4.5 and rearranging with respect to the wing loading, the thrust loading in Equation 4.11 is obtained.

$$\frac{T}{W_{TO}} = \frac{qC_{D_0}}{\left(\frac{W_{TO}}{S}\right)} + \frac{k\left(\frac{W_{TO}}{S}\right)}{q} \quad (4.11)$$

By including the motor and propeller efficiency as before and multiplying by velocity to get the power, the power loading becomes:

$$\frac{P}{W_{TO}} = \frac{V\left(\frac{T}{W_{TO}}\right)}{\eta_m \eta_p} \quad (4.12)$$

If a separate maximum speed constraint must be considered, without including a planned event of reaching the said maximum speed, the duration of the leg can be set to zero.

Ceiling

As the UAV's altitude increases, the maximum thrust of the propeller decreases, to the point where the minimum thrust required for steady sea level flight is the maximum thrust from

the propeller available at that altitude. This is referred to as the ceiling. The ceiling requirement, according to [55], assumes that the lift coefficient at minimum power is:

$$C_{L_{mp}} = \sqrt{\frac{3C_{D_0}}{k}} \quad (4.13)$$

The ceiling velocity at this lift coefficient is the velocity for which battery power required is minimized. Using a similar method as the max speed constraint, and using a conservative adjustment for air density at the ceiling altitude, the power loading becomes:

$$V_{ceiling} = \sqrt{\frac{2 \left(\frac{W_{TO}}{S} \right)}{\rho C_{L_{mp}}}} \quad (4.14)$$

$$\frac{P}{W_{TO}} = \frac{V_{ceiling} \left[C_{D_0} + \frac{k}{q^2} \left(\frac{W_{TO}}{S} \right)^2 \right]}{\eta_m \eta_p \left(\frac{\rho}{\rho_{SL}} \right) C_{L_{mp}}} \quad (4.15)$$

The altitude used for the ceiling constraint is the maximum altitude of all mission legs. If a separate ceiling constraint must be included, without considering a planned event of reaching the ceiling, the duration of the leg can be set to zero.

Rate-of-climb

The rate of climb constraint follows the same approach as the ceiling constraint. The rate of climb (ROC) is included as an additional velocity to the one corresponding to the coefficient of lift at minimum power (Equation 4.13). Thus, the velocity at the ceiling is assumed to be the velocity at minimum power. In the ceiling constraint, the rate-of-climb was assumed to be zero, thus there is no opportunity to climb further. Taking into account the required ROC, the power loading becomes:

$$\frac{P}{W_{TO}} = \frac{1}{\eta_m \eta_p \frac{\rho}{\rho_{SL}}} \left(ROC + \frac{V_{ceiling} \left[C_{D_0} + \frac{k}{q^2} \left(\frac{W_{TO}}{S} \right)^2 \right]}{C_{L_{mp}}} \right) \quad (4.16)$$

Turns

The turns constraint is considered in the same way as the maximum speed constraint, except that the lift is multiplied by a required load factor (n) that must be achieved by the aircraft. Thus the thrust loading equation becomes:

$$\frac{T}{W_{TO}} = \frac{qC_{D_0}}{\frac{W_{TO}}{S}} + n^2 \frac{k \frac{W_{TO}}{S}}{q} \quad (4.17)$$

This can then be substituted back into Equation 4.12 to obtain the power loading.

4.2.2 Weight Sizing

The goal of weight sizing is to determine the total take-off weight of the aircraft. This is done by estimating the weight of the battery required to power the UAV for the duration of the mission, and then using an estimate of the empty weight fraction (without battery or payload) and the payload weight to solve for the take-off weight. The specific inputs to the weight sizing analysis include the wing loading and power loading chosen in constraint sizing, as well as the inputs described in Table 4.3.

Table 4.3: Description of the inputs to the weight analysis. The weight analysis also requires the same inputs as the constraint analysis, given in Table 4.2

Input Parameter	Units	Description
Battery Energy Density	$\frac{J}{N}$	The amount of energy contained per unit weight of the battery.
Empty Weight Fraction, $\frac{W_e}{W_{TO}}$	N/A	The fraction of the empty weight (without the battery or payload) to the total weight. This can be obtained from historical data on similar UAVs, or iterated on.
Payload Weight	N	The weight of your payload, which is generally fixed.

The equations for determining the battery weight fraction for Take-Off, Cruise, Loiter, Best Range Cruise, and Turns mission legs are here presented, followed by a listing of equations for the outputs from the weight sizing. The equations for battery weight fractions all follow the same form by taking the energy consumed over the duration of the mission leg and dividing by the battery energy density as shown in Equation 4.18. The battery weight fractions from all the legs are then summed together to obtain the total battery weight fraction, see Equation 4.19.

$$\left(\frac{W_b}{W_{TO}}\right)_{leg} = \frac{\left(\frac{P}{W_{TO}}\right)_{leg} t_{leg}}{k_{battery}} \quad (4.18)$$

$$\frac{W_b}{W_{TO}} = \sum \left(\frac{W_b}{W_{TO}}\right)_{leg} \quad (4.19)$$

Take-Off

Since the take-off leg is usually conducted at full throttle, Equation 4.18 applies directly using the required power loading calculated during the constraint sizing. To determine the time required to take-off, the time to achieve the 20% greater than stall speed is calculated:

$$t_{takeoff} = \frac{0.7V_{TO}^2}{g \frac{P}{W_{TO}} \eta_m \eta_p} \quad (4.20)$$

Cruise, Loiter, and Best Range Cruise

For other legs of the mission profile, including Cruise, Loiter, and Best Range Cruise, the UAV is typically not run at full throttle, thus the nominal power loading of the aircraft does not apply. Instead, the lift-to-drag ratio is estimated for the leg using the wing loading,

$$\left(\frac{L}{D}\right)_{leg} = \frac{\frac{W_{TO}}{S}}{qC_{D0} + \left(\frac{W_{TO}}{S}\right)^2 \frac{k}{q}} \quad (4.21)$$

Then, a distance x is found from the velocity and duration of the leg to determine the battery weight fraction for the leg,

$$\left(\frac{W_b}{W_{TO}}\right)_{leg} = \frac{x}{k_{battery} \left(\frac{L}{D}\right)_{leg} \eta_m \eta_p} \quad (4.22)$$

For the best-range cruise legs of the mission profile, a velocity is not defined, but must be calculated. This velocity is the speed at which the UAV has the longest range, as derived on page 656 of Raymer [1].

$$V_{BestRange} = \sqrt{\frac{2 \frac{W_{TO}}{S}}{\frac{\rho}{\rho_{SL}} \sqrt{\frac{C_{D0}}{k}}}} \quad (4.23)$$

From this velocity and the duration of the best-range cruise leg, the battery weight can be found using Equation 4.22.

Turns

The Turns leg assumes a steady turn at the minimum allowed velocity, equal to 1.2 times the stall speed as calculated in Equation 4.2. This takes into account a series of turns conducted for surveillance of an area, or circling around a position for a certain time. First the load factor at the turn velocity is calculated by Equation 4.24.

$$n = \sqrt{\frac{q}{k_{battery} \frac{W_{TO}}{S}} \left(\frac{P}{W_{TO}} \frac{\eta_m \eta_p}{V} - q C_{D0} \frac{S}{W_{TO}} \right)} \quad (4.24)$$

Then the battery weight can be calculated by Equation 4.25.

$$\left(\frac{W_b}{W_{TO}} \right)_{leg} = \frac{2\pi\theta \frac{P}{W_{TO}} (1.2V_0)}{k_{battery} g \sqrt{n^2 - 1}} \quad (4.25)$$

where θ is the number of turns, one turn equal to 360 degrees.

Outputs

Once the battery weights for each leg have been calculated, the outputs can be derived. A list of the outputs provided by the weight sizing and the equations used to calculate them is given in Table 4.4.

4.3 Geometry

The workflow continues to block three in Figure 4.1, where the geometry is more completely defined with dimensions in a complete OpenVSP model. CCADE does not attempt to provide CAD functionality within the user interface. Rather, the user is given quick access to launch

Table 4.4: Weight sizing outputs derived from the inputs from both sizing operations and the mission profile.

Output Parameter	Units	Description	Equation
Take-off Weight	N	Along with Wing Area, this is one of the most important outputs. This will be used to calculate required lift for later analyses.	$\frac{W_p}{1 - \frac{W_e}{W_{TO}} - \frac{W_{battery}}{W_{TO}}}$
Empty Weight	N	The weight minus battery and payload.	$W_{TO} \frac{W_e}{W_{TO}}$
Battery Weight	N	The battery weight required to complete the mission.	$W_{TO} \sum \frac{W_b}{W_{TO}}$
Wing Area	m^2	The planform area of the aircraft necessary to satisfy the constraints.	$W_{TO} \frac{S}{W_{TO}}$
Span	m	The span of the aircraft based off the assumed aspect ratio.	$\sqrt{S * AR}$
Required Power	W	The maximum power required to satisfy the constraints.	$W_{TO} \frac{P}{W_{TO}}$

OpenVSP from the program and upload the resulting file. CCADE provides a library which, upon uploading geometry, reads the parametric data stored and logs it in the database. This allows the complete definition of the geometry to be available in a format convenient for use in all subsequent analyses.

The OpenVSP file itself is formatted simply as a text XML document, and is small enough to be quickly transferred over an internet connection and stored in the database. The geometry for lifting surfaces are broken down into components, sections, and airfoils. An example on a single-section wing component in the OpenVSP XML language is shown in Figure 4.3. Lines with ellipses show portions of code that are not read by CCADE and are considered irrelevant to the full geometrical description of the vehicle. CCADE is able to determine which components are wing components, critical to later analyses, and which are not. At this point, only wing components are stored in the database. As CCADE reads in the file, the pattern seen in Figure 4.3 is mimicked in the database structure. Entities representing components, sections, and airfoils are linked to a single geometry entity, constituting the dimensions parameters of the design. Since it is tedious and time consuming to read through the entire geometry file each time data concerning the geometry is required, CCADE stores these values in tables within the database, and then accesses them via data model objects to simplify the process.

The Helix Toolkit library [56] was used to create a 3D viewport for manipulating the geometry without the need to manually download and open it in OpenVSP. A screenshot of this viewport with example geometry is shown in Figure 4.4. By including this viewport, CCADE eliminates

```

<Component>
  <Type>Mswing</Type>
  <General_Parms>
    <Name>Wing</Name>
    ...
    <Tran_X>0.152400</Tran_X>
    <Tran_Y>0.000000</Tran_Y>
    <Tran_Z>0.076200</Tran_Z>
    <Rot_X>0.000000</Rot_X>
    <Rot_Y>3.300000</Rot_Y>
    <Rot_Z>0.000000</Rot_Z>
    <RefArea>100.000000</RefArea>
    <RefSpan>10.000000</RefSpan>
    <RefCbar>1.000000</RefCbar>
    ...
  </General_Parms>
  <Mswing_Parms>
    <Total_Area>0.634063</Total_Area>
    <Total_Span>1.981200</Total_Span>
    <Total_Proj_Span>1.981200</Total_Proj_Span>
    <Avg_Chord>0.320040</Avg_Chord>
    <Sweep_Off>0.440700</Sweep_Off>
    ...
  </Mswing_Parms>
  ...
  <Section_List>
    <Section>
      ...
      <AR>3.095238</AR>
      <TR>0.909091</TR>
      <Area>0.317032</Area>
      <Span>0.990600</Span>
      <TC>0.304800</TC>
      <RC>0.335280</RC>
      <Sweep>0.000000</Sweep>
      <SweepLoc>0.000000</SweepLoc>
      <Twist>0.000000</Twist>
      <TwistLoc>0.000000</TwistLoc>
      <Dihedral>0.000000</Dihedral>...
    </Section>
  </Section_List>
</Component>

```

Figure 4.3: A single-section wing component example using the XML language of OpenVSP.

the need to shuffle geometry files or screenshot images via email so that teammates can view and interrogate the concept geometry. An OpenVSP script executed by CCADE exports the model as a stereo-lithography (STL) tessellated geometry file any time the geometry analysis module is opened. Once the OpenVSP file has been uploaded to the database, any team member can simply open the design in CCADE to rotate, zoom, and pan about the model as though it were in a CAD system. Currently, in order to edit the geometry, the user must download the OpenVSP file and edit it manually.



Figure 4.4: A screenshot of the geometry viewport in CCADE displaying the generated STL file for all team members.

At this point, there is enough information to make a rough estimate of parasite drag. Because of limited means of approximating parasite drag, and considering that the input geometry may vary greatly, a simple flat-plate model was chosen, making a number of conservative assumptions. The model assumes the UAV is a flat plate in turbulent flow at the largest velocity of the mission profile. The viscosity of the air μ at the lowest altitude is approximated using Sutherland's viscosity formula:

$$\mu = \mu_0 \frac{T_0 + C}{T + C} \left(\frac{T}{T_0} \right)^{\frac{3}{2}} \frac{kg}{m \cdot s} \quad (4.26)$$

$$T_0 = 273.15K \quad (4.27)$$

$$C = 110.4K \quad (4.28)$$

$$\mu_0 = 1.72 \times 10^{-5} \frac{kg}{m \cdot s} \quad (4.29)$$

where T is the air temperature at the given altitude according to a linear relationship between temperature and altitude. This relationship is:

$$T = -0.0065y_{alt} + 15.004^\circ C \quad (4.30)$$

taking $15.004^\circ C$ as an average temperature at sea-level. The air density at this altitude is approximated by

$$\rho = \frac{353.05 \frac{kg \cdot K}{m^3}}{T} \left(\frac{T}{T_0 + 15.004^\circ C} \right)^{5.256} \quad (4.31)$$

Once these parameters have been calculated, the Reynolds number may easily determined from Equation 4.32,

$$Re = \frac{\rho V c_{ref}}{\mu} \quad (4.32)$$

Then the parasite drag can be estimated by finding the skin friction coefficient from Equation 4.33 (from Equation 14.114 of [57]), and then using the estimate for drag coefficient in Equation 4.34.

$$C_{f_{turb}} = \frac{0.455}{\log(Re)^{2.58}} \quad 5 \times 10^5 < Re < 10^9 \quad (4.33)$$

$$C_{D_0} = C_{f_{turb}} \frac{S_{wet}}{S_{ref}} \quad (4.34)$$

At the click of a button, CCADE will take the flow parameters from the mission profile and geometry parameters from the OpenVSP model to calculate this C_{D_0} . The model admittedly does

not consider pressure drag, which would be considerable especially for high-velocity UAVs with a fuselage. In order to account for the assumptions of the model, the value of C_{D_0} is doubled before suggested back to the designer. It is up to the discretion of the design team to decide whether the value seems reasonable. It is not intended to replace more detailed analysis or historical data. However, when students are faced with the problem of estimating a parasite drag value with little or no relevant historical data and limited time, this method can provide a starting point for estimating the drag.

It should be noted that Equations 4.26- 4.32 are also used at various other points in CCADE, but are only mentioned here. More details are given in the CCADE Users Guide found in Appendix B.

4.4 Airfoil Selection

While airfoils may be included in the OpenVSP geometric definition, CCADE currently requires that designers assign airfoils using its own interface. The airfoil selection module is devoted to analyzing multiple airfoils in order to compare performance at the operating Reynolds number, and then allow the designers to assign any airfoil of their choice to each wing cross-section joint on the aircraft. Over 1500 airfoils from the University of Illinois at Urbana-Champaign (UIUC) database are available for the students to choose for their aircraft [58]. In the future, the option will be available to upload custom airfoils. As the user selects each airfoil in the list, a plot is updated in the CCADE GUI with an outline of the airfoil, so that the user can view the shape. This is especially useful when designers seek airfoils with reflexed trailing edges, as they are useful for designing flying-wing type configurations without active control systems. A screenshot of one of these plots for the Eppler 325 airfoil is shown in Figure 4.5. The green line represents the upper surface of the airfoil, the red represents the lower surface. The scale exaggerates the thickness of the airfoil so that variations in shape are highlighted.

XFOIL, a panel CFD code, is used for calculating the maximum lift coefficient and maximum lift-to-drag ratio of the airfoil, along with the angles of attack corresponding to each of these conditions. It uses a combination of an inviscid linear-vorticity panel method and a superimposed viscous layer using a two-equation lagged dissipation integral method [59]. The combination of viscous and potential flow capture the full aerodynamics around the airfoil.

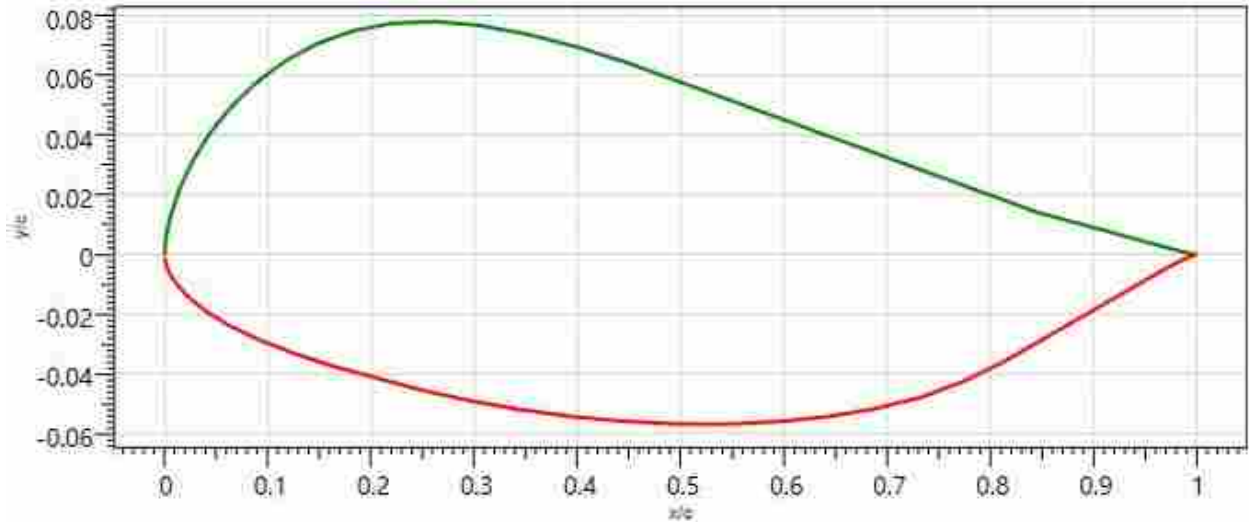


Figure 4.5: Plot of the Eppler 325 airfoil as displayed in the CCADE GUI. Plots of the airfoil help students identify airfoils such as this that have reflex.

An XFOIL script is parametrically constructed based on the procedure used on the popular website Airfoil Tools [60]. The script, executed by CCADE, divides the airfoil into linear panels according to the number of points in the airfoil data file, refines the resultant shape if necessary, and calculates the flow at a sweep of angles of attack specified by the user at the given Reynolds number, with a turbulence level, N_{crit} , of 9. According to the XFOIL documentation, this N_{crit} value corresponds to that found for most windtunnels, so it is appropriate for comparing results to windtunnel data [61]. The refinement occurs on Eppler and Selig airfoils where, according to the XFOIL documentation, the data files have coarse spacing around the leading edge. The additional inputs CCADE requires to build the script and run the analysis in XFOIL are listed in Table 4.5. If XFOIL is unable to converge on a solution, the user is advised to select angles of attack that are easier to analyze (between zero and five degrees, for example, before stall), or download the input file for manual adjustment of the parameters. Within a reasonable range of angles of attack, the script CCADE relies on should yield satisfactory results with most airfoils.

Once airfoil data has been generated, including lift, induced drag, and transition to turbulence, these results are graphed for the user, and key outputs printed to the screen. These key outputs help designers determine whether the airfoil is suited for the aircraft and the mission, and are described in greater detail in Table 4.6. The results are stored in the database, and can be accessed by any team member by selecting the appropriate airfoil in the list. Once the airfoil for each

Table 4.5: Description of additional inputs entered in the airfoil selection module of CCADE.

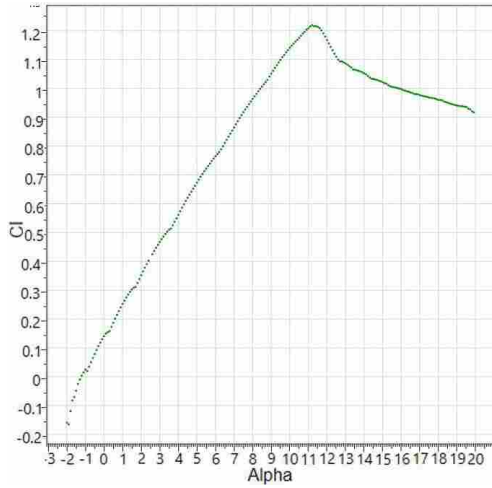
Input	Description
Minimum α	The starting angle of attack to be analyzed (assuming a sweep from low-to-high angles of attack).
Maximum α	The greatest angle of attack to be analyzed, where the analysis finishes.
α Increment	The step size, added to the previous α to obtain the new α to be analyzed.
Reynolds Number	The Reynolds number used to dictate flow conditions. This is either calculated directly from the chosen mission leg, or can be overridden by the user to analyze at an arbitrary Reynolds number.
Required C_L	The coefficient of lift required to fly at the velocity corresponding to the chosen mission leg.

section of the aircraft has been selected, it can be added in another dialog. This runs an OpenVSP script to add the airfoil file to the geometry and include it in the visualization. Then, when the VSP file is downloaded for re-parameterization in full-blown CAD, all of the airfoil dimensions are up to date and present in the model.

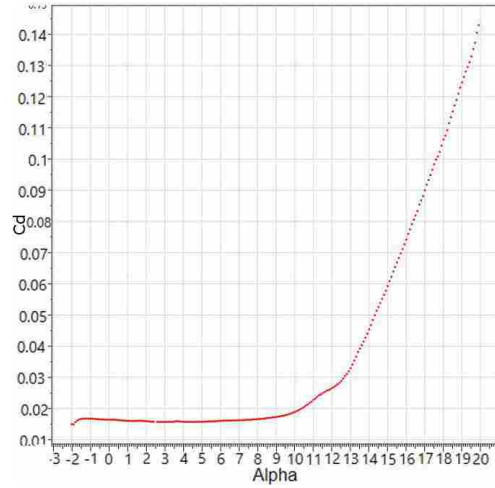
Table 4.6: Description of additional inputs entered in the airfoil selection module of CCADE.

Output	Description
Angle of attack at the required C_L	The angle of attack, if your UAV were an airfoil flying at the specified mission leg. You can think of it as the inclination of your airfoil which will affect the angle of attack of the UAV at the mission leg.
Angle of attack at the maximum L/D	The angle of attack, where the airfoil has its highest efficiency.
$C_{L_{max}}$	The coefficient of lift just before stall. This is the maximum lift you can get out of the airfoil.
Angle of attack at $C_{L_{max}}$	The angle of attack just before stall.
Stall velocity	Reducing velocity beyond this will cause the airfoil to stall. $V_{stall} = \sqrt{\frac{C_{L_{max}} q S}{2W_{TO}}}$.

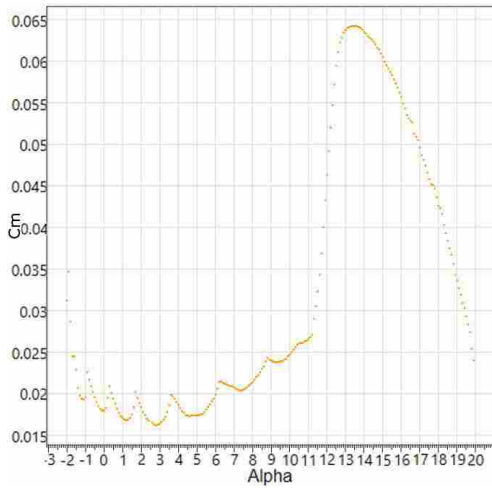
Examples of the XFOIL output plotted in CCADE are shown in Figure 4.6. The design team may examine Figure 4.6a to determine the maximum lift coefficient available from the airfoil and match that value with their sizing assumptions. They may also examine Figure 4.6c to ensure that the airfoil has a negative slope of moment coefficient for stability in the operating range, unless they plan to compensate with a tail. They may try to optimize the location of the peak of Figure 4.6e to their cruise angle of attack. All of these results are immediately available to all team members, despite the fact that only one student has actually run the calculations on his or her machine.



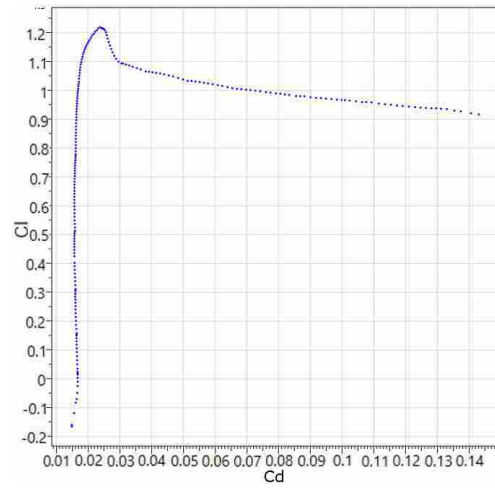
(a) Coefficient of lift versus angle of attack.



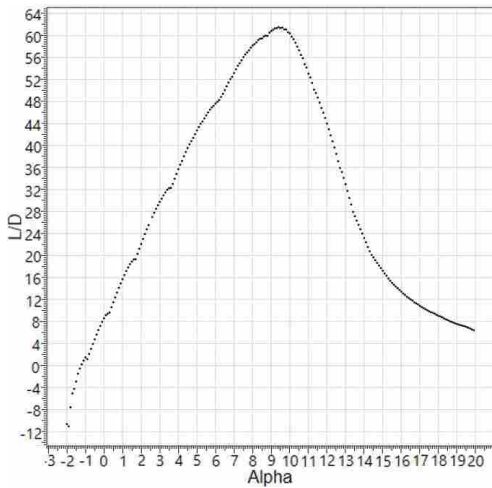
(b) Coefficient of drag versus angle of attack.



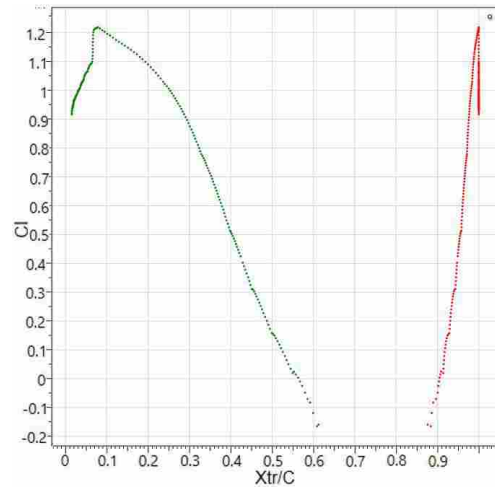
(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.



(e) Lift-to-drag ratio versus angle of attack.



(f) Transition to turbulence versus lift coefficient.

Figure 4.6: Examples of output produced by XFOIL and graphed by CCADE to display to all team members. Figure 4.6f shows the upper surface in green and the lower surface in red.

4.5 Aerodynamics

At this point, a full-wing aerodynamic analysis can be made. Using the airfoil data selected in the previous module, along with the CAD data from OpenVSP, CCADE can create a simplified three-dimensional panel geometry of the lifting surfaces to use in the vortex lattice program AVL. Fuselages per the AVL documentation are left out of the analysis because there is currently no standardized way to model them. CCADE creates 20 spanwise panels per section of the wing using a cosine distribution to provide a greater density of panels at section breaks and provide a smoother transition in panel size. Chordwise panels are also constant at 14. The resulting panel geometry is shown to the user after generating input files, and an example of such geometry is shown in Figure 4.7.

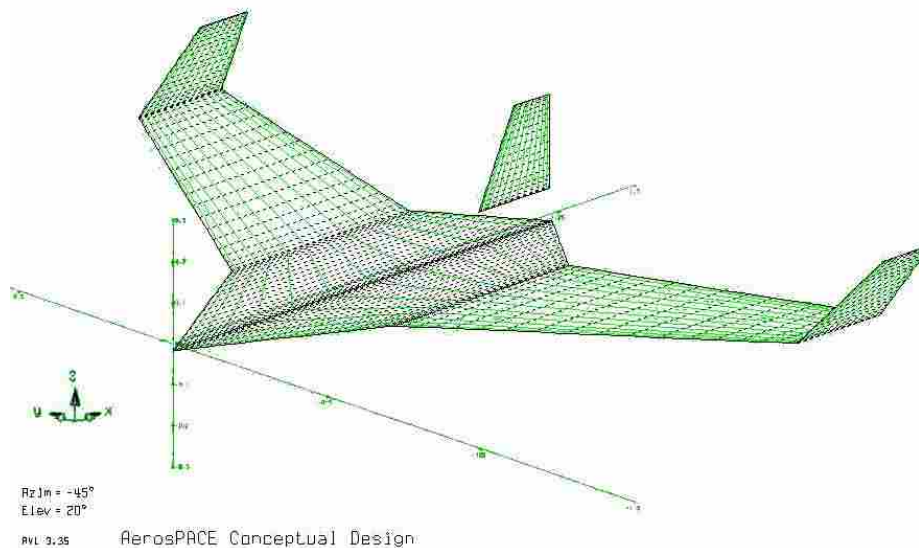


Figure 4.7: A screenshot of the panel geometry automatically generated in CCADE for use in calculating aerodynamics with AVL.

The set-up for this analysis is similar to that of the airfoil selection, with inputs identical to those found in Table 4.5. The Reynolds number for the analysis is dictated by the user's selection of a specific leg of the mission profile to analyze.

Key outputs are similar to those in the airfoil selection module, but reflect finite span wing values rather than airfoil values. This includes effects from the aspect ratio of the aircraft and three-dimensionality of the wing, among others. Examples of the output are shown in Figure 4.8.

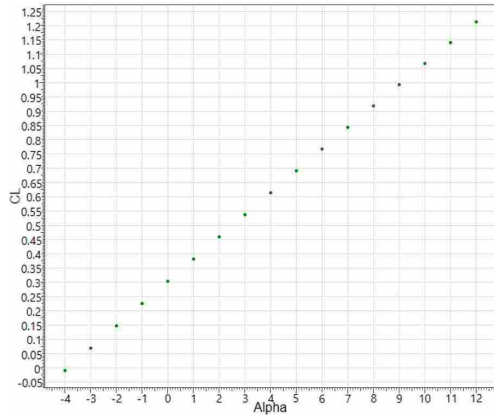
These outputs can be used to evaluate the aerodynamic performance of the UAV geometry, as well as the interaction of the airfoils selected in the previous analysis module. Values calculated by CCADE from the raw output include $\frac{L}{D_{max}}$, the angle of attack α at $\frac{L}{D_{max}}$, and the angle of attack α at the required C_L of the mission leg chosen for analysis.

Unlike XFOIL, AVL does not have coupling with any viscous code. Because of its inherent assumption of potential flow, parasite drag must be manually added to the results. CCADE automatically adds the parasite drag specified in the sizing module. The parasite drag estimate found in the geometry module can be applied to the sizing module to achieve the same effect. The parasite drag is added to the induced drag calculated by AVL, and can be seen as the minimum drag value of the drag polar in Figure 4.8d.

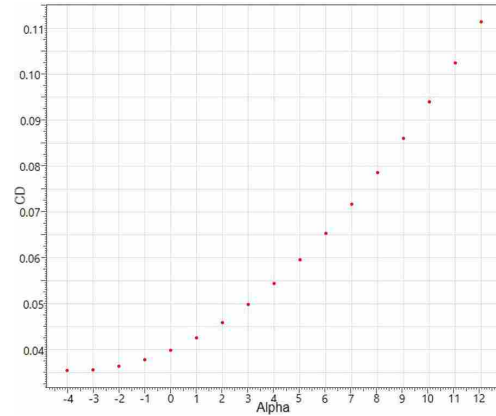
In addition to the results shown in Figure 4.8, AVL calculates a spanwise lift distribution which CCADE reads in and stores in the database. Although it is not presented at this point of the workflow, the lift distribution is required for the structures analysis, described in Section 4.8. The lift distribution is plotted in the structures module, and can be used by the designer to adjust the design as near to an elliptical lift distribution as possible for maximum span efficiency.

4.6 Stability and Control

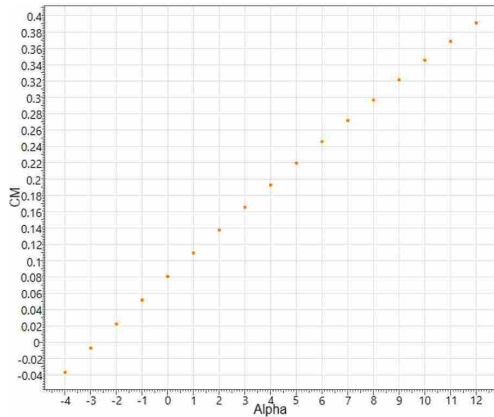
AVL is also used for the stability and control analysis. A model very similar visually to the one in Figure 4.7 is created, but for AVL to evaluate the control derivatives of the UAV, a model for the control surfaces of the aircraft must be added to the AVL input. A dialog in CCADE prompts the user to assign control surfaces to any of the sections defined in the geometry. A list of the available control surface types in CCADE, along with how they are modeled in AVL and where they might be located on the aircraft, is given in Table 4.7. The normal deflection is the direction the control surface can deflect: positive being deflected upwards and negative deflected downwards. The specification of symmetric or anti-symmetric refers to how the control surface acts in conjunction with its counterpart on a wing symmetric about the plane parallel to the centerline and normal to the span. Symmetric indicates that both control sides deflect either up or down together, and anti-symmetric indicates that when one deflects up, the other deflects down, and vice versa. The target moment is an internal instruction for CCADE to construct the AVL



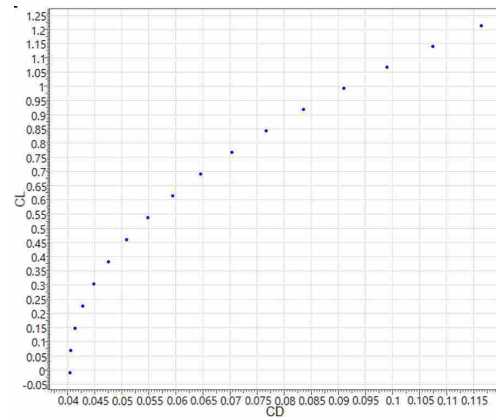
(a) Coefficient of lift versus angle of attack.



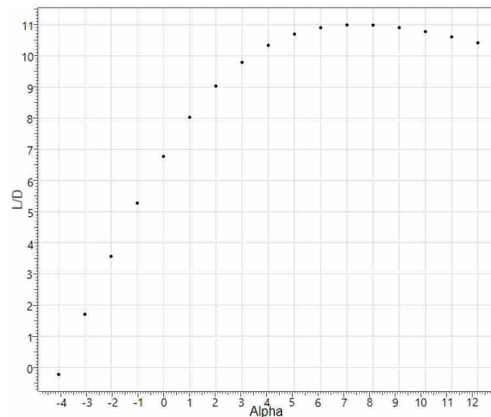
(b) Coefficient of drag versus angle of attack.



(c) Coefficient of pitching moment (about the CG) versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.



(e) Lift-to-drag ratio versus angle of attack.

Figure 4.8: Examples of output produced by AVL and graphed by CCADE to display to all team members.

script such that the code attempts to stabilize the aircraft about the roll, pitch, or yaw axis/axes by deflecting the indicated control surface.

Table 4.7: Types of control surfaces available in CCADE and the properties relevant to their modeling in AVL.

Control Surface Type	Normal Deflection	Symmetric/Anti-symmetric	Target Moment	Location
Elevator	Positive/Negative	Symmetric	Pitching	Horizontal Tail
Flap	Positive	Symmetric	None – increases lift	Main wing
Aileron	Positive/Negative	Antisymmetric	Rolling	Main wing
Rudder	Positive/Negative	No symmetry	Yaw	Vertical tail
Flaperon	Positive/Negative	Symmetric / Antisymmetric	Rolling – increases lift	Main wing
Elevon	Positive/Negative	Symmetric/Antisymmetric	Pitching/Rolling	Main wing – flying wings
Ruddervator	Positive/Negative	Symmetric/Antisymmetric	Pitching/Yaw	“V”-Tail

The user must also assign a percent chord to the control surface, which specifies how much of the chord of the wing section the control surface occupies. The standard analysis uses the required lift coefficient of the selected mission leg as a driver for the angle of attack of the UAV, and attempts to eliminate any residual pitching, yaw, or roll moments in the aircraft by deflecting the control surfaces to achieve steady, level flight.

The key outputs displayed by CCADE to the user are listed in Table 4.8. Since deflecting control surfaces normally reduces the aerodynamic efficiency of the wing, it is desirable to tune the design such that minimal control surface deflection is required. The required angle of attack should be at one that minimizes drag. Finally, the static margin should be enough to ensure adequate stability in presence of a disturbance (usually between 5-20%). The complete set of stability derivatives, neutral point, and other information from the raw AVL output are available for download by the students from the database, and can later be used in a dynamic stability analysis. This data is critical for helping students determine how to arrange components during weight and balance checks, re-evaluate the airfoils they select, and size their control surfaces.

Table 4.8: Output derived from raw AVL data in the stability and control module.

Output	Description
Required angle of attack	The angle of attack required to achieve the required lift coefficient, defined by the selected mission leg.
Control Surface Deflection	The required control surface deflection to achieve steady, level flight.
Neutral point	The position of the center of gravity where the aircraft is neutrally stable. To be positively stable (inherently return to steady, level flight after a perturbation), the center of gravity must be forward of this point.

4.7 Propulsion

The propulsion module uses textbook motor and propeller equations primarily from McCormick [24] which approximate the power efficiency and produce graphs used in matching the motor and propeller. These focus on determining an operating point for the motor and propeller which provides sufficient thrust to overcome the drag on the vehicle at the critical mission leg. For example, the goal may be to achieve the greatest efficiency at cruise condition. Another activity of the propulsion module is ensuring that at 100% thrust the vehicle has enough power to take-off. Additionally, by holding the operating revolutions per minute (RPM) constant and varying the airspeed, the designer gets a better picture of the performance of the propulsion system during acceleration.

Before conducting any analysis, the design team is required to define the motor and propeller properties, using values found from manufacturer specification sheets or testing of the propulsion system. These values are then input to Excel worksheets containing the equations described in this section using the Excel Interop library previously mentioned in Section 4.2. Graphs with the propulsion module output are updated as the inputs are changed, allowing the designers to iterate rapidly until the motor and propeller have been matched, and their combined performance assessed.

This section will first cover the inputs, outputs, and governing equations for determining motor performance. A similar procedure will then be followed for the performance of the propeller. The matching procedure is then explained and examples of the combined output presented. Finally, a brief explanation of the flight analysis involving acceleration at a specified operating point is given.

4.7.1 Motor Performance

The inputs required to determine motor performance are listed and defined in Table 4.9. These properties are normally listed on manufacturer specification sheets available online before a purchase is made. If a motor is already on-hand, the values can be very quickly determined from testing. The equations in this section are able to completely define the operating range of the motor and show where the motor has its highest efficiency.

Table 4.9: Description of inputs used to evaluate the performance of the motor.

Motor Input	Symbol	Units	Description
RPM Constant	K_v	$RPMV$	The RPMs obtained per volt applied via the speed controller to the motor.
No-load Current	I_0	A	The current the motor draws when there is no propeller or other load attached. Additional current is drawn linearly with respect to the load placed on the motor.
Motor Resistance	R_m	Ω	The electrical resistance of the motor, which increases the required current draw.
Gear Ratio	GR	N/A	The ratio of angular velocity input from the motor to angular velocity output to the driven load.
Weight	N/A	N/A	The weight of the motor. This value is used in the weight and balance module, not taken into account in this analysis.

The performance of the motor is coupled with the properties of the battery. The required properties of the battery are nominal battery voltage, and the electrical resistance of the battery. The input voltage to the motor is equal to the nominal voltage of the battery times the percent throttle, regulated by the electronic speed controller (ESC). The resistance of the battery, however, has the effect of linearly diminishing the voltage the motor sees as the current draw increases according to the relationship,

$$V_{in} = V_b - I_{in}R_{bc} \quad (4.35)$$

The RPM of the motor diminishes as the motor load increases, due to the additional torque the motor must impart to the load. The RPM of the motor is found by Equation 4.36.

$$RPM = K_v(V_{in} - I_{in}R_m) \quad (4.36)$$

In order to determine the torque the motor can impart to the load, first a thrust coefficient for the motor, K_t , is found using the relationship,

$$K_t = \frac{1352.4 \text{ oz} \cdot \text{in}}{K_v A} \quad (4.37)$$

The torque, power, and efficiency of the motor are then easily determined using Equations 4.38 through 4.40.

$$T_m = K_t(I_{in} - I_0) \quad (4.38)$$

$$P_m = (I_{in} - I_0)(V_{in} - I_{in}R_m) \quad (4.39)$$

$$\eta_m = \frac{P_m}{V_{in}I_{in}} \quad (4.40)$$

If the motor is connected to a gearbox, the inverse of the gear ratio acts as a multiplier for the RPM, thus Equation 4.36 becomes

$$P_{m_{gearbox}} = \frac{K_v(V_{in} - I_{in}R_m)}{GR} \quad (4.41)$$

4.7.2 Propeller Performance

The goal of the propeller analysis is to determine the thrust of the propeller, the power input required, and the efficiency at various RPMs. The inputs required to determine propeller performance are defined in Table 4.10. Diameter, pitch, and weight are apparent from manufacturer labeling and specifications, while the final three inputs are not usually advertised. These inputs are required to approximate the propeller performance, but require the ability to measure the propeller or test it in a windtunnel.

The method taken by CCADE to determine propeller thrust and power first approximates a thrust coefficient and power coefficient, C_T and C_P . These coefficients are defined as

$$C_T = \frac{T_{prop}}{\rho n^2 D^4} \quad (4.42)$$

Table 4.10: Description of inputs used to evaluate the performance of the propeller.

Propeller Input	Symbol	Units	Description
Diameter	D	in	The diameter of the area swept out by the propeller blades.
Pitch	p	in	The pitch or degree of twist in the propeller blade. Defined as the distance a propeller would move in one revolution if it were moving through a soft solid.
Number of Blades	B	N/A	The number of propeller blades, placed symmetrically about the hub.
Weight	N/A	lbs	The weight of the propeller. This value is used in the weight and balance module, not taken into account in this analysis.
Drag coefficient	$C_{D_{prop}}$	N/A	The drag coefficient of the propellers. Must be determined by testing or estimation based on similar propellers.
Chord-to-radius ratio at $\frac{3}{4}$ radius	$\frac{c}{R}$	N/A	The chord width of the propeller blade at the point three-quarters of the radius from the hub. This can be measured on a propeller, or estimated based on the type of flight the propeller is designed for.
Thrust coefficient C_T Coefficient	$C_{T_{coeff}}$	N/A	The slope of the thrust coefficient curve on performance charts of the propeller. This can only be empirically derived, or estimated based on similar propellers.

$$C_P = \frac{P_{prop}}{\rho n^3 D^5} \quad (4.43)$$

As previously mentioned, one assumption of the analysis is that C_T is linear with respect to the advance ratio of the propeller. The $C_{T_{coeff}}$ is the slope of this linear thrust curve, which runs to zero at the stall speed of the propeller estimated from experience [62] as

$$V_0 = n(p + 0.4D) \quad (4.44)$$

Since the advance ratio of the propeller is defined using velocity as

$$J = \frac{V}{nD} \quad (4.45)$$

it follows that the advance ratio at stall, J_0 is

$$J_0 = \frac{p}{D} + 0.4 \quad (4.46)$$

The thrust coefficient of the propeller from zero advance ratio to stall is calculated by

$$C_T = C_{T_{coefficient}}(J_0 - J) \quad (4.47)$$

The definition of thrust coefficient from Equation 4.42 is then used to determine the actual thrust produced by the propeller. An approximation from McCormick [24] is used to determine the power coefficient from the thrust coefficient. This method defines a cubic function $g(J)$ dependent only on the advance ratio of the propeller,

$$g(J) = -0.0167J^3 + 0.125J^2 - 0.0083J + 1 \quad (4.48)$$

Using this function, the power coefficient is found using Equation 4.49,

$$C_P = C_T J + \frac{1.12C_T}{2} \left[-J + \left(\sqrt{J^2 + \frac{8C_T}{\pi}} + \frac{\pi^4 \sigma g(J) C_{d_{prop}}}{32} \right) \right] \quad (4.49)$$

where

$$\sigma = \frac{\frac{c}{R} B}{\pi} \quad (4.50)$$

Once C_P has been found, the actual power required from the motor to drive the propeller can be found by using the definition of the power coefficient in Equation 4.43.

The propeller efficiency follows classical definitions of efficiency where $\eta = \frac{P_{out}}{P_{in}}$. In this case, the power in to the system is that provided by the motor, determined by C_P . The power out is the thrust produced by the propeller multiplied by the velocity the aircraft is moving at. This speed is found using the mission leg the designer chooses to make the analysis at. The resulting equation is given as Equation 4.51.

$$\eta_p = \frac{T_{prop} V}{P_{in}} \quad (4.51)$$

4.7.3 Motor-Propeller Matching

The matching procedure in the propulsion module helps the user to find the point at which the motor and propeller will operate at a given voltage, and what thrust can be expected at that operating point. The method used finds the required thrust to produce steady level flight at a cruise condition. The required power for this operating point is determined by the velocity of the aircraft via Equation 4.1. The following equations are used to find the operating point assuming that the

thrust from the propeller must only overcome the drag determined by Equation 4.8. From there, a matching exercise can be manually conducted to find the motor operating point by matching the power required by the propeller to the power output by the motor. The best motor-propeller combination will maximize efficiency at the operating point. The outputs from the propulsion module are listed in Table 4.11.

Table 4.11: Types of outputs available to be plotted from the propulsion module for assessing the power system and matching motor to propeller.

Output	Description
Motor Load	The current draw of the motor due to the load of the propeller.
Operating RPM	The RPM the motor drives the propeller at given the matching of motor to propeller.
Thrust & Drag v. RPM	Propeller thrust is plotted against UAV drag. The operating RPM is the RPM where the thrust overcomes the drag.
Power v. RPM	Propeller power and motor power are plotted together, with operating RPM shown. Since the power from the motor must equal the input to the propeller, the percent throttle must be adjusted until both curves intersect at the operating point.
Efficiency v. RPM	Motor and propeller efficiency are plotted together. The goal is to design so that the operating point is at an acceptable efficiency for both.

The matching advance ratio is found by finding the positive root of the propeller thrust curve minus the drag of the aircraft by Equation 4.52.

$$J_{match} = \frac{-\gamma + \sqrt{\gamma^2 + 4\gamma J_0}}{2} \quad (4.52)$$

where

$$\gamma = \frac{2C_{T_{coeff}}D^2}{C_{D_{prop}}S} \quad (4.53)$$

Once the matching advance ratio has been found, the definition of advance ratio can be used to find the required RPM, and then the corresponding motor load at that RPM,

$$I_{in} = \frac{\rho n^3 K_v C_p D^5}{RPM \cdot GR} + I_0 \quad (4.54)$$

where n and C_p are evaluated at the matched advance ratio from Equation 4.52.

The required operating power is then found by

$$P_{in} = \frac{(I_{in} - I_0)RPM \cdot GR}{K_v} \quad (4.55)$$

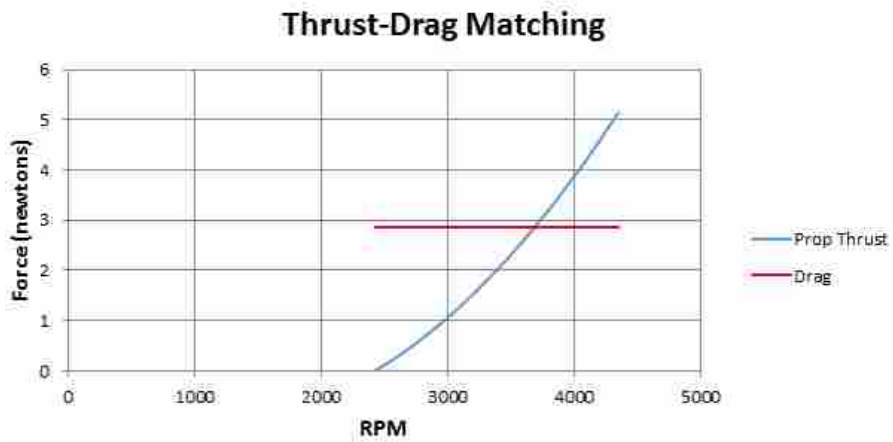
At this point, a power-matching scheme must be utilized by the user to adjust the motor performance curve by manipulating the battery input voltage. Figure 4.9 shows the three output graphs described in Table 4.11. The manual power matching scheme requires the designer to carefully watch how Figure 4.9b changes as the percent throttle is adjusted. Once the motor and propeller performance curves intersect at the operating point power, the throttle required to operate the system has been found.

4.7.4 Flight Analysis

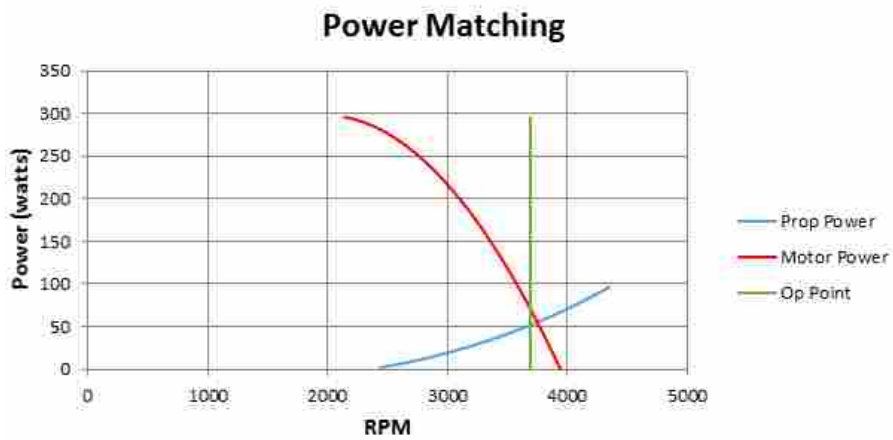
The operating point found during the matching is here analyzed and plotted against the UAV airspeed. This is done by utilizing the same propeller and motor performance equations contained in Sections 4.7.1 and 4.7.2, but holding RPM constant at the operating RPM and varying the airspeed. Examples of the output are shown in Figure 4.10. By examining how the thrust output and motor output power vary with airspeed, the designer can see the effects of acceleration on the propulsion system. At a given throttle, the thrust will vary as the aircraft picks up speed until it converges at the operating point. Likewise, as the pilot eases up on the throttle, there will be portions of the curve where the UAV response is more sensitive, represented by the slope of the curves in Figure 4.10a. Furthermore, the efficiency of the system can be examined such as in Figure 4.10b to see how the propeller efficiency changes as the aircraft accelerates.

4.8 Structures

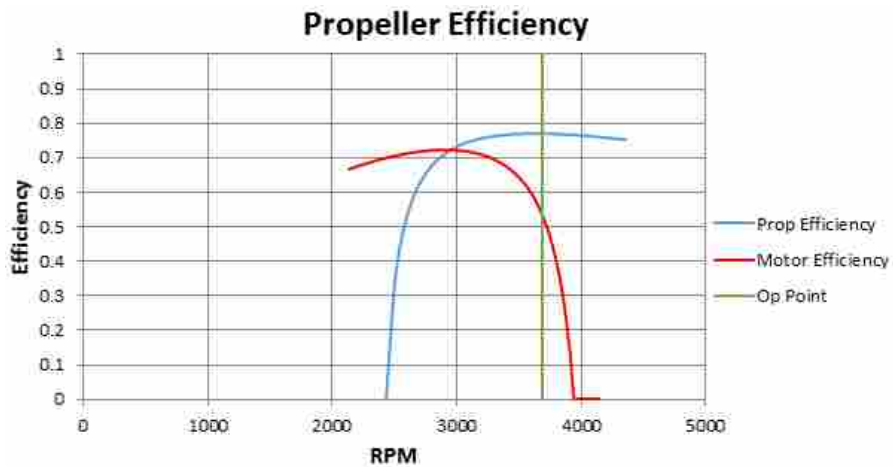
Because structural computations can very quickly become computationally expensive with the complex geometries involved in aircraft design, only a simple beam model was chosen to serve as a baseline for conceptual design. The model approximates a single spar in the wing as a cantilever beam fastened at the root of the wing body. The model assumes only aerodynamic loading of the spar due to lift, to which a load factor may be applied to simulate maneuvers.



(a) Thrust and drag versus RPM.

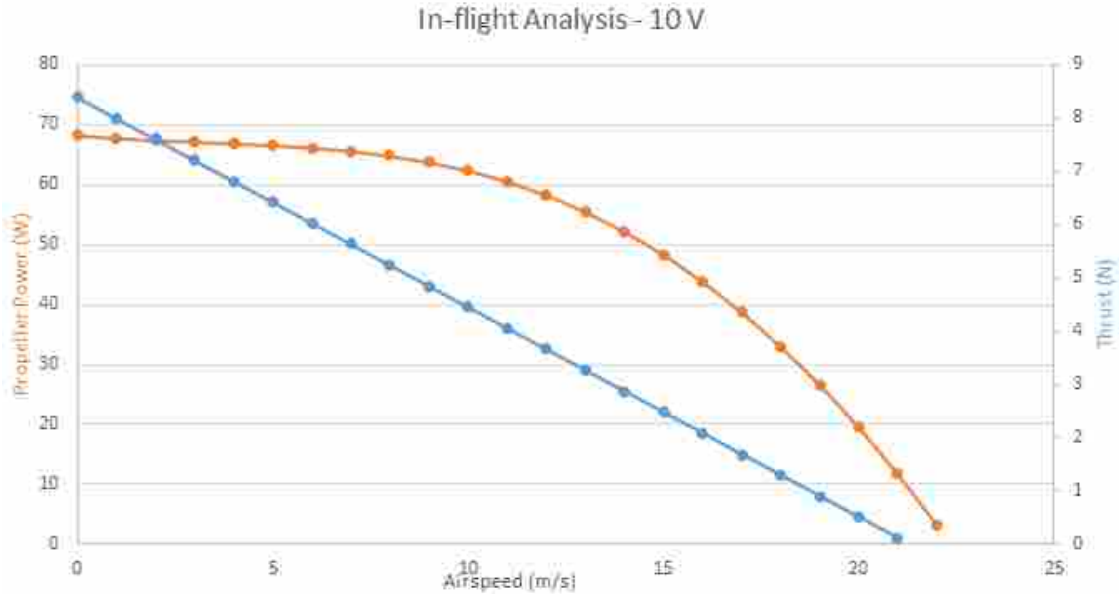


(b) Motor and propeller power versus RPM.

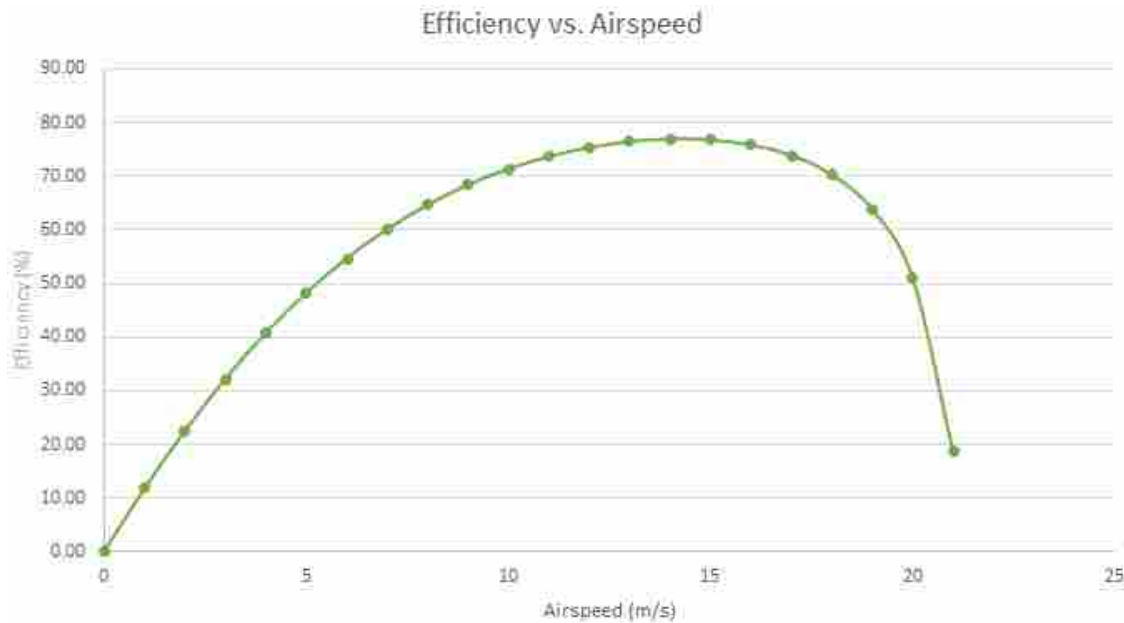


(c) Motor and propeller efficiency versus RPM.

Figure 4.9: Examples of output produced by the propulsion module and graphed by CCADE to display to all team members. Operating line determined by the matching equations graphed in green.



(a) Propeller thrust and power versus flight speed.



(b) Propeller efficiency versus flight speed.

Figure 4.10: Examples of output produced by the propulsion module to analyze the propeller at the operating point during acceleration and deceleration.

The inputs to the structures module include a definition of the spar by specifying cross-sections. The cross-sections are linearly swept together to model the complete spar geometry. The user is presented a dialog to define the properties of each cross-section along the spar for each

lifting surface of interest. The design team may define as many cross-sections as necessary, and position them along the span of the wing. It is currently the responsibility of the designer to ensure that the spar fits inside the thickness of the wing at any given point. CCADE assists the user in placing cross-sections at wing section junctions (places where the wing profile changes), since they are a logical position for changing the cross-section of the spar via a linkage, but the user may also choose to place the cross-section any place in-between these junctions. A summary of the properties that must be defined for each cross-section of the spar is given in Table 4.12.

Table 4.12: Description of the structural module inputs to describe each cross-section for a single spar in a wing component.

Cross-section input	Symbol	Units	Description
Cross-sectional Area	A	m^2	The two-dimensional area of the cross-section. CCADE does not specify fixed cross-sectional shapes to use. A shape may be chosen to base the area off of, or the dimensions of a shape derived after the spar has been sized.
Moment of Inertia	I	m^4	The resistance to bending about the neutral axis of the cross-section. Just as cross-sectional area, the moment of inertia may be defined independent of any shape. This puts the responsibility in the hands of the designer to choose values that correspond to real dimensions.
Distance to Neutral Axis	y_{na}	m	The neutral axis is the axis along which there is no bending moment in the beam. This is the furthest vertical point of the cross-section from that neutral axis.
Placement	x	m	This is the span-wise distance from the root of the cross-section. CCADE assists in placing cross-sections in locations where there is a change in the wing section.

The lift distribution data is taken from the results of the aerodynamics module, described in Section 4.5. Only the basic shape of the lift distribution, L , is required, as the actual lift can be scaled to the required lift coefficient of the mission leg at which the analysis is to be performed. The shear loading, V_s , and bending moment, M , distribution of the wing is simply calculated via an integration of the lift and shear distributions, respectively, by Equations 4.56 and 4.57.

$$V_s = \int_0^{\frac{b}{2}} L ds \quad (4.56)$$

$$M = \int_0^{\frac{b}{2}} V_s ds \quad (4.57)$$

For boundary conditions, it follows from a free-body diagram of the spar (modeled as a cantilever beam) that at the wingtip, the shear is equal to the portion of lift concentrated at the tip, and the moment is equal to zero.

Since these loads must be queried at locations that may not correspond to the panel geometry created by CCADE for the AVL analysis, the lift distribution must be parameterized. Currently, this is done using a 6th-order polynomial fit. However, since this yields inaccurate results at locations where numerical error from AVL is evident, future versions of CCADE plan to implement a second-order Taylor Series polynomial fit. This ensures that the forces are reliably integrated numerically, and not overly skewed by numerical anomalies in the AVL results. This is accomplished by calculating the first and second derivatives of the lift coefficient with respect to spanwise location. The first derivative is numerically calculated using a forward-differencing approach,

$$\left. \frac{dL}{ds} \right|_x = \frac{L_{x+1} - L_x}{\Delta x} \quad (4.58)$$

The second derivative is calculated with a central-differencing approach,

$$\left. \frac{d^2L}{ds^2} \right|_x = \frac{\left. \frac{dL}{ds} \right|_{x+1} - \left. \frac{dL}{ds} \right|_{x-1}}{2\Delta x} \quad (4.59)$$

The Taylor series polynomial is then defined as

$$L(s) = L_0 + \left. \frac{dL}{ds} \right|_x \Delta x + \frac{1}{2} \left. \frac{d^2L}{ds^2} \right|_x \Delta x^2 \quad (4.60)$$

where L_0 is the lift at the calculated AVL panel closest to the evaluation point with respect to s .

AVL provides an effective local lift coefficient with respect to chord at each spanwise row of panels, used as the input for the lift distribution calculations above. This lift coefficient is proportional to the actual lift generated by that row. A comparison of the generated AVL lift compared to the values obtained using the Taylor Series approximation are shown in Figure 4.11. This method clearly provides excellent matching of the AVL data, with the added benefit of reducing the influence of out-lying data points.

By applying the lift distribution to the spar geometry previously defined, bending moment and shear stress distributions can then be determined. From the linearly interpolated cross-sections of the spar, a distribution of cross-sectional area and moment of inertia is obtained. The cross-

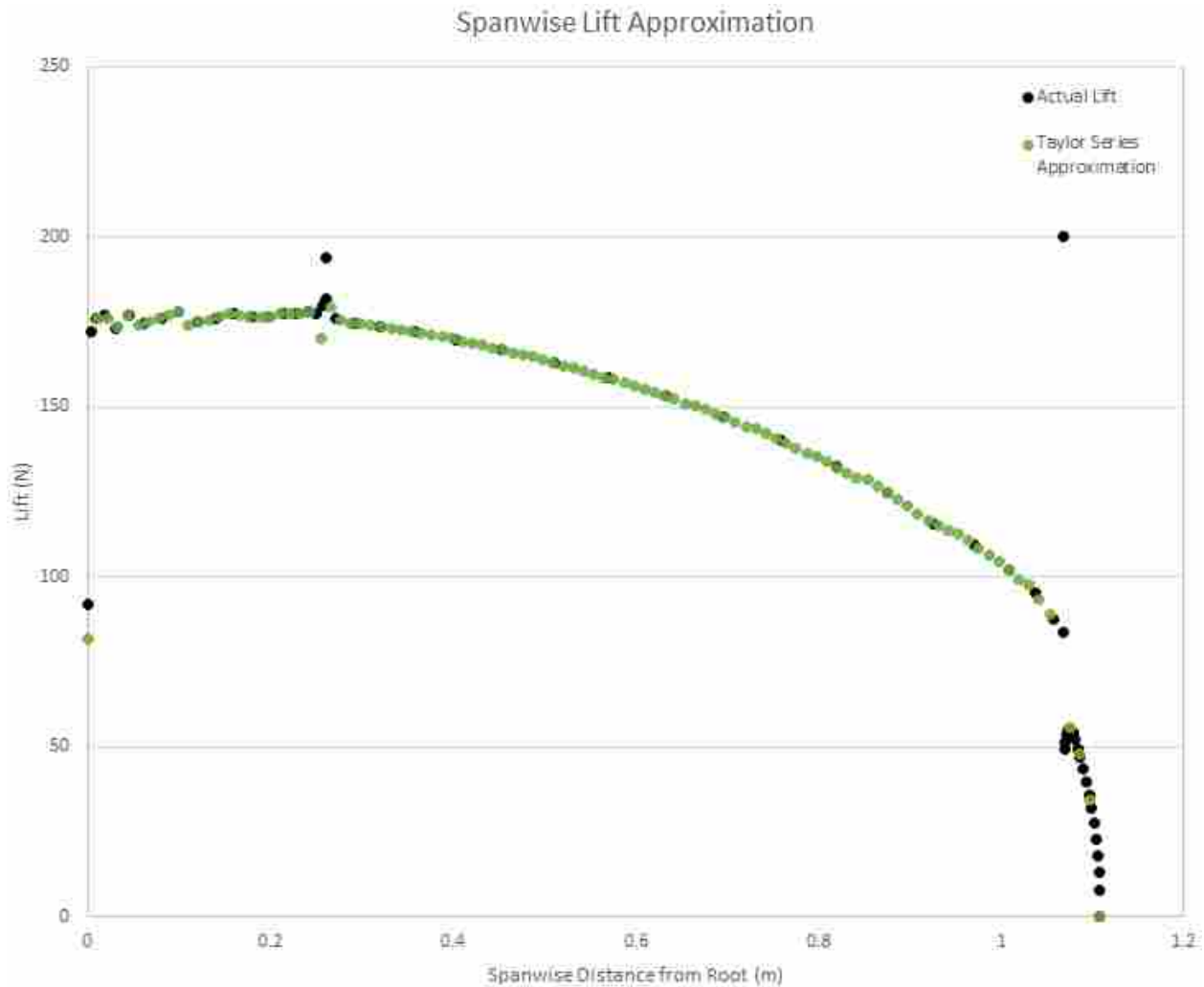


Figure 4.11: Plot of lift versus spanwise position from the actual AVL output (black) and the 2nd-order Taylor Series Approximation (green). The approximation effectively interpolates the output data by following the trend of the data.

sectional area is exactly determined using a linear relationship. The moment of inertia is assumed to vary according to a quadratic fit between sections. With these distributions, the stresses may be easily calculated for discretized sections of the beam by using the definition of shear stress and bending stress in a beam by Equations 4.61 and 4.62.

$$\tau = \frac{V_s}{A} \quad (4.61)$$

$$\sigma = \frac{My_{na}}{I} \quad (4.62)$$

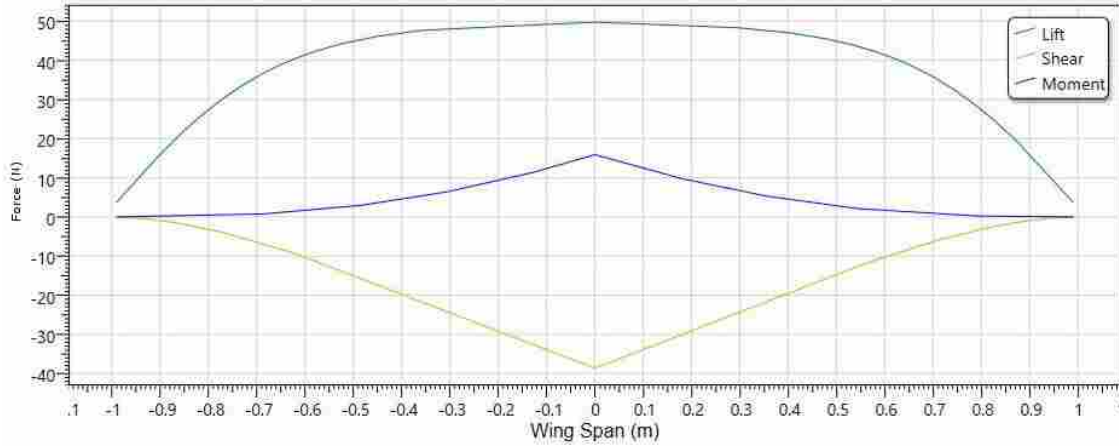
The outputs for the structures module are summarized in Table 4.13. Each of these outputs is graphed in the CCADE GUI and available as long as the spar cross-sections have been defined. The deflection of the spar is also available when a material for the spar has been selected which has an associated elastic modulus. This deflection is calculated by combining the deflection at each discretized section of the beam from the tip to the root by Equation 4.63.

$$\delta_x = \frac{V_s \Delta x^3}{3EI} \quad (4.63)$$

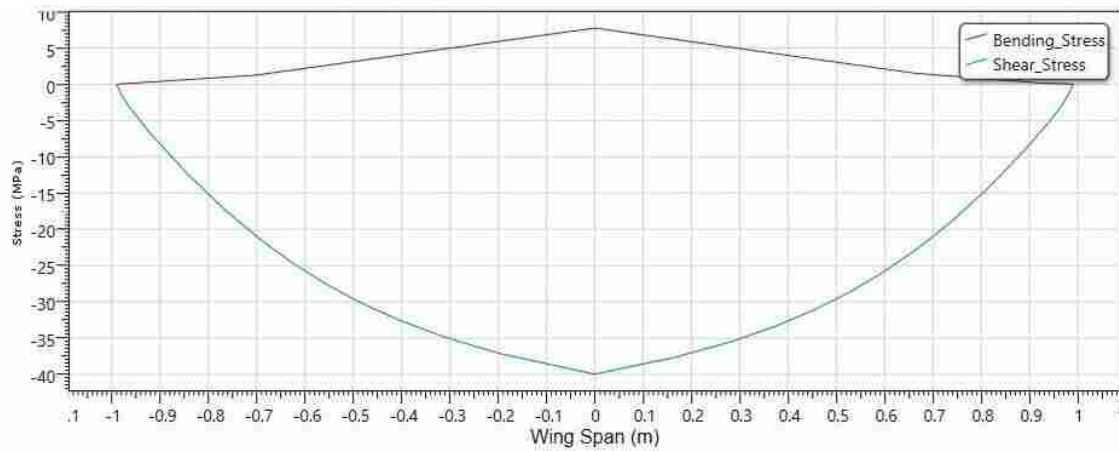
Examples of the types of plots generated by these calculations are shown in Figure 4.12. These plots can be used by teams to size the spars in their UAV and thus estimate the weight associated with the structural components.

Table 4.13: Definitions of the loads and spar output obtained from the structures module.

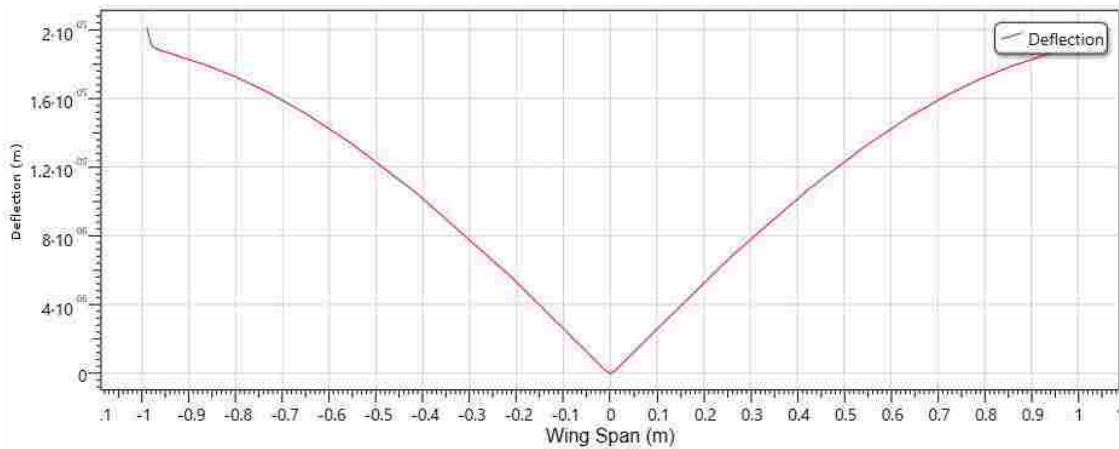
Spar Output	Description
Lift	The span-wise lift distribution calculated from the aerodynamic analysis, multiplied by the load factor from. This is the load on the wing.
Shear	This is the shear force in the spar, derived by integrating the lift distribution with respect to span-wise distance.
Moment	This is the bending moment in the spar, derived by integrating the shear distribution with respect to the span-wise distance.
Bending Stress	This is the bending stress of the spar, by using the bending moment in the equation for bending stress: where y is the distance from the neutral axis to the furthest extent of the cross-section, and I is the moment of inertia.
Shear Stress	This is the shear stress in the spar using the equation for normal stress: where V is the shear force, and A is the cross-sectional area.
Deflection	This is the deflection of the spar assuming a linearly elastic, isotropic material operating according to beam theory. Maximum deflection is δ , where δ is the deflection, P the load on the wing, L the length of the beam to that point, E is the modulus of the material, and I the moment of inertia of the cross-section.



(a) Lift, shear and moment versus spanwise position.



(b) Shear and bending stress versus spanwise position.



(c) Wing deflection versus spanwise position.

Figure 4.12: Examples of output produced by the structures module and graphed by CCADE used for sizing the main wing spar, assuming that the spar is the only load-carrying member under the applied loads.

CHAPTER 5. DESIGN EXPLORATION

This chapter discusses the design exploration and optimization section of the workflow described in Chapter 4. The work of this chapter fits into block 6 of Figure 4.1. First, the problem statement for the design exploration module is given, including the optimization problem and obstacles of implementation in CCADE. The methodology of the CCADE solution for the design exploration module is explained, most of which is left for future work. Finally, a proof-of-concept of the optimization workflow is presented, which illustrates the possible uses and effectiveness of the module.

5.1 Problem Statement

The purpose of the design exploration module is to enable students the ability to conduct trade studies and explore the local design space of the configuration they have analyzed in CCADE. Students in AerosPACE have expressed great interest in the ability to conduct trade studies during the conceptual design to hone in on a better design. The design exploration module may include aerodynamic, weights, propulsion, stability and control, and structural analyses together in a multi-disciplinary analysis, or any subset of them for a less complicated trade study. Although optimizing the system including every analysis discipline is useful, often localized optimization using a single discipline or a small number of competing disciplines can yield even greater learning and understanding. When interpreting results of a system-level optimization, the risk might be greater that students perceive the optimization as a “black box” which produces an optimal design on-demand. With a reduced scope, as illustrated by the results of the proof-of-concept in Section 5.3.3, the final geometry is unlikely to be realistic, since there are trade-offs between other disciplines of which the optimization is unaware. This helps students recognize and put greater emphasis on relationships between design parameters rather than blindly accepting the resultant configuration from the optimization routine. One of the main goals of this work is to parameterize the creation

of optimization workflows of varying complexities, such that the design studies may be tailored to the needs of the team. The level of detail and number of analysis disciplines may be chosen by the team, along with the design variables, constraints, and design objectives.

After a first-iteration of the workflow described in Chapter 4, sufficient information is available to run a multi-disciplinary optimization or trade study by modifying the values associated with the design configuration. To simplify the process, the configuration itself is held static, meaning that components and wing sections are neither added nor deleted. The key parameters for these components, such as aspect ratio, span, average chord, etc. may be easily manipulated, however, to explore a wide range of alternatives.

The other area of focus is related to the collaborative nature of CCADE. Optimus is used as a process automation and optimization platform for the design exploration. As a large commercial package requiring licensing and separate installation, it is unreasonable to attempt to deploy a copy of the software and a license along with CCADE, as has been done with the other analysis codes. This discourages the idea of deploying a copy of the software along with CCADE. A method must be devised to provide access to Optimus's capabilities without distributing individual copies of the software itself. A process for interpreting the results of the trade studies and displaying them to the user must also be developed.

5.2 Methodology

The proposed solution is two-fold: code which resides on client machines to prepare input for processing by Optimus, and code located on the server to handle the design exploration requests, execute them, and then enter post-processing information to the database. This is a significant deviation from the way CCADE functions earlier in the workflow. Whereas CCADE otherwise operates as a “thick-client,” meaning that the heavy computational load is carried by the client machines, this solution proposes a “thin-client” solution for the design exploration module. This involves shifting the responsibility of running the optimization routine from the client to a server. A schematic of the proposed architecture for this module is given in Figure 5.1. The flow of this schematic begins with the client machine sending a request to the Optimus Web Service for a job to be placed in the queue. This request includes a configuration file that Optimus is able to interpret via its Python application programming interface (API) to build the requested workflow

and include the specified design variables, constraints, and objectives. The queue operates on a first-in, first-out basis, with jobs waiting in line for their chance to run on a single server license of Optimus. Since the server used to host the service and database for CCADE does not have the appropriate hardware to run Optimus, it is suggested that a separate server machine be acquired specifically for this purpose. The server can easily send the job request at the front of the queue to this machine when it is available to run. Once the job has completed, this machine passes the results back to the server for it to store in the database and send notification back to the client. The client then queries the database via the CCADE Data Service described in Chapter 3 to obtain the results and display them to the user.

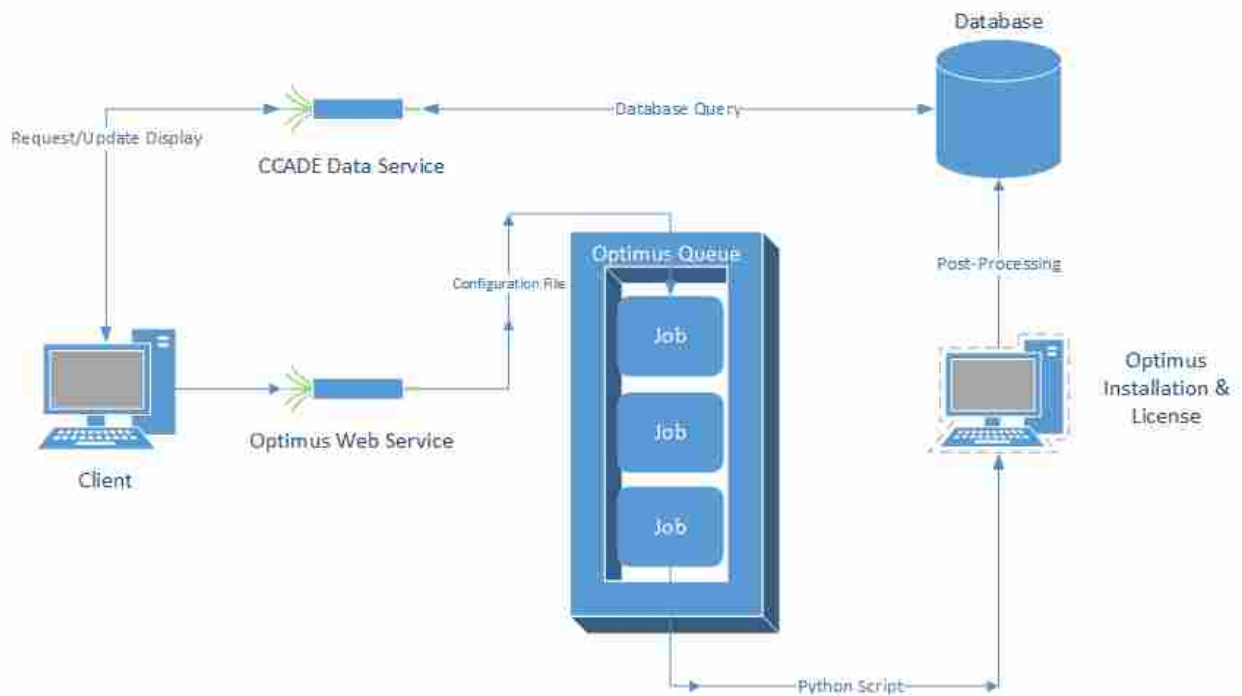


Figure 5.1: The proposed architecture for the design exploration module. The thin-client sends a job request to the server, where it waits in a queue until it reaches the front of the line. After the job is run by the server, the results are stored in the database and the client can then query the database to display them.

The role of the CCADE GUI is to present the available options for students to build the workflow. Students should be able to select which analysis disciplines they intend to include in the trade study, which design parameters will be allowed to vary, and which will remain static.

This information is sufficient for executing a design of experiments (DOE), which will give the students a clear picture of the design space they have specified, and show them what direction to take their design to improve performance. To run a computer-driven optimization scheme, the GUI should allow students to decide what constraints on the analysis output must be satisfied while attempting to optimize their design objectives. A screenshot of the current design of the GUI for the design exploration module is shown in Figure 5.2. A user can select parameters from the tree

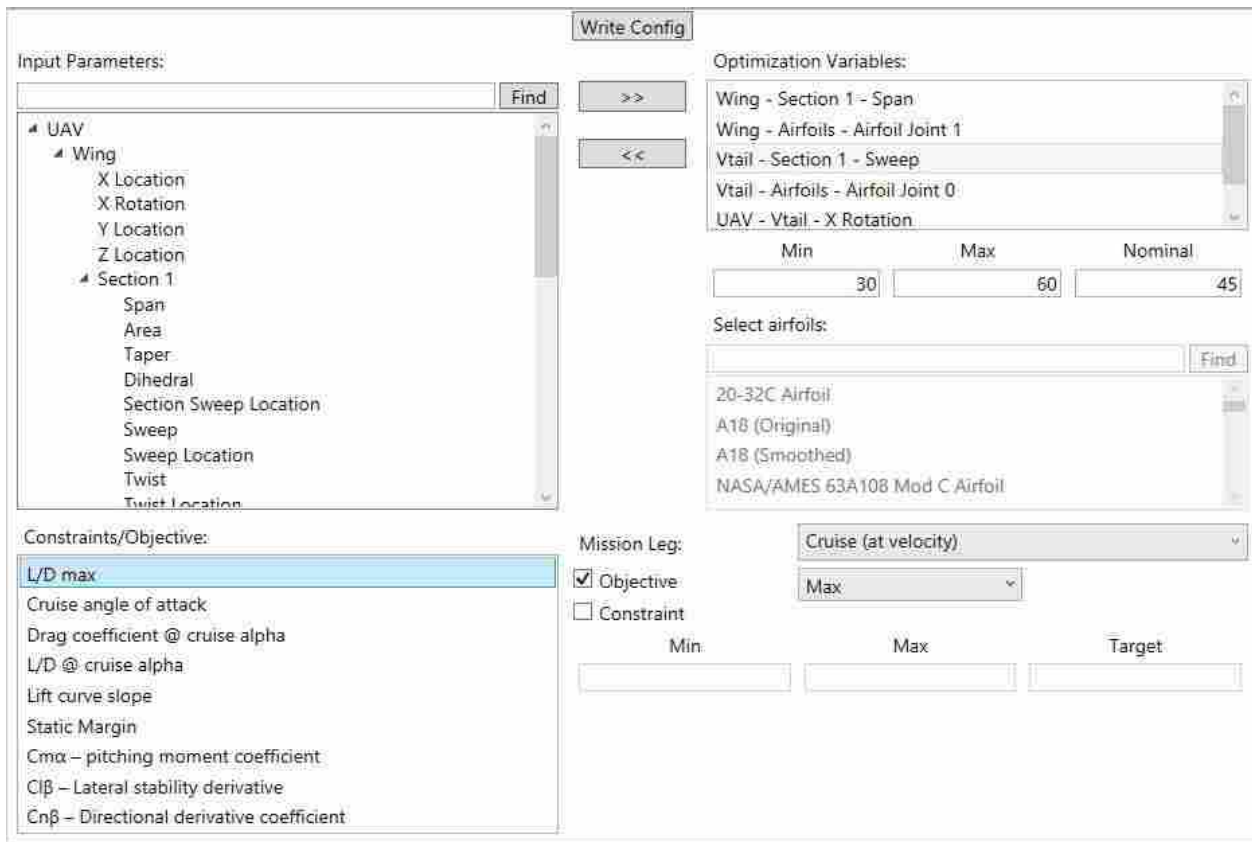


Figure 5.2: A screenshot of the design exploration module GUI interface. The module currently only handles setup for a trade-study including aerodynamic and stability analyses.

on the left organized by geometry, and then click the arrow to define it as a design variable. By selecting the design variable, the user may give upper and lower bounds, as well as a nominal value if it is a numerically continuous parameter. In the case of an airfoil variable, which must be made up of a list of discrete airfoils to consider, users can select as many airfoils as they wish to consider from the box including all airfoils from the UIUC Airfoil Database. A list of key outputs

is in the lower-left, and once defined as either design constraint or objective, can have minimum, maximum, and target values assigned to them. In the case of a design objective, the user can choose to minimize or maximize it. All of these decisions must then be translated into a format which is standardized for reading by the Optimus API, and can be serialized to transmit over the web. This is accomplished by writing a simple text file with headers distinguishing variable parameters, constraints, and objectives. An example of this configuration file is given in Appendix C.1. For it to provide sufficient information for Optimus to build the workflow, it must contain:

- The analyses to be conducted. These may be related to a single discipline, or they may be multi-disciplinary, potentially encompassing the entire design workflow.
- Flight parameters, from the mission leg selected for analysis
- The design variables, along with their nominal values and bounds, to be used in the optimization. These may be either continuous variables or discrete.
- Current control surface and airfoil data, since these are defined internally by CCADE and not necessarily present in the OpenVSP geometry
- The design constraints, with minimum, maximum, or target values
- The design objectives, and whether to minimize or maximize ,Flight mission data and parameterized base geometry

By parametrically defining the workflow, designers are provided the flexibility to determine which type of study is most useful and prudent at any time.

The Optimus Web Service acts as a job manager, accepting request messages from the multiple client machines and funneling them into the Optimus Queue. The job definition contains a copy of the configuration file, identifying information of the user who made the request, and which UAV design the request belongs to. This information is important for the Web Service to return notification back to CCADE that the job has been queued, when execution of the job has begun, and when the job has finished. It also is critical when storing the results in the database, in order to correctly attach the results of the job back to the appropriate UAV design. While running the job, Optimus periodically updates a log file showing a summary of the results of each iteration

of the DOE or optimization. These updates could also be sent back to the client to show them how their analysis is progressing even before the job has completed. That way, any problems in the setup could be identified before valuable time and computational resources were spent running an erroneous job.

Once a job is ready to be executed, the configuration file must first be interpreted into a workflow Optimus can execute. This method was developed in conjunction with researchers from Optimus who coded the example Python scripts shown in Appendix C.2 for use in the proof-of-concept described in Section 5.3. This script modifies a template Optimus workflow which would contain the definition of every analysis discipline and place-holders for design variables and outputs. An example of the template workflow used with the proof-of-concept is shown in Figure 5.3. This example only includes geometry modification using OpenVSP, and aerodynamic, stability and control analysis using AVL. It represents the general format of the workflow, with blue and red ovals representing inputs and outputs, respectively, green cylinders representing text files, and orange squares with a lightning bolt representing batch scripts for executing the analyses. The workflow follows the flow and direction of the arrows, allowing parallel as well as serial processes. The configuration file will cause design variables to be added in the appropriate group of inputs, and for sections of the workflow to be suppressed in case the full range of analysis disciplines is not considered. The outputs are hard-coded into the workflow, so that should additional outputs be needed, a modification of the template itself is required.

A single server operating Optimus as a service is sufficient given the relatively small number of students using the software. It is unlikely that scaling will become an issue unless the AerosPACE course expands considerably from its current size. By implementing this functionality as a web service, the design exploration code compliments the collaborative nature of CCADE.

Once the Optimus routine has completed, the results must be passed back into the database model to make the information accessible to the entire team. Common post-processing options may be made available to the user from the CCADE GUI, but any user may also download the raw output data from the database to perform custom post-processing.

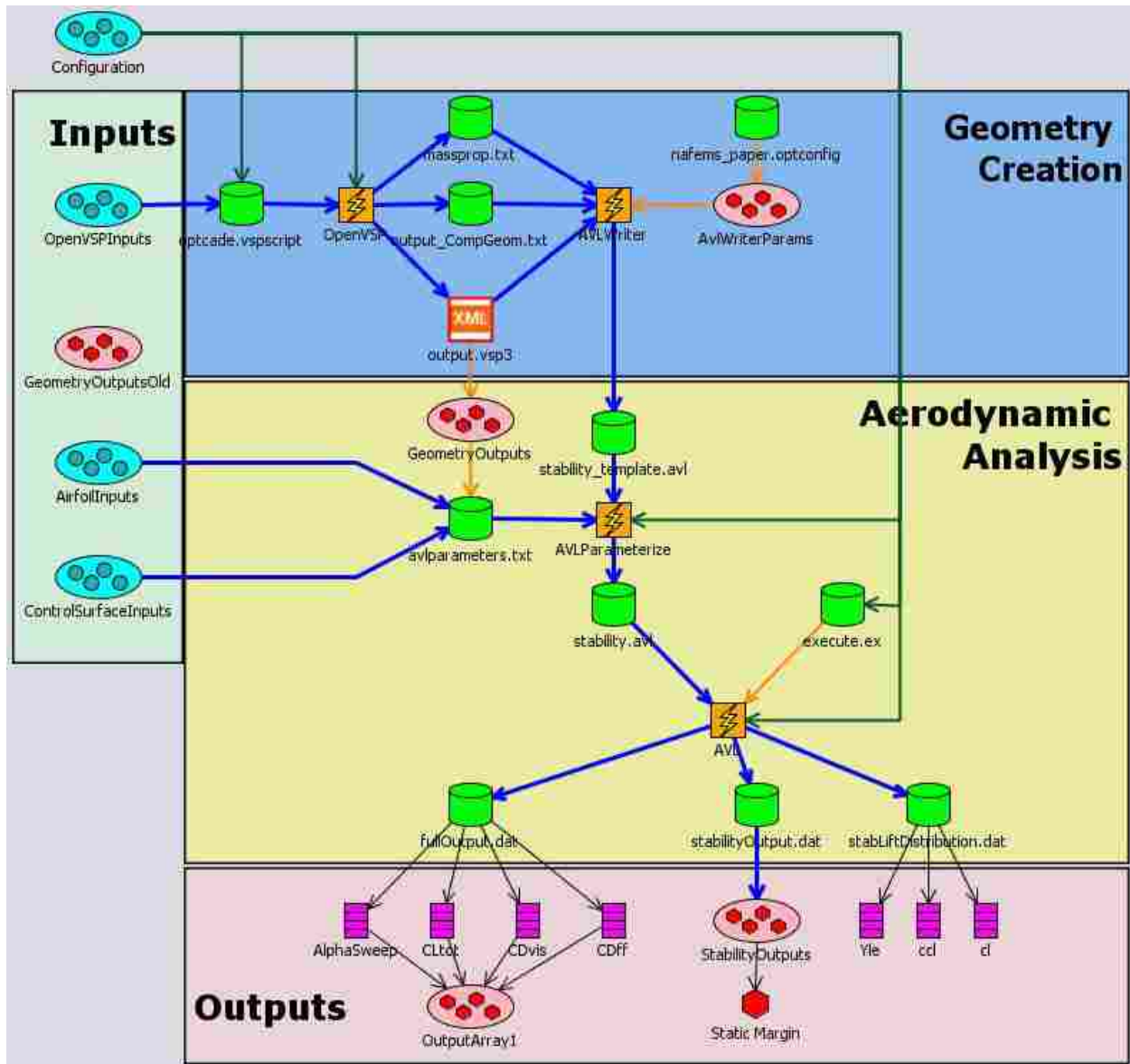


Figure 5.3: A visualization of the template Optimus workflow. Inputs are shown as blue ovals, outputs as red ovals, hexagons, or stack of rectangles, files are shown as green cylinders or block containing “XML”, and batch scripts are shown as orange squares with a lightning bolt.

5.3 Proof of Concept

A proof-of-concept was conducted to show the capabilities of using Optimus as a design exploration framework, and the feasibility of parametrically constructing a workflow that can be executed automatically for varying geometry. The selected problem involved the trade-off between aerodynamics and stability/control. This section defines the available inputs and outputs for selec-

tion from CCADE developed for the proof-of-concept, and lists the variables selected for the test case run. Then the setup of the workflow for the problem is described. The results are presented with a discussion of their value as it pertains to AerosPACE, including how it can increase the learning and productivity of students.

5.3.1 Design Variables, Constraints, Objectives

The CCADE GUI developed for this proof-of-concept, shown in Figure 5.2, exposed a variety of geometric parameters for inclusion in the design exploration. For easier navigation, the parameters are grouped by the aircraft component and wing section they belong to; no fuselage variables were considered for the proof-of-concept, but could be included in future versions. The available input parameters are shown in their hierarchical relationships in Figure 5.4. The top-level containers for parameters are components, which in the current version of CCADE are the wing bodies belonging to the aircraft. The wing body has several component-level parameters, shown in green in Figure 5.4. Of the wing sections, the number of which is variable depending on the design, each section has a set of parameters which define it, shown in blue in Figure 5.4. Eight of these parameters, shown in light-blue, are considered driving parameters. Of the driving parameters, only three may be independently manipulated. The selection of which three parameters to manipulate is up to the designer when the geometry is modeled.

Selection of section-level design variables is limited to the driving section parameters used in the initial construction of the geometry in OpenVSP. Non-driving parameters such as sweep and twist can be added even if they were not present in the base geometry. Each wing section may have a control surface associated with it, the percent chord of which may be chosen as a design variable. Finally, each wing component has a set of airfoil cross-sections at section joints. These are located at the root of the aircraft, and occur between wing sections out to the tip, where the final “joint” is defined. Airfoil joints are considered discrete variables, with the user selecting from a finite list of airfoil data files, which will be used during the optimization. All other variables and constraints are continuous. A pre-determined list of potential constraints or objectives are available for the user to decide which are important to their particular case. This list of constraints includes:

- Maximum lift-to-drag ratio

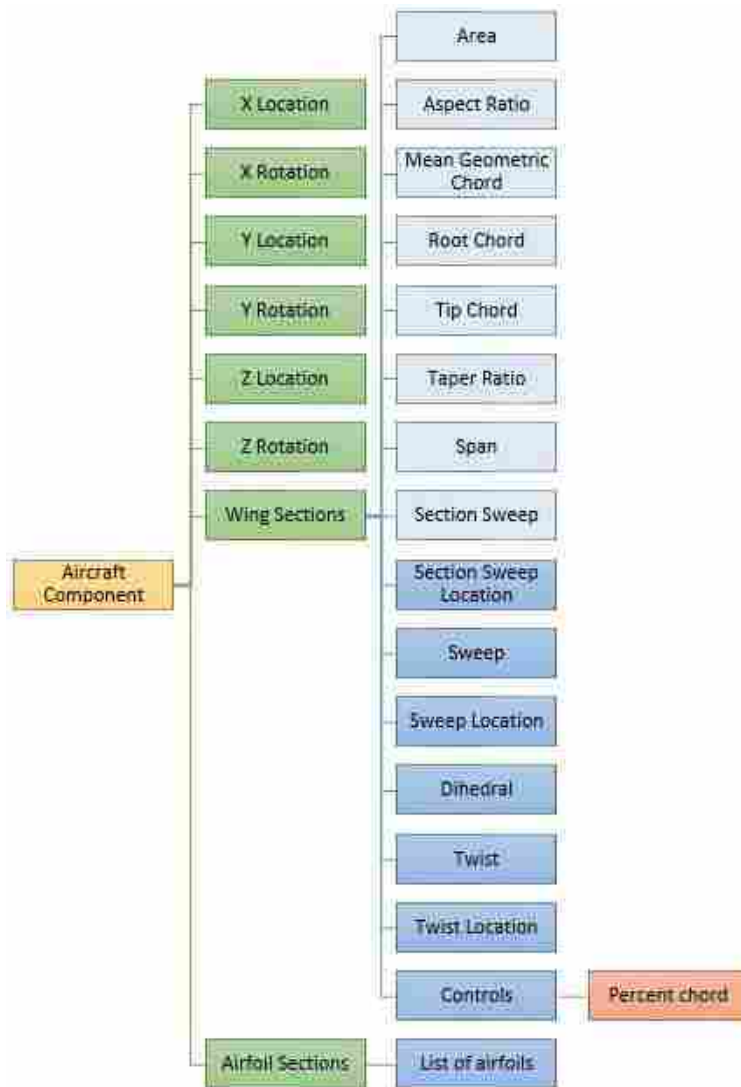


Figure 5.4: A diagram of how the available geometry input parameters in CCADE are organized. Only three of the driving parameters, shown in light-blue, are shown, depending on the original geometry definition.

- Angle of attack at specified flight condition
- Drag coefficient at flight angle of attack
- Lift-to-drag ratio at flight angle of attack
- Lift curve slope
- Control surface deflections

- Static Margin
- $C_{L\beta}$ – Lateral stability derivative coefficient
- $C_{n\beta}$ – Directional derivative coefficient
- $C_{m\alpha}$ – Pitching moment coefficient

These outputs can be assigned either as constraints or objectives in the optimization. If a DOE is conducted, constraints and objectives are not necessary.

5.3.2 Setup

Although an attempt to optimize the entire UAV system including all disciplines currently addressed by CCADE is under development, this proof-of-concept only considers an aerodynamic and stability analysis of the aircraft, given the geometry and definition of control surfaces. This is considered sufficient to show the merits of the proposed architecture and demonstrate the benefits of using Optimus for the design exploration. AVL is used for determining the aerodynamic performance and stability of the varying geometry, with the necessary inclusion of OpenVSP to provide the interface with the geometry for modification. Despite it including only two analysis disciplines, this work is a valuable addition to AerosPACE, since a reduced scope can help students more easily grasp the relationships between a smaller number of parameters. Future work will build on this proof-of-concept and provide even greater functionality.

OpenVSP is used to keep track of dependencies in the geometry and calculate key geometric and mass property outputs such as mean aerodynamic chord, planform area, wetted area, and center of gravity of the outer mold line of the aircraft (assuming uniform distribution). The OpenVSP definition of the geometry is converted into a flat panel geometry read in by AVL using previously written libraries from CCADE. A script is then written from the flight mission data and optimization configuration to instruct AVL to calculate the desired outputs, including the aerodynamic coefficients and lift distribution, as well as the stability of the aircraft. Since AVL assumes inviscid flow, the parasite drag of the vehicle must be provided. The value for parasite drag input in the CCADE sizing module is used as a fixed value for the analysis.

For the proof-of-concept case, Table 5.1 defines the parameters used as design variables, constraints, and objectives for the optimization. For reference, shapes of the airfoils used for the optimization can be found in Appendix D. For the stability analysis, a lift coefficient of 0.3 was to be attained. A DOE was conducted using the generated workflow using the Latin Hypercube sampling method with 100 points. An optimization routine was also run using the differential evolutionary algorithm available in Optimus, since the airfoil variables can only be evaluated as discrete airfoils. The genetic algorithm had a population size of 20 and was run for 30 generations, giving a total of 600 cases analyzed. The crossover rate used to encourage diversity in the population was 0.85.

Table 5.1: Specifications of the design variables, constraints, and objectives in the proof-of-concept DOE and optimization.

	Parameter	Lower Bound	Upper Bound	Nominal
Design Variables	Main Wing, Section 1 Sweep	10	50	30
	Main Wing, Section 2 Aspect Ratio	3	7	4
	Main Wing, Section 3 Dihedral	50	90	75
	Elevon Percent Chord	10	60	30
	Main Wing, Root (Joint 0) Airfoil	From airfoils: CJ 25209, Eppler 186, Eppler 325, Eppler 330, Eppler 340, EH 1.5/9.0, EH 2.0/10.0, MH-61, MH-78, Phoenix, RS-001 T10, RS-004 A, Selig 5010, Selig 5020, SD 7003		
	Main Wing, Joint 1 Airfoil			
	Main Wing, Joint 2 Airfoil			
Constraints	Static Margin	0.1	0.2	
	Cl.beta		0	
	Cm.alpha		0	
	Cn.beta	0		
Target-based objectives—		Target		
Objectives	Static Margin	0.15		
	Elevator Deflection	0		
	Aileron Deflection	0		
	Rudder Deflection	0		

5.3.3 Results and Discussion

Which graphs and results to offer to the user in CCADE is a subject of future work. In the author's opinion, some of the more useful post-processing options available from Optimus include the following:

- Summary - a listing of the inputs and outputs of all runs
- Correlation Values - a matrix with Pearson and Spearman correlations between all inputs and outputs
- Scatter Plot - shows relationship between any two parameters
- Scatter 3D Plot - shows relationship between any three parameters
- Histogram Plot - a plot showing the distribution of experiments run with the corresponding input value or resulting in ranges of an output variable value
- Correlation Scatter Plot - a matrix showing the Pearson and Spearman correlations, histograms, and scatter plots of all input and output combinations
- Optimum Summary - a description of the starting and ending points of the optimization routine
- Optimization Progress Plot - a series of plots showing the change in input and output parameters as the optimization routine progresses

Some other post-processing options that require more advanced interpretation, but might merit more investigation include:

- Self-Organizing Map (SOM) Plot - an artificial neural network representing correlation between parameters, used to determine interesting regions of the design space
- Bubble Plot - shows relationship between any four parameters (by using the coordinates of the bubble, its size and color)
- Cluster Scatter Plot - a variation of the scatter plot which shows the aggregate regions occupied by similar experiments

This section attempts to give a sampling of the results obtained from the proof-of-concept DOE and optimization. The presented results should illustrate the power of Optimus in exploring the design space, and the value it represents to students still in the conceptual design phase.

Figure 5.5 shows an example of a scatter plot generated by Optimus from the results of the DOE. The example compares the pitching moment coefficient derivative with elevon percent chord. There is a clear linear relationship between the two, despite the variation in other parameters in the experiments, which accounts for scatter in the trend. Optimus allows the user to change the parameters plotted on each axes to any which were chosen at the optimization setup. This helps students recognize how certain inputs effect the output, in what direction they effect them, and how strongly.

Figure 5.6 is an example of a correlation values table, which has a similar goal of showing students the sensitivity of each parameter to others, but the information is condensed into a single table. Each parameter is examined against all the others, resulting in a square matrix of correlation values from statistical methods developed by Pearson and Spearman. The ratings range from zero to one, with zero meaning the parameters have no influence on each other, and one meaning that a change in one yields an identical change in the other. The diagonal of the chart is the influence of each parameter on itself, which is of necessity unity, and by definition, the matrix is symmetric about the diagonal. The advantage of such a chart is that not only can relationships between inputs and outputs be examined, but the trade-offs between outputs are also apparent. For example, the matrix indicates that any change in pitching moment coefficient derivative has a profound effect on the static margin of the aircraft. This can be easily confirmed when it is realized that static margin of the aircraft is defined as $\frac{C_{m\alpha}}{C_{L\alpha}}$.

The results from the differential evolutionary optimization are shown in Table 5.2. The tables shows the nominal starting values and the final optimum values based on the objective function, "GOAL," which is a measure of the goodness of a design based on how well its outputs reflect the design objectives. The value of GOAL is not significant, but the change in value is. The change shown in Table 5.2 shows a 99% decrease in the GOAL, which is a significant improvement in the objective function. The elevon percent chord increased from 30% in the original design to almost 40%. The aspect ratio in the center wing section likewise increased, from 4 to about 6.5. Interestingly, the airfoil selection changed completely from the original design point, leaving

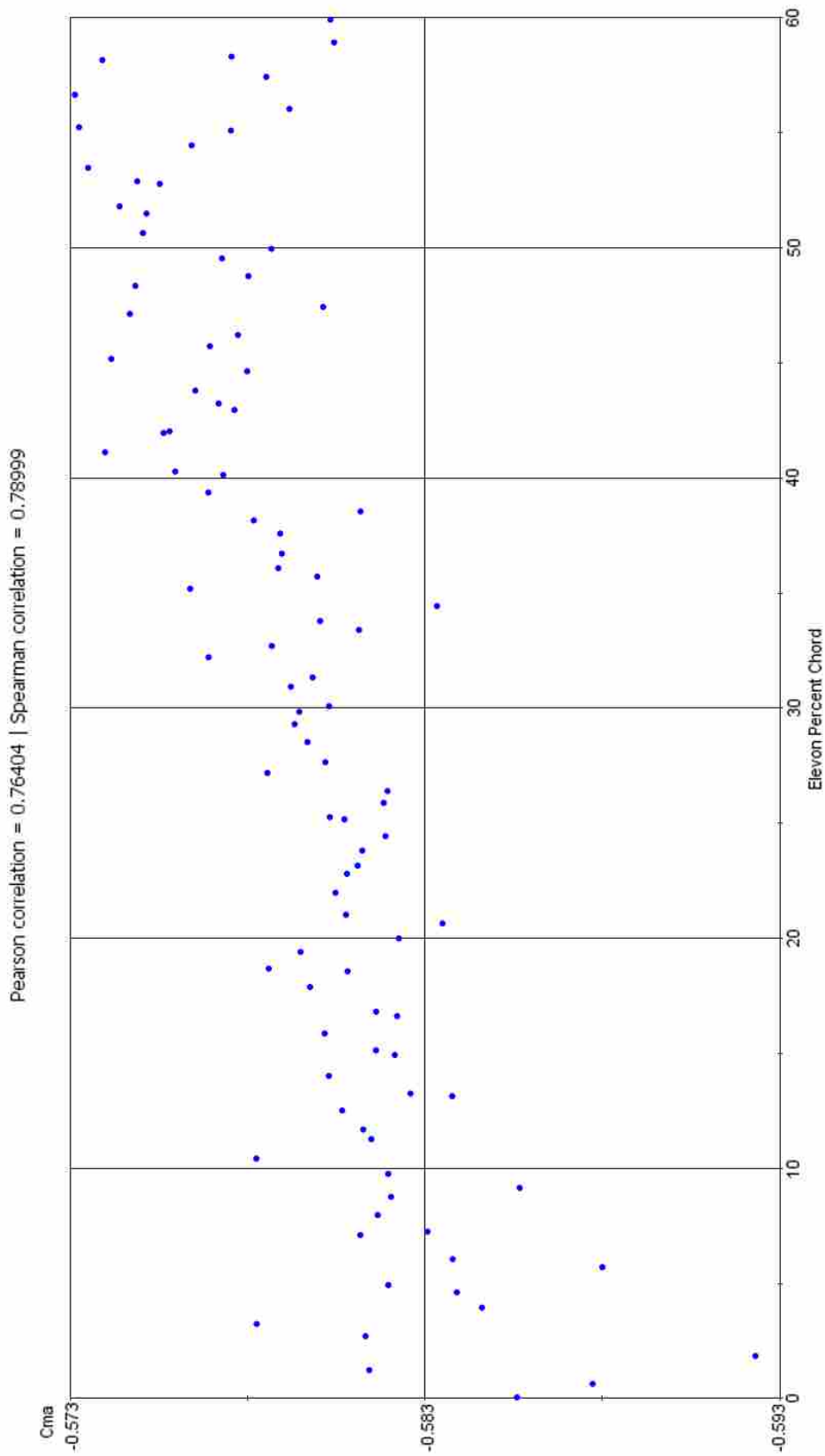


Figure 5.5: A plot from the DOE comparing change in pitching moment coefficient with change in elevon percent chord. There is a clear decrease in pitching moment negative slope with increasing elevon percent chord, even after considering variation from other parameters.

Pearson (Spearman)	Wing 1-1 Airfoil	Elevon Percent Chord	AlphaStab	CDiffStab	Clb	Cma	Cnb	Static Margin
Wing 1-1 Airfoil	1.000 (1.000)	-0.056 (-0.055)	-0.118 (-0.104)	0.566 (0.553)	0.276 (0.265)	0.129 (0.136)	-0.110 (-0.104)	-0.105 (-0.076)
Elevon Percent Chord	-0.056 (-0.055)	1.000 (1.000)	0.684 (0.689)	0.153 (0.155)	0.573 (0.571)	0.764 (0.790)	0.339 (0.343)	-0.790 (-0.813)
AlphaStab	-0.118 (-0.104)	0.684 (0.689)	1.000 (1.000)	0.251 (0.270)	0.680 (0.645)	0.768 (0.831)	0.786 (0.788)	-0.842 (-0.874)
CDiffStab	0.566 (0.553)	0.153 (0.155)	0.251 (0.270)	1.000 (1.000)	0.785 (0.800)	0.304 (0.413)	0.098 (0.113)	-0.386 (-0.423)
Clb	0.276 (0.265)	0.573 (0.571)	0.680 (0.645)	0.785 (0.800)	1.000 (1.000)	0.654 (0.707)	0.309 (0.287)	-0.770 (-0.761)
Cma	0.129 (0.136)	0.764 (0.790)	0.768 (0.831)	0.304 (0.413)	0.654 (0.707)	1.000 (1.000)	0.488 (0.536)	-0.979 (-0.988)
Cnb	-0.110 (-0.104)	0.339 (0.343)	0.786 (0.788)	0.098 (0.113)	0.309 (0.287)	0.488 (0.536)	1.000 (1.000)	-0.525 (-0.566)
Static Margin	-0.105 (-0.076)	-0.790 (-0.813)	-0.842 (-0.874)	-0.386 (-0.423)	-0.770 (-0.761)	-0.979 (-0.988)	-0.525 (-0.566)	1.000 (1.000)

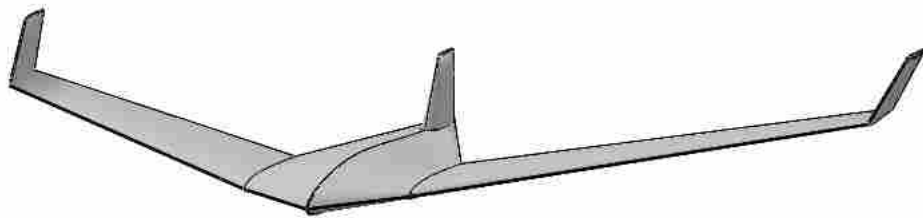
Figure 5.6: A table showing the sensitivity of a subset of the parameters to each other. The parameters with the highest sensitivity to each other have higher values (warmer colors). The diagonal should be neglected since it represents the sensitivity of a parameter to itself.

the SD 7003 in favor of the Eppler 330, and moving the Eppler 186 to the center wing section. Although it is difficult to explain the reasoning behind the airfoil selections, the Eppler 330 and 186 both see near minimums in their drag polar at a lift coefficient of 0.3, which may have contributed to their fitness in the optimization. The profiles of the airfoils used, along with the results from an XFOIL analysis at a Reynolds number of 50,000 are shown in Appendix D. The angle of attack at cruise condition was improved from 7.5° to 6.5° , and the elevator portion of the elevon deflection has decreased dramatically from -10° to a little more than -1° . Figures 5.7a and 5.7b show respectively the starting and ending points of the optimization. It is clear that the optimization has not taken into account enough of the influences of other analysis disciplines to produce a feasible design. Despite this, the goals of the optimization have been accomplished, which are to

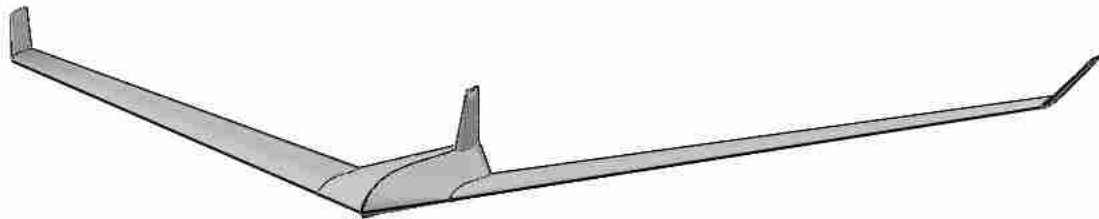
Table 5.2: The starting point and optimum point of the differential evolutionary algorithm. The optimum was in the 556th experiment.

	Start	End [556]
Inputs		
Elevon Percent Chord	30	39.25069
Main Wing, Section 1 Sweep	30	34.86905
Main Wing, Section 2 Aspect Ratio	4	6.545918
Main Wing, Section 3 Dihedral	75	50.67122
Main Wing Root Airfoil	Eppler 186	Eppler 330
Main Wing, Joint 1 Airfoil	SD 7003	Eppler 186
Main Wing, Joint 2 Airfoil	SD 7003	Eppler 330
Outputs		
Alpha of Attack	7.40121	6.58965
Coefficient of Drag	0.00616	0.00549
Elevator Deflection	-10.0072	-1.37942
C_{l_β}	-0.11403	-0.13351
C_{m_α}	-0.58142	-0.58328
C_{n_β}	0.02754	0.026769
Static Margin	0.166353	0.167492
GOAL	1.00E+08	1922762

broaden the learning of the designers and provide them a tool to study the influence of their chosen parameters to the performance.

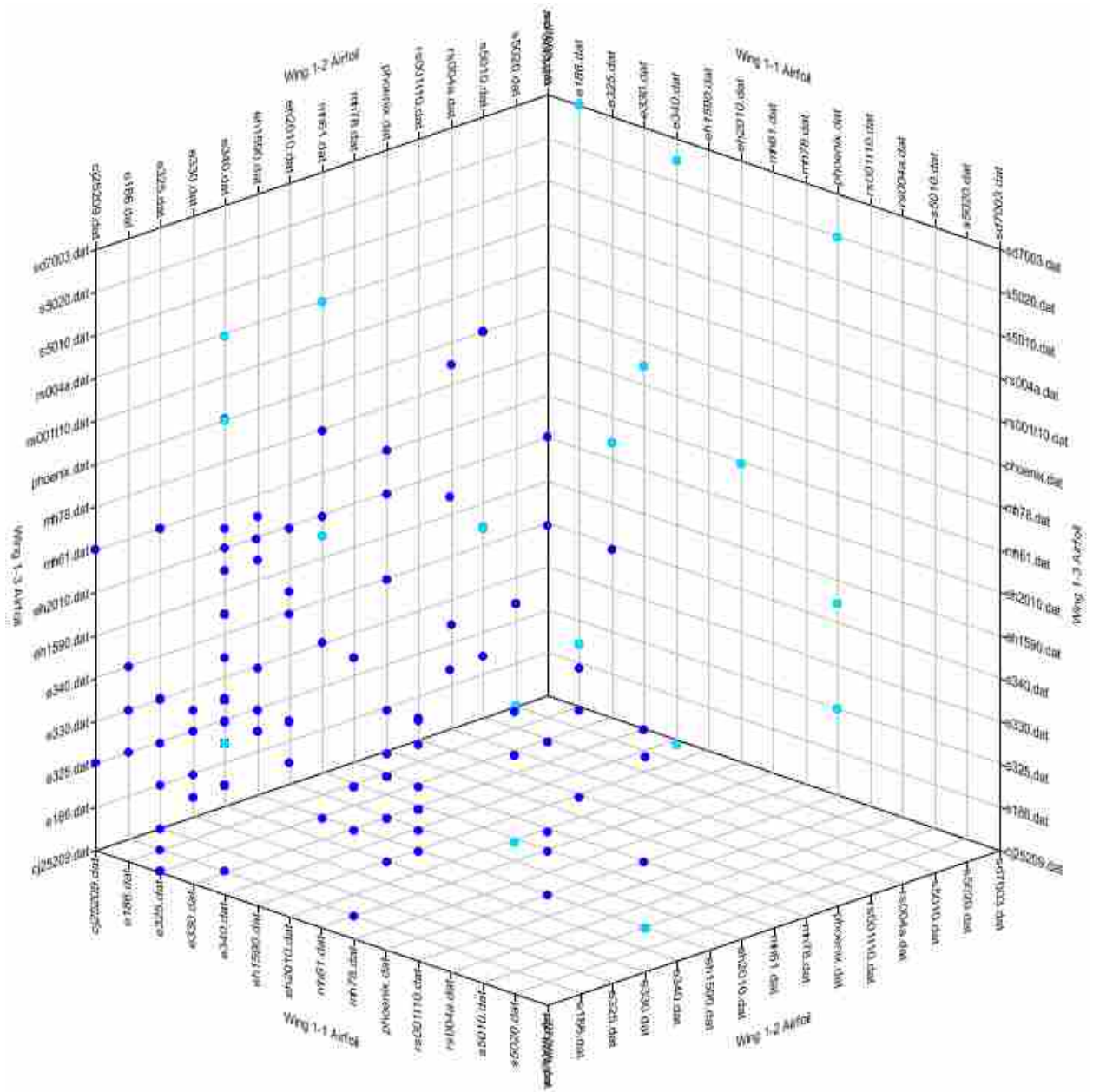


(a) Best UAV of first generation.



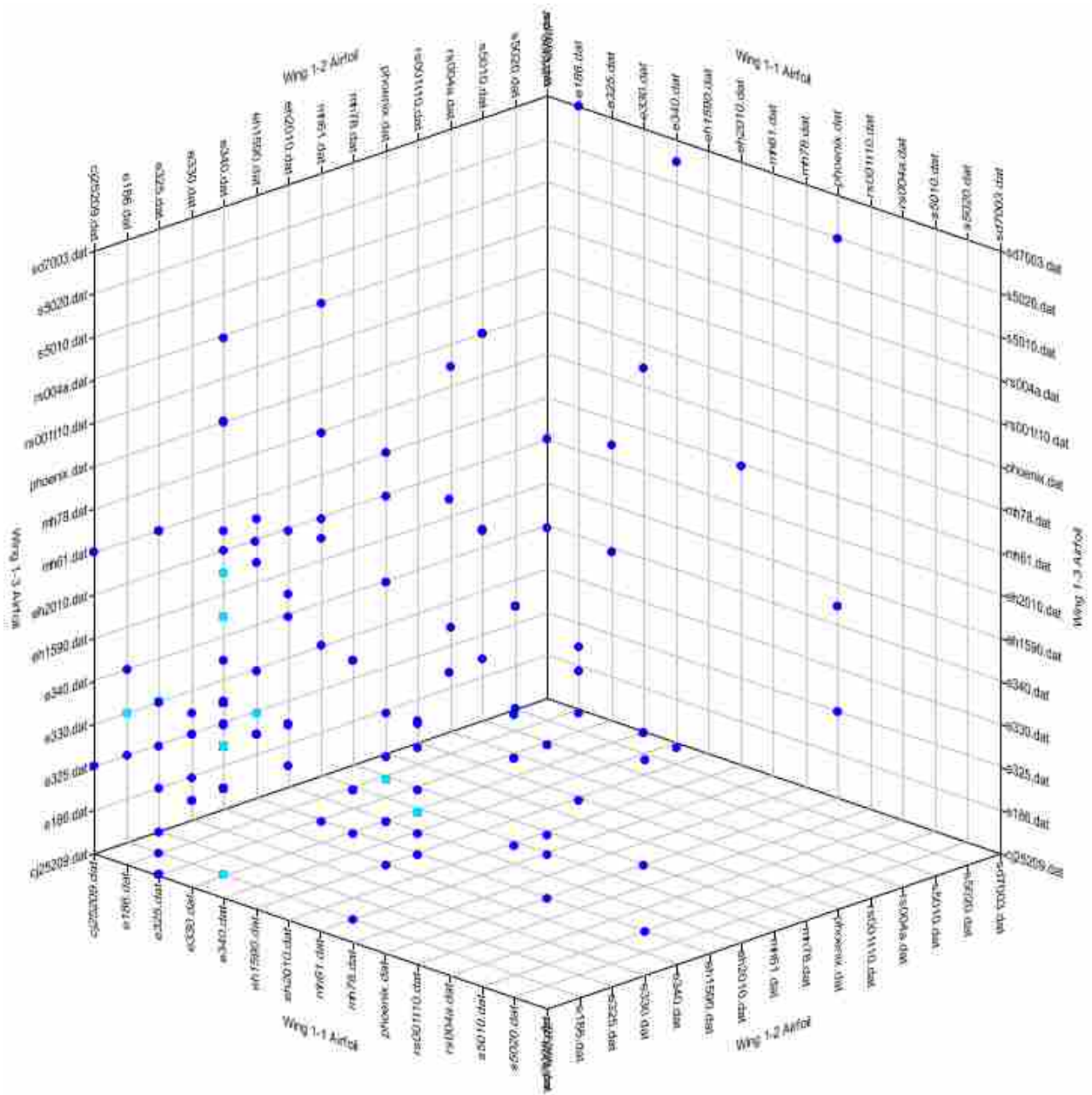
(b) Best UAV of final generation.

Figure 5.7: Visualization of the UAV design geometry used for the proof-of-concept optimization study.

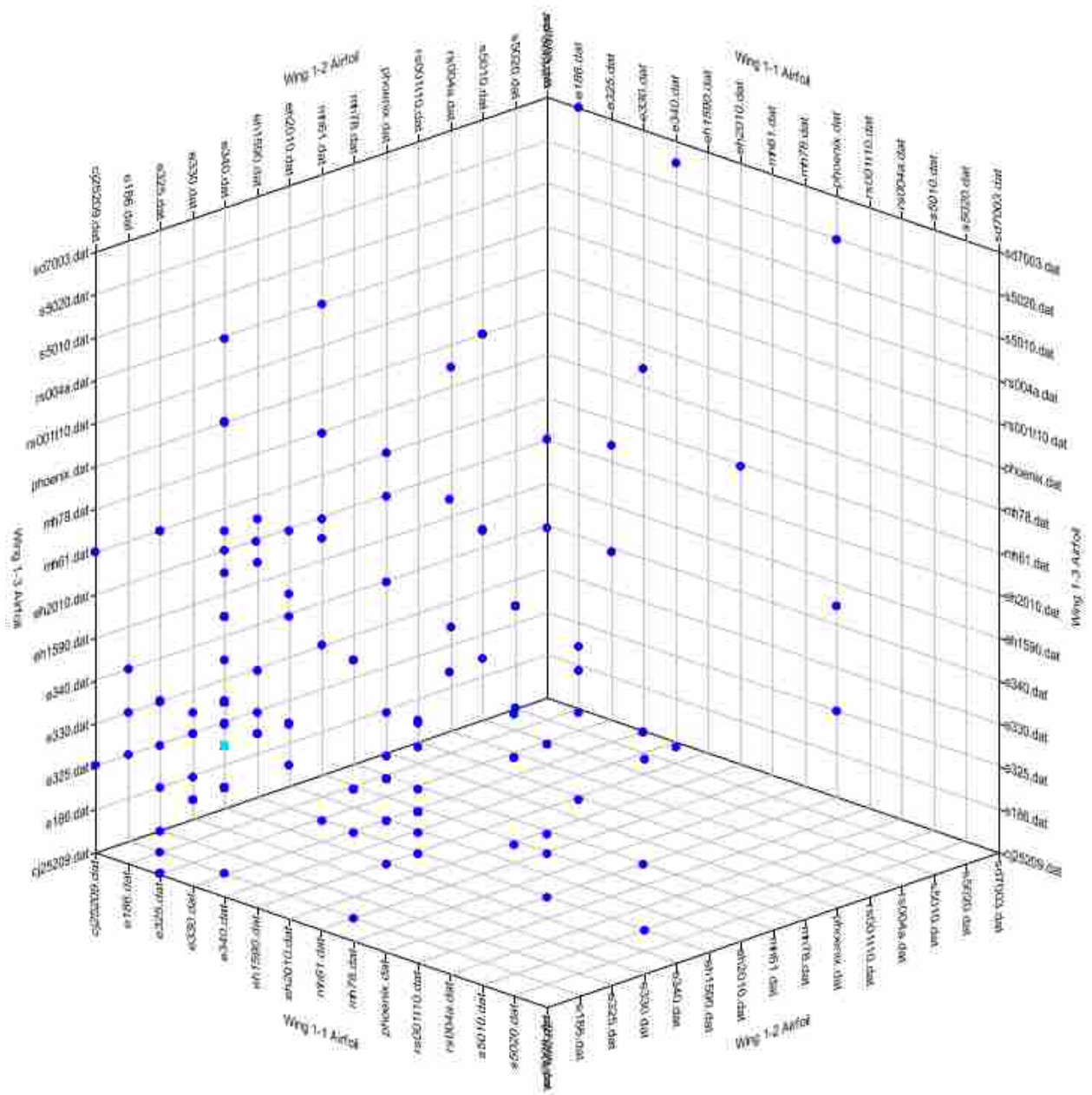


(a) First generation.

Figure 5.8: Three-dimensional scatter plots showing the progression of the genetic algorithm through the design space. All experiments are shown in blue. Generations are highlighted in turquoise.



(b) 7th generation.



(c) Final generation (30).

As an example of this learning, see Figure 5.8. This three-dimensional scatter plot shows the progression of the genetic algorithm in the selection of airfoils for each of the inner wing joints. At first, the distribution is wide and fills the design space. By the seventh generation, shown in Figure 5.8b, the population has narrowed down to a much smaller region. Past the twelfth generation, the population has found the final combination of airfoils, and only a few experiments continue to explore other combinations via mutation. Hence, the population converges quite rapidly to the optimum shown in Figure 5.8c. This can give students confidence in their selection, and also open their minds to possibilities and combinations that may have been difficult to pinpoint without a computer-driven optimization.

The progress of the genetic algorithm is further detailed in Figure 5.9, where several charts showing parameters varying over all the generations are shown. The braces show the diversity in the population at each generation, and the blue line shows the changes in the optimum value of each generation. Where bounds are given for constraints or design variables, the upper bound is shown in turquoise, and the lower bound in magenta. A target line associated with the elevator deflection design objective, is shown in green. This shows how the algorithm is exploring the design space, with the population having a very wide range spanning the bounds at the first generation, to a very narrow set of designs at the final iteration.

This proof-of concept has shown that the results of a computer-driven DOE or optimization can have a profound influence on the learning of students and their understanding of their own design. They can easily discover what the strengths of their design are, and what influence their design decisions have on the performance of the UAV. Inclusion of multiple analysis disciplines is necessary for the optimization to converge on an acceptable design, and is a point of future work. However, even a DOE that includes a single discipline or smaller set of disciplines can help students make more intelligent engineering decisions during the conceptual design. The automation and parametric generation of the Optimus workflow for CCADE makes this possible by providing access to these methods to students who are likely unfamiliar with operation of the software or even the optimization methods themselves. Such a tool introduces them to optimization methods and teaches them to interpret the results in a productive and useful way.

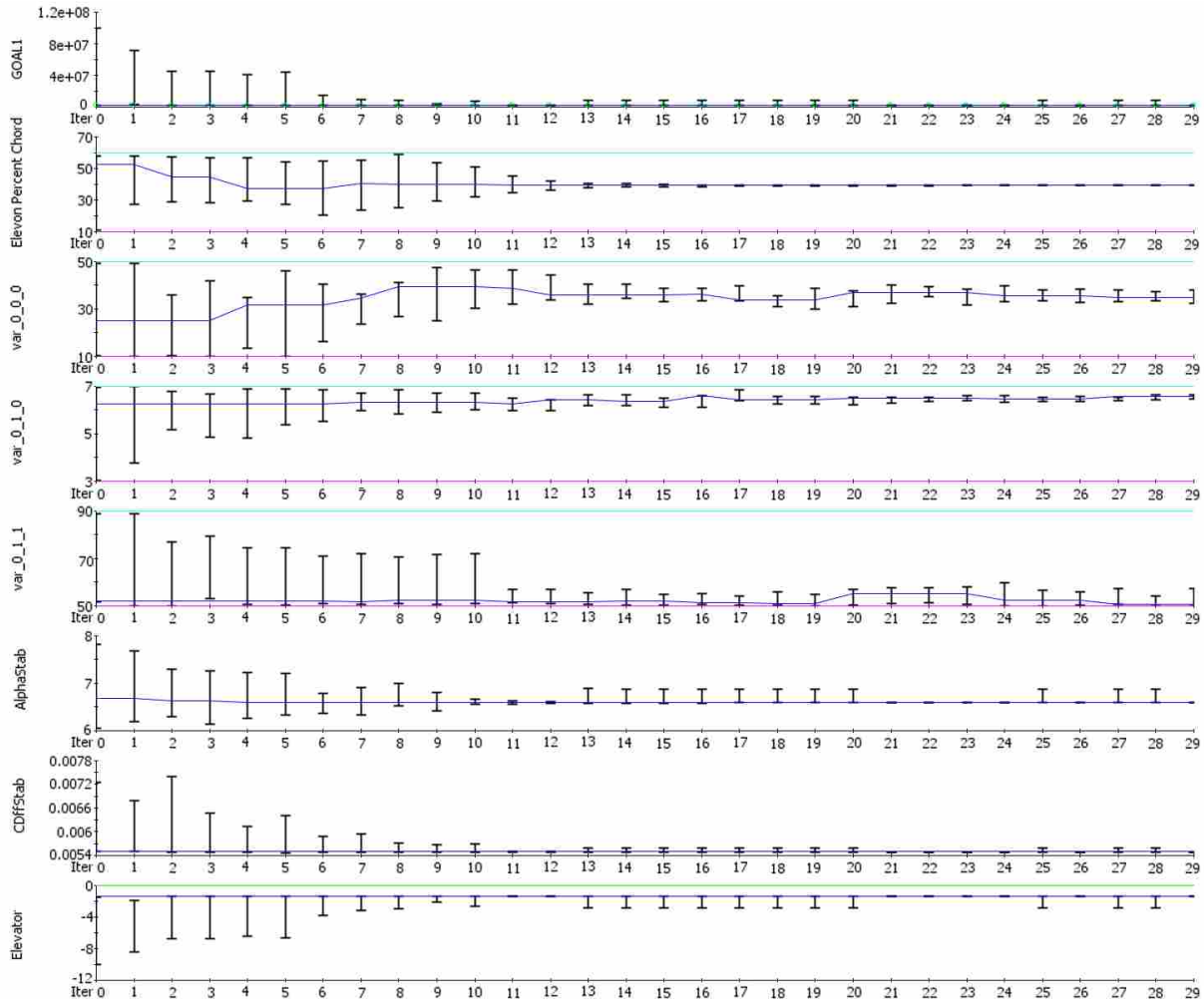


Figure 5.9: The optimization progress by generation (iteration). Each parameter is shown as a separate plot, with braces showing the diversity of the population, and the upper/lower bounds and target values shown. Parameters shown are, from top to bottom, the Optimus-defined objective function, elevon percent chord, section 1 sweep, section 2 aspect ratio, section 3 dihedral, angle of attack at cruise, total drag coefficient, and elevator deflection.

CHAPTER 6. IMPLEMENTATION IN AEROSPACE

This chapter will examine the implementation of CCADE into the AerosPACE course, and the resulting effects on students' learning. One of the primary objectives of CCADE is to serve as an educational tool, to help students more effectively traverse the conceptual design, to enhance their understanding of fundamental principles and relationships that govern the conceptual design process, and to assist collaboration, bolster teamwork, and bridge gaps between geographically dispersed team members. First, the CCADE's implementation in the 2014-2015 AerosPACE course is described. Then the experience of the students using CCADE is presented, based on feedback and observations while collaboratively using the software. Finally, suggestions for the implementation of CCADE in future projects is given, taken from lessons learned from the 2014-2015 course experience.

6.1 Course Implementation

Development of CCADE was ongoing parallel to instruction during the AerosPACE 2014-2015 course. Students were expected to use CCADE during the first 10 weeks of the first semester, allocated for the conceptual and preliminary design. During this time, student activity was monitored to evaluate student use of the software, ensure that bugs encountered were resolved in a timely manner, and see what benefits were gleaned from using CCADE during the conceptual design.

CCADE was introduced to the students in a two-hour lab, where the background and motivation were explained and a demonstration of its capabilities given. Since at this point, CCADE only had the base architecture and analysis modules for mission profile generation, Sizing, and Geometry, the lab only covered a high-level description of the CCADE workflow and detailed demonstration of the existing modules, similar to the overview given in the User Manual in Appendix B. Mission Profile, Initial Sizing and Constraints, and Geometry modules. The lab was

given at the beginning of the semester after formal lectures teaching the students to perform the analysis of these modules individually and manually. Concurrent development of CCADE was scheduled to correspond with the AerosPACE lecture schedule such that each analysis module was available at least as soon as the principles had been taught in a formal lecture. As the semester progressed, the benefits of using each analysis module in CCADE were presented during lectures introducing the principles, but time did not typically allow for further demonstrations.

In each analysis module, CCADE provides a link to the lecture which explains the background of the analysis being performed and the design principles involved. As the semester progressed, these links were updated with updated versions of CCADE as the corresponding analysis modules became available. This was intended to help students make connections between the principles they are learned in course lectures and their practical design work outside of class.

Help sessions were held with individual students and teams to encourage use and show how CCADE could increase productivity. All AerosPACE students were invited to attend the help sessions, held via an online conferencing service, but attendance was not required. Of 44 total students in the course, 11 attended at least one CCADE help session. Individual sessions saw between three and six attendees at four scheduled help sessions. An additional five sessions were held for individual students and small teams requesting help. Few enough students attended at any time so that attention could be given to individual problems, walking students through methods of using the software to accomplish their design studies. This one-on-one environment was found to be particularly helpful to both students and developer, since both were viewing the design simultaneously, stimulating important questions concerning the principles behind the analyses as well as the methods. Often, communication during these help sessions focused more on design principles and relationships between UAV design parameters than on actual use of the software. During the sessions, bugs in the software were discovered via collaboration because of unexpected use cases and differences in student hardware or file structure while running CCADE. The problems found were quickly addressed and solutions implemented in future versions of CCADE.

6.2 Student Feedback

A total of 18 distinct designs were created and modified by 18 total student user profiles within CCADE. These designs were not one-to-one with users. Rather, multiple designs were

created by some users, and some users simply modified existing designs with their team members. Unfortunately, the combination of bugs in the software, lack of access to all analysis modules at the beginning of the semester, and lack of a User Manual to help students use and apply the software contributed to premature student abandonment of CCADE. Only three of the designs were analyzed completely using CCADE, and with the exception of one, these were done under supervision by the author to help complete the analysis. While students did not use CCADE as a primary workspace for evaluating designs, they did utilize various aspects of it to improve their productivity. For example, students reported using the software to coordinate sizing activities, share geometry with each other, perform trade studies with airfoils, and size control surfaces. Especially useful for the students was the ability to automatically generate input files for AVL. This feature was used extensively to run AVL from CCADE, and to generate template files that were used in independent trade studies by the students. This saved students time and allowed them to focus on the principles of the design. Many of the suggestions in Section 6.3 are aimed at increasing student acceptance of CCADE and equipping them to fully utilize its capabilities. Continued development of CCADE will reduce the occurrence of bugs which hinder student progress on the design.

Because CCADE was undergoing heavy development during the course, there were no reported team-wide design sessions held during the 2014-2015 year. However, students reported using the software in smaller groups for design activities. The tiered structure of CCADE made information sharing between these small groups of team members much easier, and improved the transparency of the analyses. Typically students met virtually to collaboratively do design work using the software, and spoke with each other over microphone to coordinate their efforts. Students found the collaborative features of CCADE useful for sharing information in an intuitive and convenient way.

Although CCADE might not have been used as extensively as intended, those who did use it reported that it stimulated creative thought and enabled a more rapid assessment of a UAV configuration concept. In the end, this contributed to a larger volume of concepts considered, by enhancing the ability of teams to collectively work on the design. It also contributed to students' success under tighter time constraints, as critical design reviews were held earlier in the semester, requiring earlier design prototyping. As CCADE continues to be developed, it is expected that these benefits will increase, improving the students' learning and productivity.

6.3 Suggestions for Future Implementation

The goal in future AerosPACE courses is to increase student acceptance and use of CCADE. This section focuses on work that has been done since the 2014-2015 AerosPACE conceptual design or is suggested for future work to improve the student experience in support of their productivity using CCADE.

One way to encourage this is to reduce the frequency of encountering progress-hampering bugs in the software. This will come with continued development and testing, and is considered key to overcoming student abandonment of CCADE. When the students are able to trust that the program will not unexpectedly close or produce error messages in normal situations, they are less likely to get frustrated with the software and more likely recognize the value of it.

During student help sessions, it was previously mentioned that a considerable amount of time was spent reviewing design principles and understanding the results of analyses in context of the design. This experience motivated a change in the software that enabled faculty to view and access all student team designs. Previously, they were required to create a user profile which was only had access to a single team's designs. In the future, faculty may find CCADE to be an excellent environment for small-group instruction and active learning. Since results from analyses are so quickly and easily accessible, CCADE is an ideal setting for guided practice and experimentation. In addition, faculty support of the software will encourage student use of CCADE.

The User Manual in Appendix B was written after the conclusion of the 2014-2015 AerosPACE conceptual design. It details each of the features of CCADE and gives examples of use cases, recommended upper and lower bounds of inputs and suggestions for interpreting outputs. The manual is expected to greatly enhance the ability of students to effectively use CCADE. In addition, a series of short tutorial videos demonstrating the use of each module in CCADE are suggested as a supplement to this User Manual, which can be shown during lectures or assigned to students for viewing outside of lecture. Such videos could answer many of the most common questions or give tips or best practices for using the software to increase productivity and teamwork.

It is recommended that all lectures covering fundamental principles of the design mention how the features of CCADE enable certain analyses and provide a venue for every discipline of design. The following lab session could involve a demonstration of how to use CCADE for that particular analysis, as well as an interactive activity inviting student to work collaboratively with

their peers conducting practice analyses. By interpreting results together with their faculty and teammates, this would solidify learning from formal instruction, and help students understand exactly how to go about the design process. This would give students practical experience utilizing design principles, as well as increase their trust and acceptance of CCADE.

CHAPTER 7. CONCLUSION

After a brief summary of the work of this thesis, this chapter discusses the impact of CCADE on the AerosPACE course, broader academia, and industry. Finally, suggestions are given for future work involving CCADE, including additions to be made in the code and methods for implementation in future AerosPACE courses.

7.1 Summary

A collaborative conceptual aircraft design environment has been developed which bridges gaps in dispersed university teams in the AerosPACE course by providing a work-space for them collaboratively generate and evaluate concepts for small scale, electric UAVs. In this unique design and analysis environment, students are provided with tools to rapidly analyze a wide range of UAV configurations. These design activities can be conducted simultaneously among team members using a shared database model of the design, which allows them to teach and learn from each other, as well as significantly improve communication and data sharing between engineering disciplines. The conceptual design workflow is encapsulated in a user-friendly graphical environment which automatically generates input files for open-source analysis tools, and interprets the output for display to the user in an intuitive way. Geographically dispersed teams are connected in a client-server context and can collaboratively manipulate a UAV design via persistent database models. The user interface of CCADE takes advantage of the distributive expertise of the design team and allows for both concurrent and independent design. The software demonstrated the ability to significantly increase the number of conceptual design configurations analyzed and bolster student understanding of fundamental aircraft design principles during the 2014-2015 AerosPACE course. This interactive, immersive design environment enables greater exploration of the available design space, helps strengthen team relationships, increases productivity, and contributes to a better overall UAV concept to advance to preliminary and detailed design.

CCADE is equipped with design and analysis modules for setting up mission requirements in the form of a mission profile, performing initial constraint and weight sizing, uploading and viewing concept geometry, selecting appropriate airfoils, and assessing the aerodynamic, stability, propulsion, and structural performance of a UAV design configuration. Its architecture allows for a wide range of configurations, enabling creativity among teams to explore beyond the traditional wing-and-tube configuration. By automating input, execution, and communication between analysis tools including Excel worksheets, OpenVSP, XFOIL, and AVL, CCADE relieves the burdens of data management and coding on the part of the designer.

Although introduced to students during the 2014-2015 AerosPACE course, CCADE had a number of challenges in implementation that caused usage of the software to be less than ideal. Further development of the software, documentation of a User Manual (see Appendix B), and more focus on CCADE during course lectures and labs are expected to improve student acceptance and use. With these improvements and increased student, faculty, and coach use of the software, CCADE is expected to prove a valuable addition to the AerosPACE course.

7.2 Impact of CCADE

Although focused on application within the AerosPACE capstone course, CCADE's contributions also have implications for industry and in the larger body of collaborative CAx research. Lessons learned in this research can be applied to other collaborative senior design projects, conceptual design team data management in aerospace industry settings, and other computer-aided design tools seeking to promote greater collaboration among users. This section will examine the impact of CCADE in AerosPACE and its general contributions to workflow automation, collaborative design, and design exploration and optimization.

7.2.1 Workflow Automation

In the AerosPACE course, students coming into the course may have little to no experience in aircraft design, but the semester schedule requires rapid generation and evaluation of concepts in preparation for preliminary and detailed design. By encapsulating the design workflow as described in Chapter 4, CCADE automates much of the tedious input file preparation and scripting of

analysis tools. Normally, these tasks consume a significant percentage of the designer's time during the conceptual design phase. When multiple team members are performing the same tasks in parallel, repetition of this input file preparation can significantly slow down productivity. CCADE allows the designers to access the results of analysis software without any coding expertise or experience using the respective software interfaces. By providing quick and easy access to the most relevant output data, students are taught which outputs to look for and how key aircraft design parameters influence the performance of their vehicle. In this way, CCADE enables a greater emphasis on design and the fundamental engineering principles, freeing the design from spending valuable time wrestling with the analysis codes themselves. This enables designers do what they do best, and allow them to investigate concepts on the edges of the design space that might otherwise be sacrificed to meet a demanding schedule. Furthermore, by stepping students through the design workflow, CCADE familiarizes them with the inputs and outputs involved in the entire design process. Users are exposed to the entire conceptual design workflow including every major discipline, helping them become aware of the most important parameters, and which disciplines are involved with calculating or consuming the data. They are then better equipped to negotiate the trade-offs between disciplines to achieve design objectives at later stages of the design.

7.2.2 Collaborative Design

Collaboration between designers separated by geographic location is not an easy skill to learn. During AerosPACE, students are asked for perhaps the first time to work together with a team of students from different universities in various engineering disciplines. It is important that students share their knowledge and help each other grasp the necessary concepts for aircraft design. By putting students together in a technical design environment, CCADE helps students share their learning in a more relevant environment than a traditional team meeting. In such an environment, team members who might otherwise feel isolated by the geographically dispersed nature of the team or their own unfamiliarity with aircraft design may feel more included and find it easier to significantly contribute.

CCADE has data sharing built into its architecture. By storing all design and analysis data in a centralized database, there is no need to hunt down misplaced files or hound a teammate for his or her results. Furthermore, CCADE's interface provides an environment for simultaneously

collaborating with their peers and manipulating the design. This enables a higher level of communication where discussion is focused on the key design parameters and where decisions made by the team can be immediately evaluated. This sort of discussion has the potential to shrink the conceptual design and reduce the number of design turn-backs required later on due to poor communication.

7.2.3 Design Exploration and Optimization

Often, due to the overhead involved with learning to use optimization software and setting up the appropriate workflow, time constraints force optimization during the conceptual design to be performed manually. By automating the generation of an optimization workflow using inputs and outputs specified by the user, CCADE provides a framework that gives designers access to optimization methods without any prior experience or expertise. This enables detailed trade studies which give the designers insight into the sensitivity of the design to various parameters. Beyond saving time and improving learning, the results of an optimization or design of experiments can help to significantly improve the end design.

7.3 Future Work

Development of CCADE is ongoing, as it is expected that the software will be used in future AerosPACE courses or similar projects as a conceptual design tool. The weights and performance modules of the software are still under development. With the completion of these modules, the entire planned workflow will be complete, and students will have access to the full range of conceptual design activities. The weight module will involve a simple calculation of center of gravity based on the weights defined for different components earlier in the workflow. Users will have the opportunity to adjust the placement of components such as the motor, batteries, payload, etc. in order to ensure aircraft stability. The performance module will involve a calculation of the actual power loading and wing loading based on the actual weight of the aircraft, the calculated motor power, and the planform area specified in the OpenVSP geometry. These parameters can then be used to verify that the design requirements have been met, as well as provide range and endurance estimates.

The current constraint analysis is limited to a small set of constraints, which could be expanded upon or defined in a more free-form manner to allow users to define their own constraints. Analytic calculation of velocities at minimum drag and minimum power could be added, as well as a separate constraint series shown on the chart for the stall speed. In particular, adding a constraint reflecting a hand-launch of the aircraft would be a valuable improvement, since this was a design requirement for UAVs designed for the 2014-2015 AerosPACE course.

The design exploration module described in Chapter 5 also requires additional work to completely implement it in CCADE. The work of automating the Optimus workflow generation must be completed for the remaining analysis disciplines, and the Optimus web service must be developed to accept jobs from multiple clients and execute the optimization. Progress logs could be sent back to the team while the optimization routine is running, so that proper execution is ensured. This infrastructure will eliminate the need for multiple installations and licenses of Optimus among team members and provide an easy way for students to schedule their optimization processes.

Another area of future work might identify dependent analyses in the workflow and the optimum order of their execution. Then, when a design input is modified, a background process might run these dependent analyses automatically and thus immediately show the full impact of the design change.

Currently, the list of airfoils to analyze and apply to the design includes only airfoils in the UIUC database. The list is not maintained, and thus additions to the UIUC database are not automatically included in future versions. Future versions of CCADE could generate an airfoil data file by specification of NACA airfoil number, or allow the user to upload a custom airfoil in order to circumvent this disability. Such capability would ensure that designers always have access to methods for analyzing the best airfoils for their particular UAV design.

The geometry module in CCADE currently only supports OpenVSP version 2.9. Version 3 of OpenVSP, released after the first semester of the 2014-2015 AerosPACE course, included several significant improvements, including an API which enabled the automation workflow of the design exploration module proof-of-concept presented in Section 5.3. However, the new version adopted a file format incompatible with previous versions of OpenVSP, and cannot be read by the CCADE geometry module. It is recommended that the methods developed for the optimization

proof-of-concept be applied to the geometry module to enable this functionality, and instruct the user that only version 3 models are able to function in the design exploration module.

The current chat box available for users to communicate with each other in CCADE does not divide users into teams or groups working on the same design. Custom work groups and other chat enhancements might be prudent in future versions.

Improvements in notifications to clients viewing the same page are also a point of ongoing development. Currently the functionality of updating the display for all clients viewing the same analysis module in CCADE exists only for students running the software on the BYU network. The functionality must be extended to include out-of-network users such that the entire team can receive immediate updates on changes made to the design within CCADE. This will alleviate users from the current requirement of clicking out of an analysis module and then returning back into it to view changes made by another user. Future tests with multiple users should also be conducted in order to improve the experience and address unexpected needs for synchronous design.

Given the nature of storing data within CCADE, the architecture would naturally lend itself to future research in analytics that could assist in intelligent selection of key sizing parameters in future courses, team member evaluation, or intelligent team formation.

REFERENCES

- [1] Raymer, D. P., 2012. *Aircraft Design: A Conceptual Approach*, 5th ed. American Institute of Aeronautics and Astronautics, Reston, Virginia. viii, 9, 11, 34, 46
- [2] Senn, C., 2013. N-Tier Entity Framework User Guide. <http://ntierref.codeplex.com/>. viii, 32, 33
- [3] Hedden, C. R., Blakey, M., Bush, W., Grumman, N., Dannenberg, K., Langford, J., and Flight, A., 2012. "Aviation Week Workforce Study 2012". *Aviation Week*. 1
- [4] Attwood, K., 2006. "Bridging the skills gap". *Electronics World*, **112**, p. 11. 1
- [5] Dutson, A. J., Todd, R. H., Magleby, S. P., and Sorensen, C. D., 1997. "A Review of Literature on Teaching Engineering Design Through Project- Oriented Capstone Courses". *Journal of Engineering Education*, Jan. 1, 19
- [6] Zender, F., Wright, M., Richey, M., Gorrell, S., Mcpherson, K., and Madni, A. M., 2014. "Aerospace Partners for the Advancement of Collaborative Engineering (AerosPACE) - Connecting Industry and Academia through a Novel Capstone Course". In International Conference on E-Learning in the Workplace, New York City, New York. 1
- [7] Danielson, S., Kirkpatrick, A., and Ervin, E., 2011. "ASME vision 2030: Helping to inform mechanical engineering education". In Proceedings - Frontiers in Education Conference, FIE 2011-6143065, Rapid City, South Dakota. 1
- [8] Gorrell, S., Jensen, C. G., Stone, B., Red, E., Richey, M., Zender, F., Wright, M., French, D. E., Hayashibara, S., Johnson, C., and Sullivan, J., 2014. "Aerospace Partners for the Advancement of Collaborative Engineering". In 121st ASEE Annual Conference & Exposition, Paper ID #9767, Indianapolis, Indiana, American Society for Engineering Education. 1
- [9] Brandt, S. A., Stiles, R. J., Bertin, J. J., and Whitford, R., 2004. *Introduction to Aeronautics: A Design Perspective*, 2nd ed. American Institute of Aeronautics and Astronautics, Reston, Virginia. 9, 38, 42
- [10] MacDonald, R., and Gloudemans, J., 2015. Open Vehicle Sketch Pad. <http://www.openvsp.org/>. 10
- [11] Fredericks, W. J., Antcliff, K. R., Costa, G., Deshpande, N., Moore, M. D., Miguel, E. A. S., and Snyder, A. N., 2010. "Aircraft Conceptual Design Using Vehicle Sketch Pad". In 46th AIAA Aerospace Sciences Meeting, Paper 2010-658, Orlando, Florida. 12
- [12] AVID, 2015. *AVID PAGE (Parametric Aircraft Geometry Engine)*. <http://www.avid aerospace.com/software/avid-page>. 12

- [13] Eller, D., 2009. SUMO - Aircraft Modeling & Mesh Generation. <http://www.larosterna.com/sumo.html>. 12
- [14] Rodriguez, D. L., and Sturdza, P., 2006. "A Rapid Geometry Engine for Preliminary Aircraft Design". In 44th AIAA Aerospace Sciences Meeting and Exhibit, Paper 2006-929. 12
- [15] MacDonald, R. Openvsp 3.0.4 released. <http://openvsp.org/blogs/announcements/2015/02/18/openvsp-3-0-4-released>. 12
- [16] Drela, M., and Youngren, H., 2013. XFOIL. http://web.mit.edu/drela/Public/web/xfoil/xfoil_doc.txt. 12
- [17] Drela, M., and Youngren, H., 2014. AVL - Athena Vortex Lattice. <http://web.mit.edu/drela/Public/web/avl/>. 13
- [18] Deperrois, A., 2014. XFLR5 - Analysis of Foils and Wings Operating at Low Reynolds Numbers. <http://www.xflr5.com/xflr5.htm>. 13
- [19] Melin, T., 2000. "A vortex lattice MATLAB implementation for linear aerodynamic wing applications". Master's thesis, Royal Institute of Technology (KTH), Dec. <http://www.redhammer.se/tornado/thesis.pdf>. 13
- [20] Filkovic, D., 2008. "Graduate Work". PhD thesis, University of Zagreb. <http://www.3dpanelmethod.com/documents/Graduate%20Work.pdf>. 13
- [21] Vorview. <http://www.avid aerospace.com/avid-vorview/>. 13
- [22] MotoCalc. <http://www.motocalc.com/motocalc.htm>. 14
- [23] Giese, G., and Persson, C., 2010. Drive Calculator Manual. http://www.drivecalc.de/DC34/DCHelp/help_en.html. 14
- [24] McCormick, B. W., 1995. *Aerodynamics, aeronautics, and flight mechanics*. Wiley, New York. 14, 60, 64
- [25] Chaput, A. J., 2015. "Multi-section Wing Capability for the Vehicle Sketch Pad Structural Analysis Module". In 53rd AIAA Aerospace Sciences Meeting, SciTech. Jan. Paper 2015-1015. 14
- [26] Dhondt, G., and Wittig, K., 1998. Calculix. <http://www.calculix.de/>. 14
- [27] Roskam, J., and Malaek, S., 1989. "Automated Aircraft Configuration Design and Analysis". *SAE Technical Paper 891072*. 15
- [28] OPTIMAL AIRCRAFT DESIGN. *ADS - Aircraft Design Software*. <http://www.pca2000.com/en/software/ads-rapports-techniques.php>. 15
- [29] Simos, D., 2012. Piano: Project Interactive Analysis and Optimization. <http://www.piano.aero/>. 15
- [30] Zhang, M., 2011. "Application and Development of the CEASIOM-SUMO-EDGE Suite for Rapid AeroData Assessment of Aircraft Flying Qualities". Licentiate, KTH Royal Institute

- of Technology, Stockholm, Sweden. Paper 34345. <http://swepub.kb.se/bib/swepub:oai:DiVA.org:kth-34345>. 15
- [31] Böhnke, D., Rizzi, A., Zhang, M., and Nagel, B., 2013. “Towards a Collaborative and Integrated Set of Open Tools for Aircraft Design”. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Paper 2013-0222, Reston, Virginia. 15
- [32] Zill, T., Ciampa, P. D., and Nagel, B., 2012. “Multidisciplinary Design Optimization in a Collaborative Distributed Aircraft Design System”. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Paper 2012-0553, Nashville, Tennessee. 16
- [33] Deshpande, S., Watson, L. T., and Love, N. J., 2013. ADML : Aircraft Design Markup Language for Multidisciplinary Aircraft Design and Analysis. Tech. rep., Virginia Polytechnic Institute and State University - Department of Computer Science, Blacksburg, VA. <https://vtechworks.lib.vt.edu/handle/10919/24826>. 16
- [34] Hwang, H., Jung, K., Kang, I., Kim, M., Park, S., and Kim, J., 2006. “Multidisciplinary aircraft design and evaluation software integrating CAD, analysis, database, and optimization”. *Advances in Engineering Software*, 37(5), May, pp. 312–326. 16
- [35] Iqbal, L. U., Crossley, W. A., Weisshaar, T. A., and Sullivan, J. P., 2008. “Higher Level Design Methods Applied to the Conceptual Design of an MALE UAV”. In *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Paper 2008-5908, Victoria, British Columbia, Canada. 16
- [36] Iqbal, L. U., and Sullivan, J. P., 2008. “Application of an Integrated Approach to the UAV”. In *46th AIAA Aerospace Sciences Meeting*, Paper 2008-144, Reno, Nevada. 16
- [37] Al-shamma, O., and Ali, R., 2012. “Interactive Aircraft Design for Undergraduate Teaching”. In *Applied Aerodynamic Conference: Modelling & Simulation in Aerodynamic Design Process*, Bristol, UK, Royal Aeronautical Society. 16
- [38] Raymer, D., 1995. “RDS Professional: Aircraft Design on a Personal Computer”. *SAE Technical Paper 951160*. 17
- [39] DASSAULT SYSTÈMES, 2015. *Isight & SIMULIA Execution Engine*. <http://www.3ds.com/products-services/simulia/products/isight-simulia-execution-engine/>. 17
- [40] PHOENIX INTEGRATION, 2015. *PHX ModelCenter®*. <http://www.phoenix-int.com/software/phx-modelcenter.php>. 17
- [41] NOESIS SOLUTIONS, 2013. *Optimus*. Leuven, Belgium. <http://www.noesisolutions.com/Noesis/>. 17
- [42] Merkel, M., and Schumacher, a., 2003. “An Automated Optimization Process for a CAE Driven Product Development”. *Journal of Mechanical Design*, 125(4), p. 694. 17

- [43] Lu, S. C.-Y., Elmaraghy, W., Schuh, G., and Wilhelm, R., 2007. “A scientific foundation of collaborative engineering”. *Annals of the CIRP*, **56**(1), pp. 605–634. 18
- [44] Hammond, J., Koubek, R. J., and Harvey, C. M., 2001. “Distributed collaboration for engineering design: A review and reappraisal”. *Human Factors and Ergonomics in Manufacturing & Service Industries*, **11**(1), pp. 35–52. 18
- [45] Red, E., Holyoak, V., Jensen, C. G., Marshall, F., Ryskamp, J., and Xu, Y., 2010. “v -CAx : A Research Agenda for Collaborative Computer-Aided Applications”. *Computer-Aided Design and Applications*, **7**, pp. 1–18. 18
- [46] Red, E., Jensen, G., French, D., and Weerakoon, P., 2011. “Multi-user architectures for computer-aided engineering collaboration”. In *17th International Conference on Concurrent Enterprising*, Paper 12289975, pp. 1–10. 18
- [47] Richey, M., Zender, F., Schrage, D., Jensen, G., McPherson, B., Fehr, J., Symmonds, M., and French, D., 2012. “An Innovative Approach to an Integrated Design and Manufacturing Multi-site “Cloud-based” Capstone Project”. In *ASEE International Forum*, Paper GC 2012-5608. 18
- [48] Zender, F., Schrage, D., Richey, M., and Jensen, G., 2013. “Wing Design as a Symphony of Geographically Dispersed, Multi-disciplinary Undergraduate Students”. In *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Paper 2013-1503, Boston, Massachusetts, Structures, Structural Dynamics, and Materials and Co-located Conferences. 18
- [49] Amadori, K., Johansson, B., and Krus, P., 2007. “Using CAD-Tools and Aerodynamic Codes in a Distributed Conceptual Design Framework”. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, Paper 2007-964, Reno, Nevada. 18
- [50] Johansson, B., Jouannet, C., and Krus, P., 2003. “Distributed Aircraft Analysis Using Web Service Technology”. In *SAE World Aviation Congress & Exhibition*, SAE Technical Paper 2003-01-3007, Linköping, Sweden. 18
- [51] Fife, N. L., 2005. “Developing a Design Space Model Using a Multidisciplinary Design Optimization Schema in a Product Lifecycle Management System to Capture Knowledge for Reuse”. Master’s thesis, Brigham Young University. *All Theses and Dissertations*. Paper 261. <http://scholarsarchive.byu.edu/etd/261>. 19
- [52] Prince, M., 2004. “Does Active Learning Work? A Review of the Research”. *Journal of Engineering Education*, **93**(July), pp. 223–231. 19
- [53] Manuel, P. D., and Alghamdi, J., 2003. “A data-centric design for n -tier architecture”. *Information Sciences*, **150**(3-4), pp. 195–206. 20
- [54] Mattingly, J. D., 2002. *Aircraft Engine Design*. AIAA education series. American Institute of Aeronautics & Astronautics. 38
- [55] McClamroch, N., 2011. *Steady Aircraft Flight and Performance*. Steady Aircraft Flight and Performance. Princeton University Press. 38, 44

- [56] Bjorke, O., et al., 2015. *Helix Toolkit, 3D Toolkit for .NET*. <https://github.com/helix-toolkit>. 48
- [57] Granger, R., 2012. *Fluid Mechanics*. Dover Books on Physics. Dover Publications. 51
- [58] Selig, M. UIUC Airfoil Data Site. http://m-selig.ae.illinois.edu/ads/coord_database.html. 52
- [59] Drela, M., 1989. XFOIL: An analysis and design system for low Reynolds number airfoils. http://web.mit.edu/drela/Public/papers/xfoil_sv.pdf. 52
- [60] Lambden, M., 2014. Airfoil lift and drag polar diagrams. <http://airfoiltools.com/polar/index>. 53, 155
- [61] Drela, M., and Youngren, H., 2001. *XFOIL 6.9 User Primer*. http://web.mit.edu/aeroutil_v1.0/xfoil_doc.txt. 53
- [62] Sullivan, J. Personal Communication. 63

APPENDIX A. DATABASE STRUCTURE

In this appendix, entities within the CCADE database model are listed, organized by table. Table names are in blue at the top of each box, with entity names consisting of the table name after the period. Then properties of the entity are listed, with primary keys denoted by a “P” and foreign keys denoted by an “F” to the left of the property name. The data type of the property is given in green to the right of the property name, along with the corresponding length, if applicable. Finally, the primary key is highlighted once again at the bottom, followed by relationships to other entities. Arrows also signify relationships between entities, with the appropriate symbol showing the type of relationship according to Oracle relationship cardinality notation.

A.1 Independent Database Entities

The figure shows three screenshots of database entity definitions from the AerosPACE_CCADE_dbo schema. Each screenshot displays the table name at the top, followed by a list of properties with their data types and lengths. Primary keys are marked with 'P' and foreign keys with 'F'. Relationships are listed at the bottom of each entity definition.

AerosPACE_CCADE_dbo.Materials	
P *	Id INTEGER
*	Density FLOAT (53)
*	Modulus FLOAT (53)
*	Name NVARCHAR (50)
PK_Materials (Id)	

AerosPACE_CCADE_dbo.HelpUrls	
P *	Id INTEGER
*	Type VARCHAR (50)
*	Url VARCHAR (max)
PK_HelpUrls (Id)	

AerosPACE_CCADE_dbo.UAV_Designs	
P *	idUAV_Design INTEGER
*	Name VARCHAR (50)
F *	TeamId INTEGER
*	isDeleted INTEGER
PK_UAV_Designs (idUAV_Design)	
FK_UAV_Design_Team (TeamId)	
IX_FK_UAV_Design_Team (TeamId)	

Figure A.1: Entities which do not rely on relationships to other entities. UAV_Designs is the top-level entity for all designs, HelpUrls is a list of HTML links which make course lectures immediately available to students, and Materials is a source for all user- or administrator-defined materials.

A.2 User Management Entities

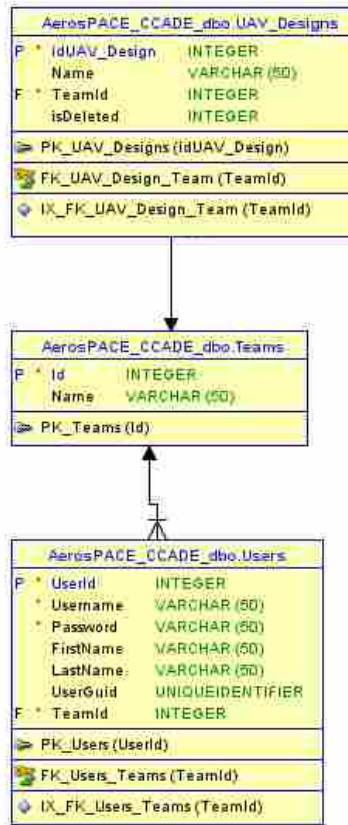


Figure A.2: Entities for users, including students, faculty, and coaches, and the teams they belong to.

A.3 Mission Profile Entities

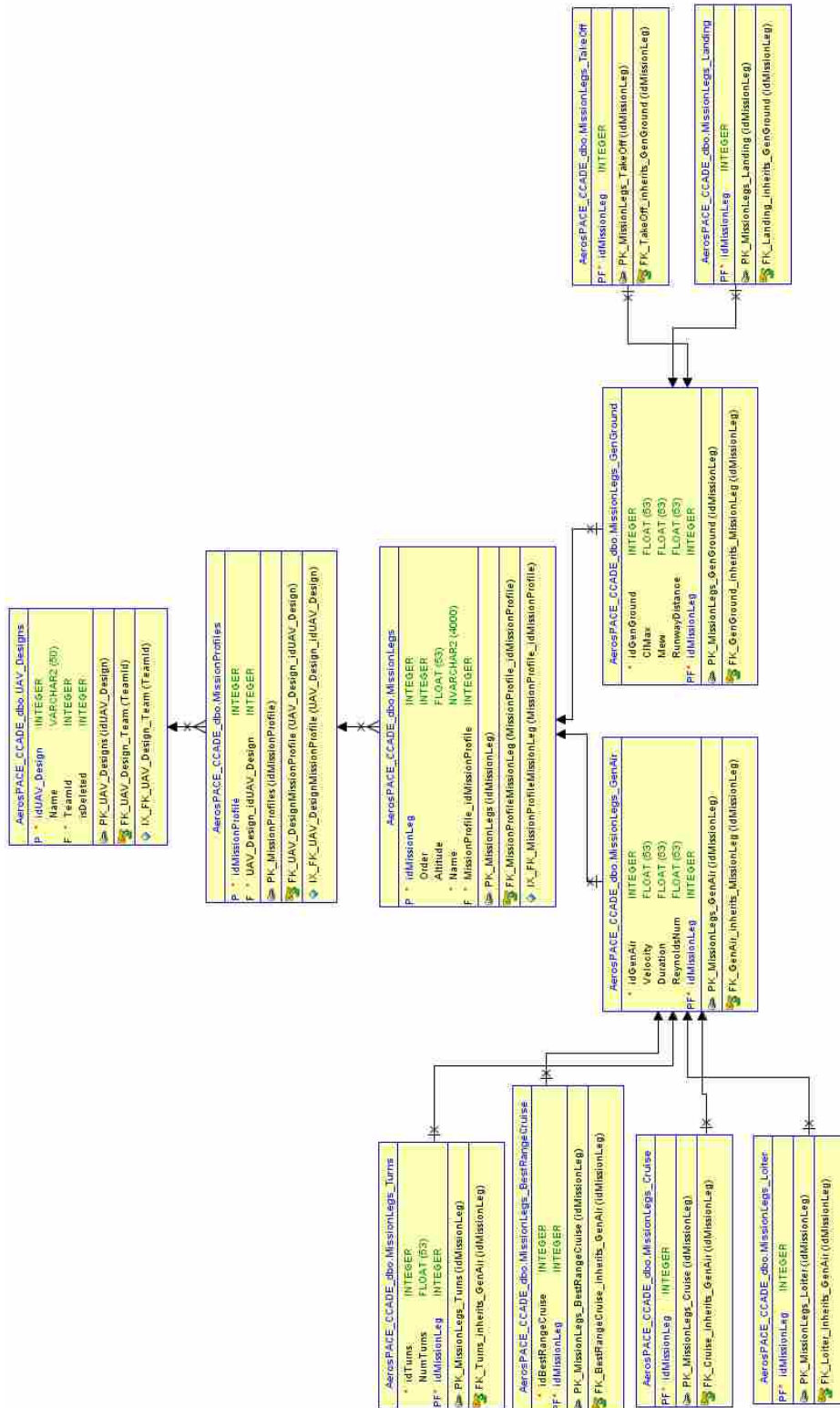


Figure A.3: Entities which define the mission profile. All mission leg types inherit from the entity MissionLeg to enable polymorphism.

A.4 Sizing Entities

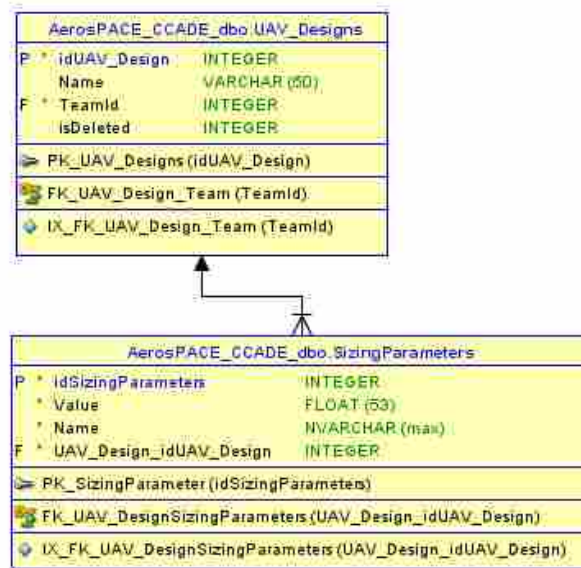


Figure A.4: Sizing parameters are stored in under a common entity set using a name and value.

A.5 Geometry Entities

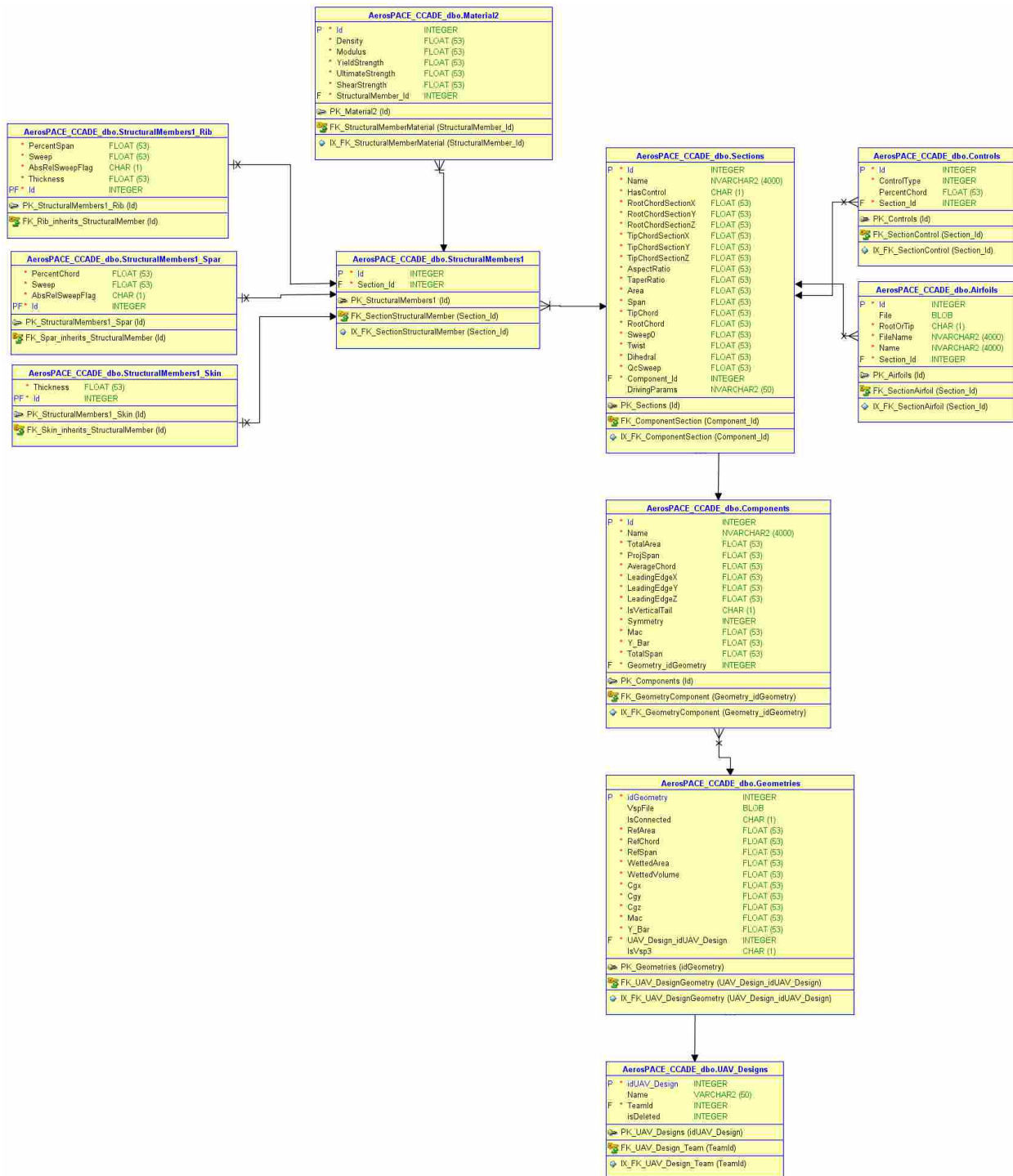


Figure A.5: Entities which completely define the geometry for the UAV. Geometry is broken down into Components, Sections, Airfoils and Controls.

A.6 Analysis Storage Entities

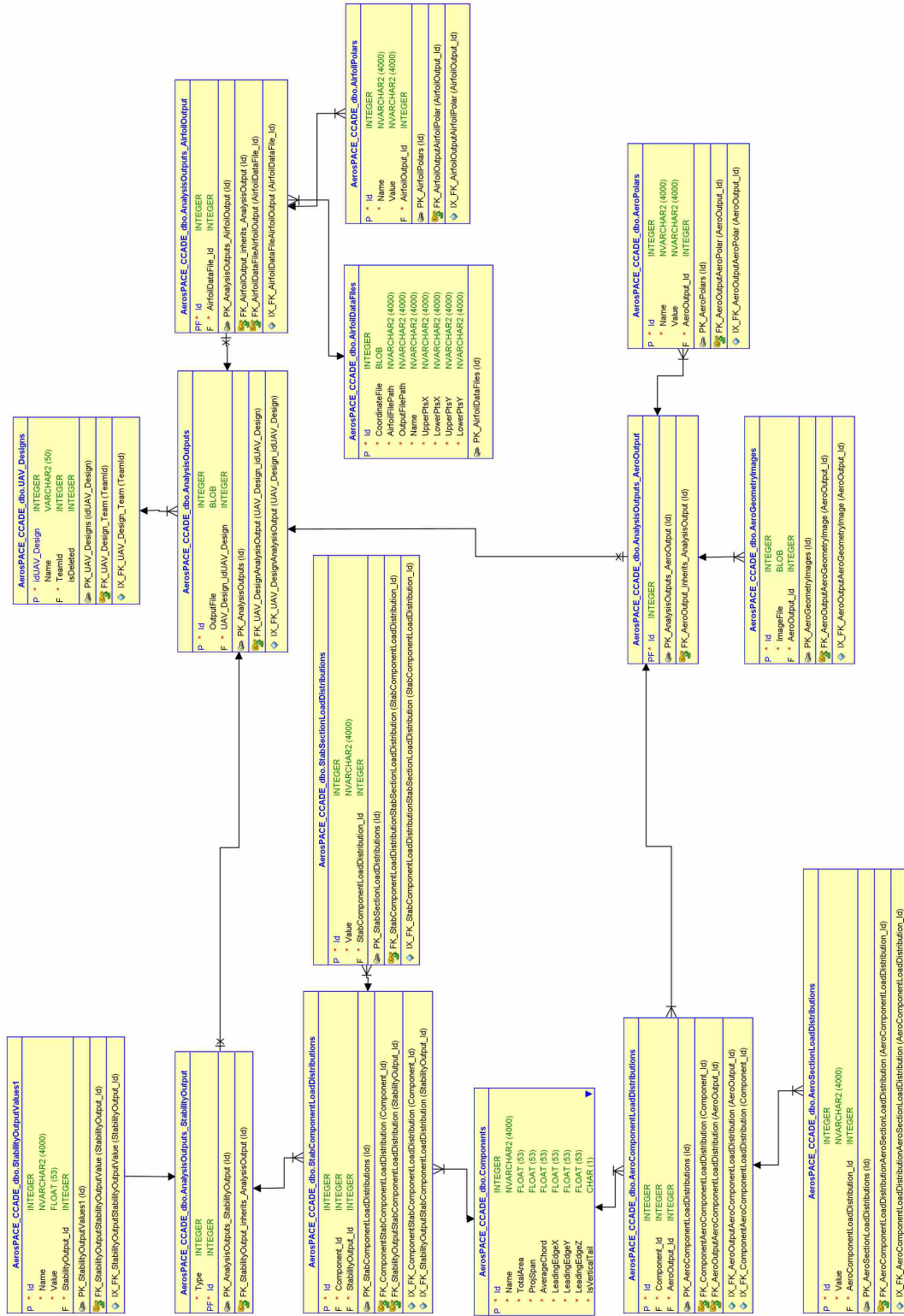


Figure A.6: All analysis storage entities inherit from AnalysisOutputs. The disciplines defined here include Airfoil Selection, Stability and Control, and Aerodynamics.

A.7 Propulsion Entities

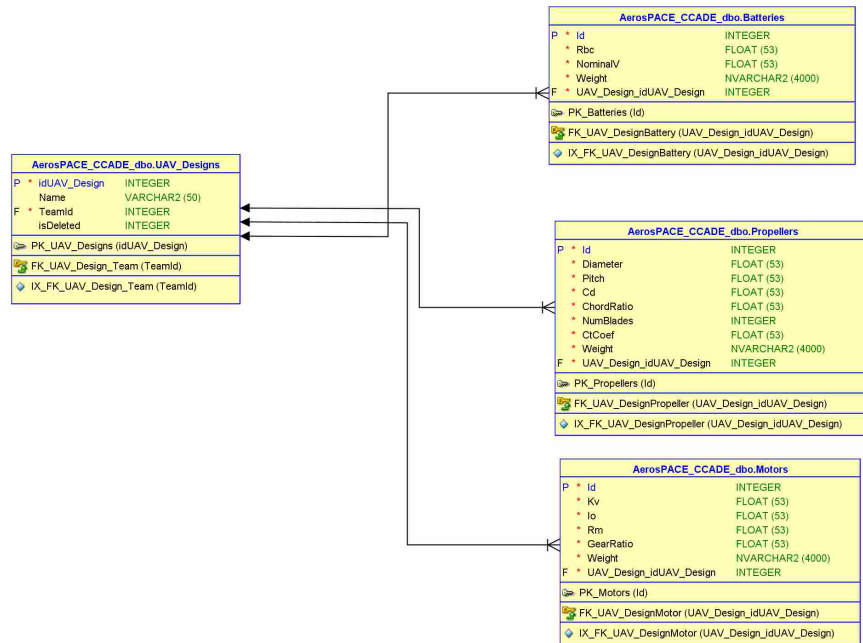


Figure A.7: Entities for the propulsion components including Motor, Battery, and Propeller.

A.8 Structures Entities

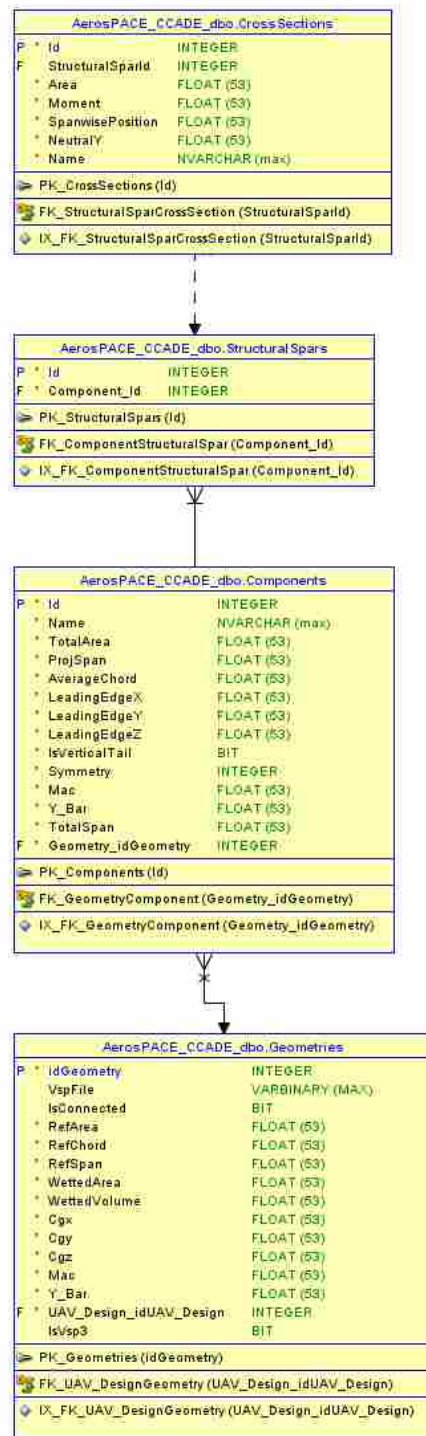


Figure A.8: Entities for storing spar design data and output. Spars are defined by their individual cross-sections.

APPENDIX B. CCADE USER MANUAL

AerosPACE CCADE

Version 0.7beta

User Manual

Updated 5 March 2015

Contents

Introduction	1
Summary	1
Background	1
Getting Started.....	2
Installation Instructions	2
Registration	2
Creating a Design	3
CCADE Design Workflow	5
Mission Requirements	7
Initial Sizing and Constraints	9
Geometry Definition	11
Airfoil Selection	13
Independent Analyses.....	16
Aerodynamics	16
Stability and Control	18
Propulsion	21
Structures.....	25
Conclusion.....	28

Introduction

Summary

AerosPACE CCADE (Collaborative Conceptual Aircraft Design Environment) is a novel collaborative design tool for the generation and evaluation of small-scale electric-powered UAV concepts in a multi-university, multi-disciplinary setting. Developed at BYU, this integrated design and optimization software enables the immersion of team members from different universities in an environment which stores design information and analysis results in a central database and automates portions of the design and optimization process. Analysis codes for initial sizing, aerodynamics, propulsion, stability and control, and structures are included. CCADE reduces the time required to analyze concepts, allowing students to increase the volume of concepts they consider, increase the quality of designs produced, and decrease lead time to preliminary and detailed design.

This User's Manual gives instructions on how to use the software, as well as a detailed explanation for all of the inputs and outputs that may be obtained.

Background

AerosPACE is a multi-disciplinary, multi-university collaborative capstone program bringing together stakeholders from industry, academia and government to build core competencies for the next generation of aerospace innovators in a sociotechnical, collaborative environment founded in the learning sciences. Its vision is to motivate students to enter the aerospace profession and fill gaps in student competencies using latest research from the NSF I/UCRC Center for e-design and industry. Over the course of two semesters, students from several universities, teamed with their geographically dispersed peers, are tasked with designing, building, testing, and flying a UAV that accomplishes a certain mission defined in a Request for Proposals (RFP).

To address these problems, a collaborative conceptual aircraft design environment (CCADE) has been developed which establishes a conceptual design analysis workflow, steps students through the first iteration, offers optimization methods, and connects students and their designs through a cloud-based collaborative design environment. The objectives of CCADE are to integrate all the low-fidelity analysis software in a predefined workflow, one that is flexible enough to handle a variety of configurations and allow full mobility of the students through the design space, and facilitate collaboration and sharing of information through storage of analysis results and design decisions in a centralized database server. CCADE aims to significantly reduce the time necessary to evaluate each design, making it easy to move on to preliminary and detailed design much quicker. Moreover, CCADE is designed to facilitate data sharing and increased interaction between geographically dispersed team members. By making the analysis results immediately available to all team members, an online round-table can be much more interactive and all members encouraged to participate.

Getting Started

Installation Instructions

You must first download the latest version of CCADE from the AerosPACE CorpU website. Log in to access the site, then access the [“Integration” page](#). Under “Files” on the right-hand side of the page you should see each version of CCADE posted, the most recent listed first. Simply click on the link to begin downloading the package.

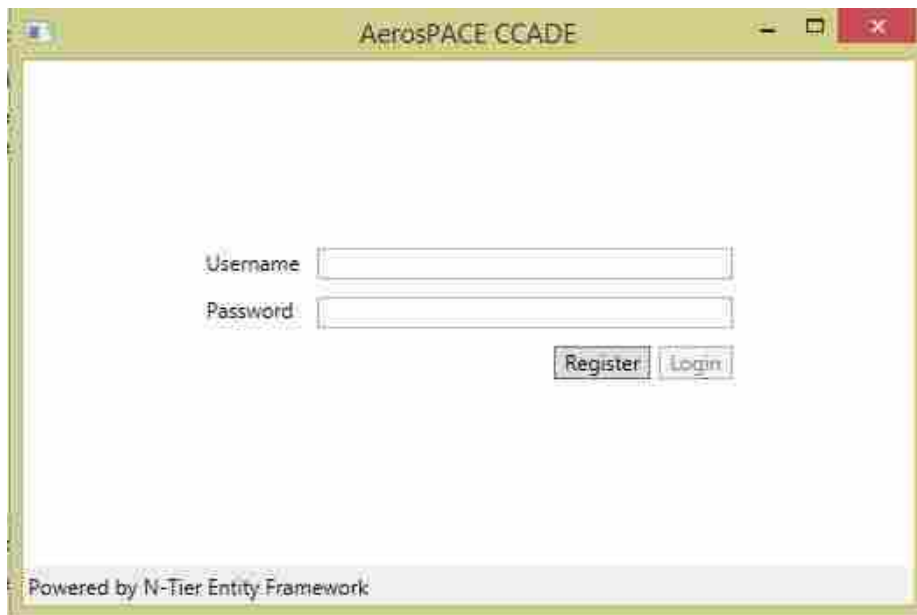
To install CCADE, all that is needed is to extract the package to the directory you wish to run from. This can be any directory that does not include spaces. Extracting to “C:\” is recommended.

Important: In order to avoid any issues executing analysis software from CCADE, you must save in a directory path that contains no spaces.

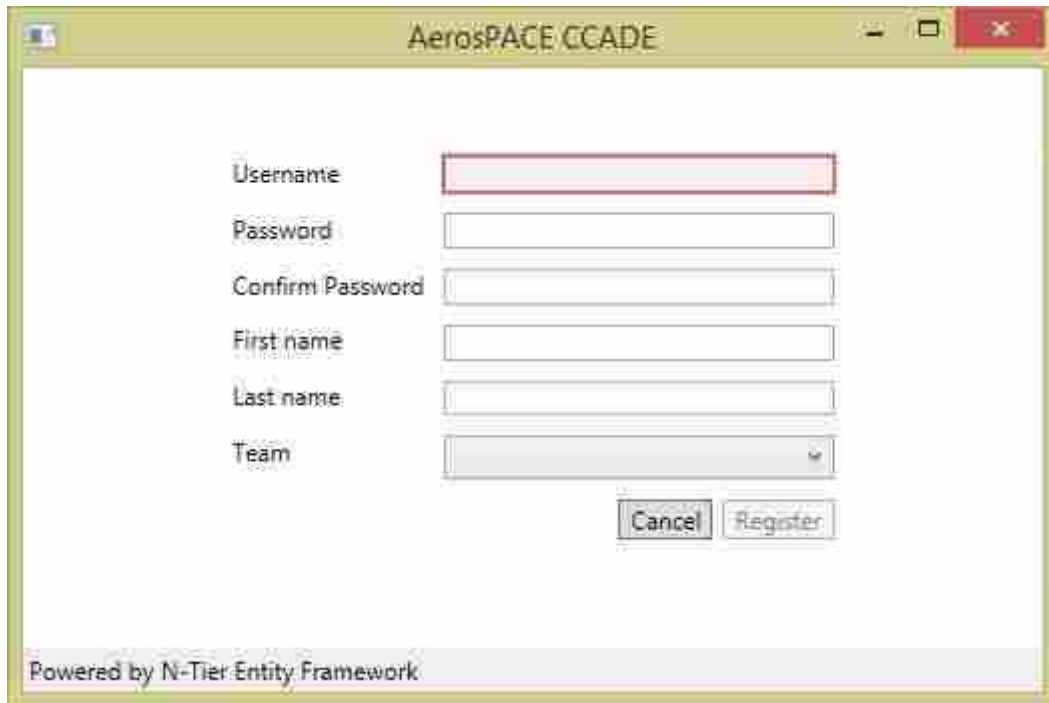
To open CCADE, enter the folder you extracted the contents of the zip file to, and double-click “Aerospace.CCADE.Client.Domain.AerospaceCCADE.exe”. By default, your explorer window may hide extensions on files. Make sure you look at the “Type” of the file and confirm that it is an “Application” and not a “CONFIG File”. Otherwise you may unwittingly try to open “Aerospace.CCADE.Client.Domain.AerospaceCCADE.exe.config”.

Registration

After you open the program, you should be greeted by the login page:



If this is the first time you are using CCADE, you will need to register a username and password and indicate which team you are on. Click on the “Register” button to access the registration form:



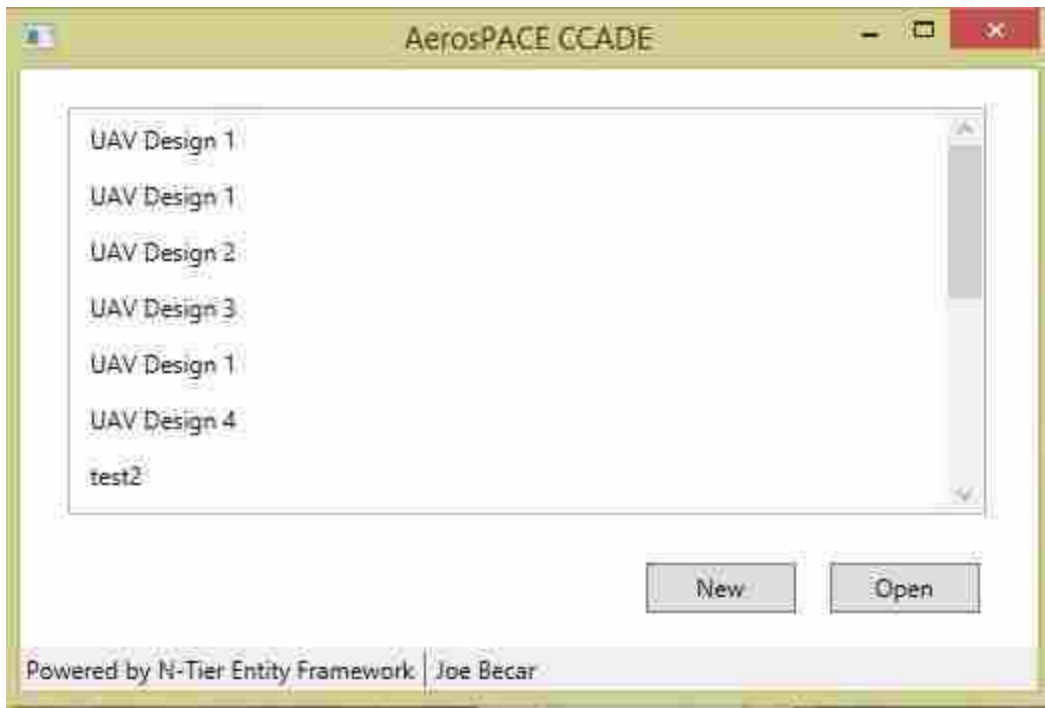
The screenshot shows a registration window titled "AerosPACE CCADE". It features a form with the following fields: Username, Password, Confirm Password, First name, Last name, and Team (a drop-down menu). Below the form are "Cancel" and "Register" buttons. At the bottom left of the window, it says "Powered by N-Tier Entity Framework".

Select a username and a password, and fill in your name (to identify yourself to other team members). Finally, select your team’s name from the drop-down box. Selecting a team lets you access all the designs created by that team. Once you register, you will automatically be logged in. If you have already registered, simply enter your username and password at the login page and click “Login”.

Creating a Design

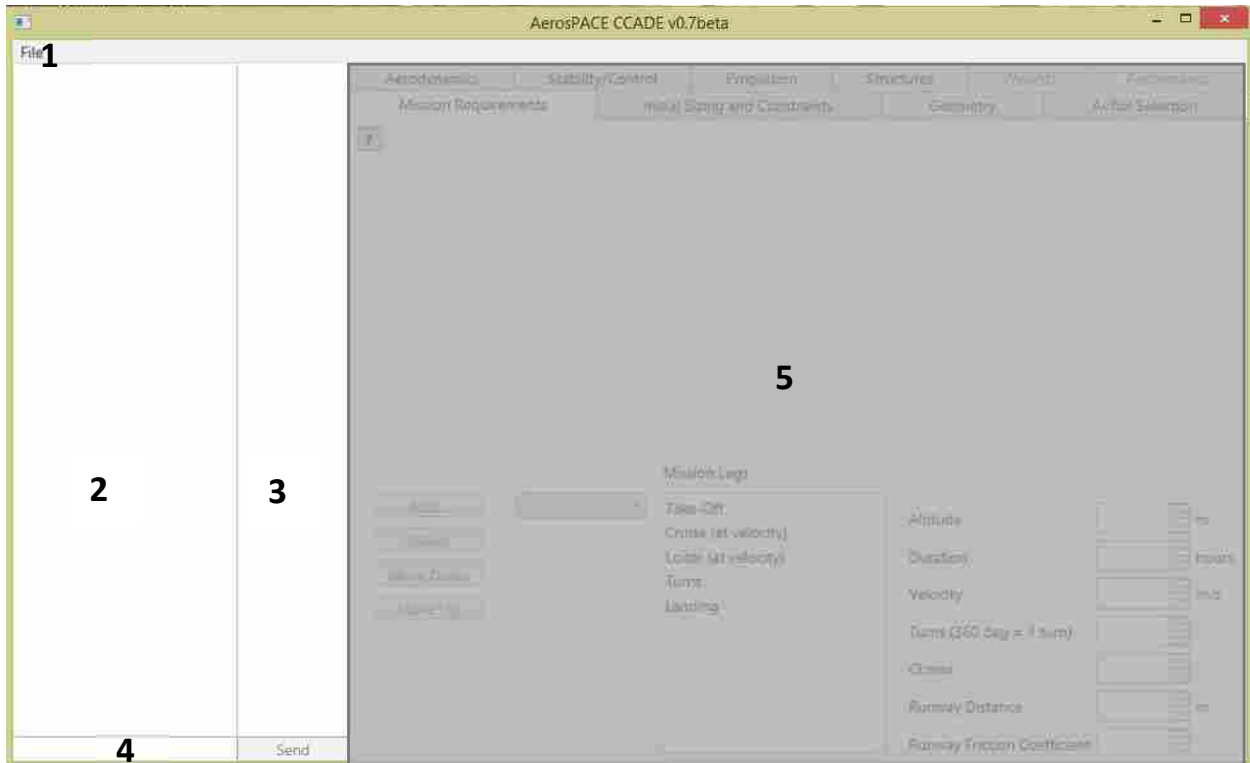
The most basic unit in CCADE is the UAV Design. No information is passed between designs, so you can make drastically different concepts and analyze them all in CCADE. Simply create a new design for each concept and run similar analyses to compare and contrast the performance of each. If this is the first time your team is using CCADE, click on the “New” button. The design will be created with a default name of “UAV Design #” where the number is the total number of designs created by your team so far. You can rename a design by right-clicking on the design name, and selecting “Rename” in the context menu.

If your team has already created some designs, the dialog may look something like below. Simply select which design you wish to open and click “Open”. You are now ready to start doing some real design work!



CCADE Design Workflow

Once you have selected a design to work on, CCADE will open the Main Design Window. A brief dissection of general elements is given below:

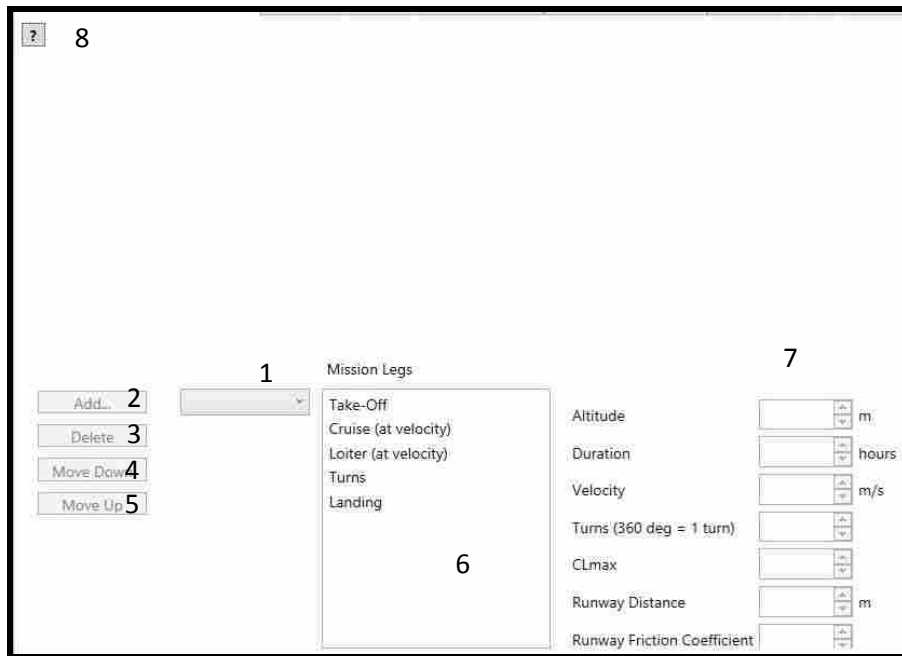


1. File menu, contains the following options:
 - a. Close Design – this will close the current design you are on, and bring you back to the design selection window. Here you can quickly and easily switch between different designs.
 - b. Logout – this will close the design and bring you back to the login window where you will be asked to enter you username and password again.
2. Chat area – your team’s chat messages will be shown here.
3. Chat members box – all active users on your team are shown here
4. Chat box – enter the message you wish to send in this box, and click “Send” to broadcast it to the team
5. Main Design area – here is where the bulk of the design work is done. Each tab represents a stage of the conceptual design or a discipline of analysis to be completed on the design.

Most data you enter in CCADE is saved automatically, so you should experience no setbacks if you need to close the window at any point. Simply “x” out of the application when you are done.

All data is queried directly from the database before it is used. This ensures that your work immediately reflects changes made by team members involved in the same design. Although more work is being done to display those changes immediately to you if you are working in the same tab in CCADE, this will not prevent the changes from actually affecting the analysis. The work-around currently to see the changes made by your team members is to move into a different tab, and then return to the one you are both working on. In this way, the graphical interface will access the data stored in the database once more and update the display for you.

Mission Requirements



In every tab, there is a help button (8), which when clicked will open your default browser to a lecture on the topic of the current tab. In this way, you can easily access information which you may have forgotten or get additional help on the work of the current tab.

In the Mission Requirements tab, you will specify a mission profile to use during the course of the analysis. They are used to derive the constraints for power and wing loading during sizing. By selecting a type of mission leg from the combo-box (1), and then clicking Add (2), you can add a mission leg to the profile, shown in (6). If you need to remove a particular leg from the mission profile, select it in (6) and then click Delete (3). Although the order of the mission legs has no bearing on the analysis conducted, you may organize it as you like by selecting a leg and moving it up (4) or down (5). Each time you select a leg in (6), its properties will be shown in (7). Each mission leg has an Altitude associated with it. This altitude is distance above sea level, and is used to derive the air density to be used at point. There are six types of “mission legs” available to use (see Table 1), and each have certain parameters associated with them. These parameters must be input before other analyses can be completed.

Table 1: Types of mission legs and their required input parameters.

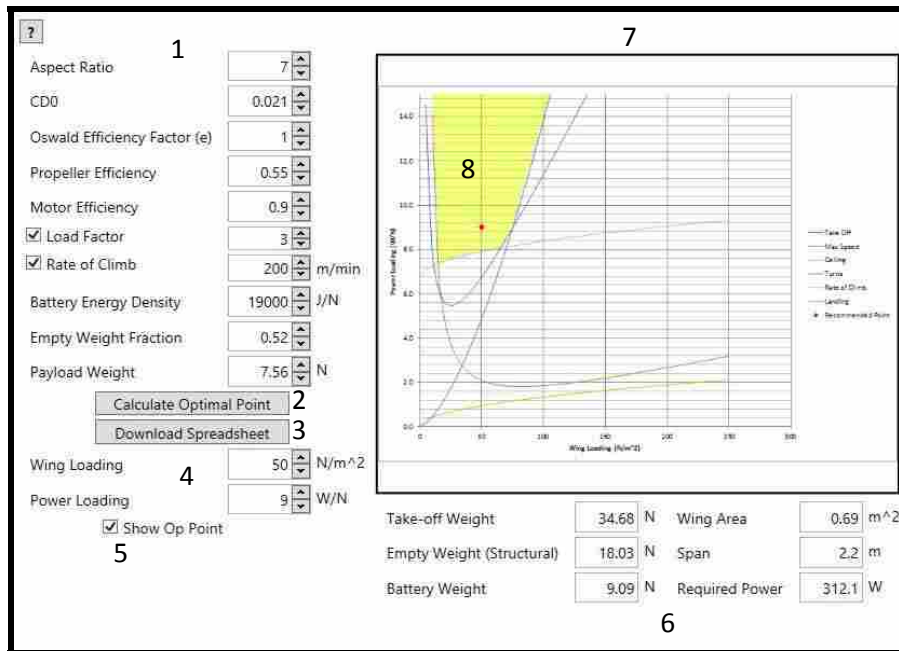
Mission Leg Type	Description	Required Input Parameters
Take-Off	Acceleration to lift-off via a runway	Altitude, CLmax, Runway Distance, Runway Friction Coefficient
Cruise (at velocity)	Steady flight	Altitude, Duration, Velocity
Loiter (at velocity)	Steady flight, typically at a lower velocity than cruise	Altitude, Duration, Velocity
Best Range Cruise	Optimizing velocity to use the least battery power during the	Altitude, Duration

Mission Leg Type	Description	Required Input Parameters
	duration, or maximize the range flown	
Turns	Turning with load-factor determined by velocity	Altitude, Velocity, Turns
Landing	Final landing of the vehicle	Altitude, CLmax, Runway Distance, Runway Friction Coefficient

CCADE only allows take-off from a runway. A guess of the maximum lift coefficient must be made at this point, and may be refined later. The number of turns input for the Turns leg, as indicated, shows that 1 turn is equal to 360 degrees. Thus, if the aircraft were required to make ten 180-degree turns during the mission, one would input 5 as the number of turns.

Be sure to include one and only one take-off and landing for each mission profile, and at least one flight leg (Cruise, Loiter, or Best-Range Cruise). Once you have entered all of the required input parameters for each mission leg in your profile, you may move on to the next section.

Initial Sizing and Constraints



The initial sizing and constraints tab interfaces with an Excel spreadsheet created by Dr. John Sullivan of Purdue University. The inputs consist of the values on the left in (1), and the outputs are listed on the right in (6) and the plotted constraints in (7). A description of each input and output is given in Table 1 and Table 2. Contained in (4) are the wing loading and power loading, which define the design point, shown as a red diamond in the constraint chart plotted in (7). This design point, or operating (op) point, can be toggled on and off using the checkbox in (5).

Wing loading is a measure of the required lift to fly relative to the size of the airplane. It is calculated by dividing the take-off weight by the wing area. Power loading is a measure of the motor power available to the required lift. It is calculated by dividing the required power by the take-off weight.

In this analysis, we use the constraint equations to limit the acceptable wing loading values. Once you have filled out the first 7¹ inputs in (1), the chart of constraints will begin plotting. Any time you change an input, the chart will update. The design space is the area above every constraint curve, highlighted as (8)². You can easily calculate an “optimum” point for the wing loading/power loading by clicking on (2). This will calculate the minimum power loading/wing loading combination and then apply a 5% safety factor to the power loading. You will probably want to select a point that provides a better margin of safety, or perhaps adjust the wing loading to achieve the scale of UAV you are looking for.

¹ Unless load factor and rate of climb are unchecked, as they are optional constraints. You might only fill out a minimum of 5 inputs.

² This highlighting is not provided in CCADE, but just used to illustrate the design space for this manual.

The results in (6) show the results from a weight sizing spreadsheet also developed by Dr. Sullivan. This spreadsheet takes as input the wing loading and power loading from the constraint sizing, and the mission profile data to get flight times. It calculates the total power needed to fly the mission, and from the battery energy density gets the required battery weight. The required power is the maximum of all specified flight legs. The span is simply the square root of the wing area multiplied by the aspect ratio.

If you would like to download the actual spreadsheet where the results are being calculated, you may click on (3). This spreadsheet will contain both constraint sizing spreadsheets and weight sizing spreadsheets.

Table 2: Constraint and Weight Sizing parameters. Parameters required for constraint sizing are also required for weight sizing. Parameters only required by weight sizing are boxed at the end of this list.

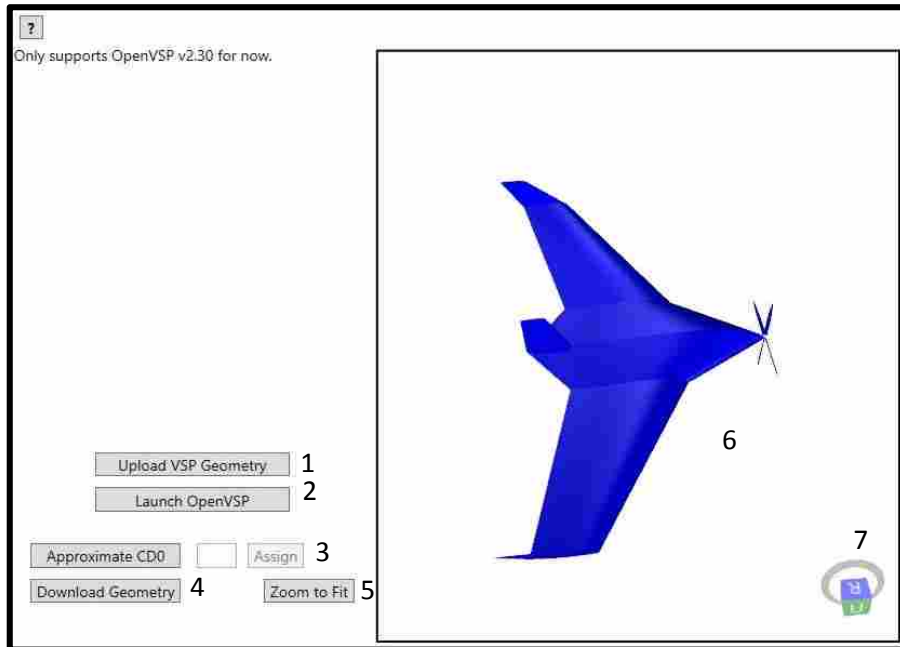
Input Parameter	Description
Aspect Ratio (AR)	Span (b) to constant chord (c) ratio, $\frac{b}{c}$ or more generally, using wing area (S), $\frac{b^2}{S}$
CDO	Parasite drag, mainly due to skin friction. This value can be approximated with low-fidelity in the Geometry tab. It may be updated when a better approximation exists.
Oswald Efficiency (e)	Represents increase in drag-due-to-lift (induced drag, C_{Di}), as compared to an ideal wing. It is defined using the coefficient of lift (C_L) as $e = \frac{C_L^2}{\pi C_{Di} AR}$
Propeller Efficiency	The efficiency of the propeller, defined as the ratio between power out and power in.
Motor Efficiency	The efficiency of the motor, defined as the ratio between power out and power in.
Load Factor	Ratio of lift-to-weight. A higher load factor comes from more demanding maneuvers.
Rate of Climb	The rate of change in altitude, proportional to the excess power available.
Battery Energy Density	The amount of energy contained per unit weight of the battery. By multiplying the mAh (milli-Amp hours) and voltage of the battery, you can easily get the joules contained, then simply divide by the weight.
Empty Weight Fraction	The fraction of the empty weight (without the battery or payload) to the total weight. This can be guessed from historical data on similar UAVs, or iterated on.
Payload Weight	The weight of your payload, which is generally fixed.

Table 3: Weight Sizing outputs and their descriptions.

Output Parameter	Description
Take-off Weight	Along with Wing Area, this is one of the most important outputs. This will be used to calculate required lift for later analyses.
Empty Weight	The weight minus battery and payload.
Battery Weight	The battery weight required to complete the mission, based off flight time and battery energy density.
Wing Area	The planform area of your aircraft necessary to complete the mission, calculated with the wing loading and the take-off weight.
Span	The span is simply the square root of the wing area multiplied by the aspect ratio. ³
Required Power	The maximum power required of all specified flight legs.

³ See the definition of Aspect Ratio in Table 1.

Geometry Definition



In the geometry tab, you will be prompted to upload an OpenVSP file you have previously created. As of the time of this writing, only OpenVSP Version 2.3 file types are supported by this import. Work is being done to implement version 3.x cross-compatibility. Simply click on (1) and select your OpenVSP file. The file will be read and the geometry imported to the database, and an STL file generated for viewing in the geometry viewport (6). In this viewport, you and your teammates may manipulate the view by:

- Rotating – Hold Right-Mouse button and drag
- Panning – Hold Middle-Mouse button and drag
- Zooming – Scroll with the Middle-Mouse wheel

In order to quickly orient the view normal to one of the globally-defined axes, click on the corresponding face of the box in the viewport (7). To quickly zoom in so that the model fits nicely within the viewport, click on “Zoom to Fit” (5).

If you have not yet defined an OpenVSP model, you may click on (2) to launch OpenVSP 2.3. Once you are done, return to CCADe and use (1) to upload the file you created. Once the geometry has been uploaded properly with no error messages, you may click “Approximate CD0” by (3) to estimate the parasite drag based off of the ratio of wetted area (S_{wet}) to planform area (S_{ref}). This is based off a flat-plate model for skin friction (f_s) using the largest Reynolds number (Re) in your mission profile:

$$f_s = \frac{0.455}{\log(Re)^{2.58}}$$

$$C_{D0} = f_S \frac{S_{wet}}{S_{ref}}$$

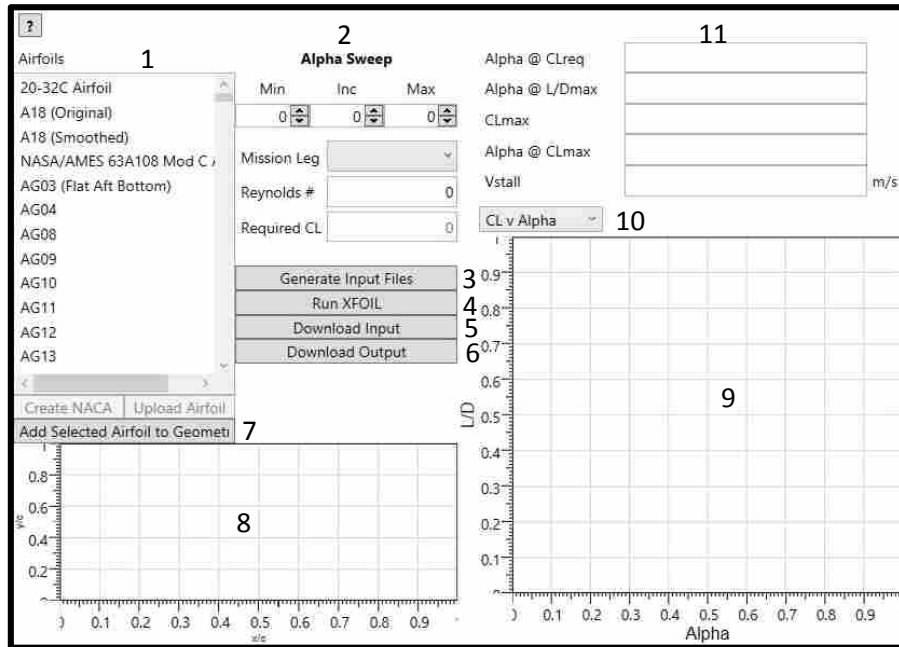
Simply click “Assign” to assign the value in the textbox to the CD0 in the constraints tab, which will be used in future analyses.

There is no need to define airfoils or control surfaces in OpenVSP, as that will be done later in CCADE. At this point there is no way to import such assignments should they exist in the VSP geometry, so you will need to assign airfoils and control surfaces in CCADE regardless of whether they exist already.

If you would like to download the OpenVSP file from the database to use later, you may click (4) and choose a location on your machine to save the file to. Once you have completed the geometry upload, you can move on to Airfoil Selection.

One assumption of the geometry in CCADE is that all configurations are symmetric. If yours is not, you may encounter errors in later analyses.

Airfoil Selection



In the Airfoil Selection tab, you have the opportunity to analyze a number airfoils with XFOIL and select ones that will work well for your aircraft. All of the airfoils contained in the UIUC Airfoil Coordinates Database⁴ are listed in (1). As you select an airfoil in the list, its profile is previewed for you in the chart in (8). This might be especially helpful, for example, if you are trying to identify airfoils with reflex to use for a flying wing concept. It might also help you identify airfoils with more or less thickness for structural components, payload bays, or other components.

Once you have identified an airfoil which you would like to analyze, you can select a range of angles of attack (AOA) to analyze over. You select a minimum AOA in the first field under (2), which must be one that can be easily converged on in XFOIL, recommended low negative AOA (-4 to 0 degrees is usually an acceptable range). If you do not select a reasonable angle of attack, the analysis may not be able to converge on the first run, and will skew the remainder of the angles of attack. The second field is the increment which is added to the minimum value until it reaches the maximum value in the third field. The other inputs for the analysis are the Reynolds number and coefficient of lift. These are both calculated from the flight mission profile data, sizing data, and the stored geometry, based on the mission leg you select from the combo-box. If you would like to adjust the Reynolds number manually, you may change it in the text box. Normally you will optimize the airfoil for the longest duration or otherwise most critical mission leg, and then ensure that the performance is acceptable for take-off, for example.

Once you have the inputs defined, you may click (3) to generate the necessary input files for XFOIL to run. Then click (4) to trigger execution of XFOIL. It will run the analysis at the defined angles of

⁴ Website - http://m-selig.ae.illinois.edu/ads/coord_database.html

attack, and then read in the output. Buttons (5) and (6) allow you to download the input generated by (3) and the output generated by XFOIL, respectively. The main outputs are given in a series of charts shown in (9). These include:

- Coefficient of Lift (C_L) versus AOA
- Coefficient of Drag (C_D) versus AOA
- Coefficient of Moment (C_M) versus AOA
- Lift versus Drag
- Lift-to-drag ratio (L/D) versus AOA
- C_L versus laminar-to-turbulent transition X (this is the x value where the flow transitions from laminar to turbulent, which greatly increases drag and gets closer to separation).⁵

These outputs are raw data from XFOIL. The other outputs shown by CCADE in (11) are calculated from this raw data, which are intended to help you make good design decisions. These outputs are described in Table 4.

Table 4: Outputs calculated by CCADE from raw XFOIL output.

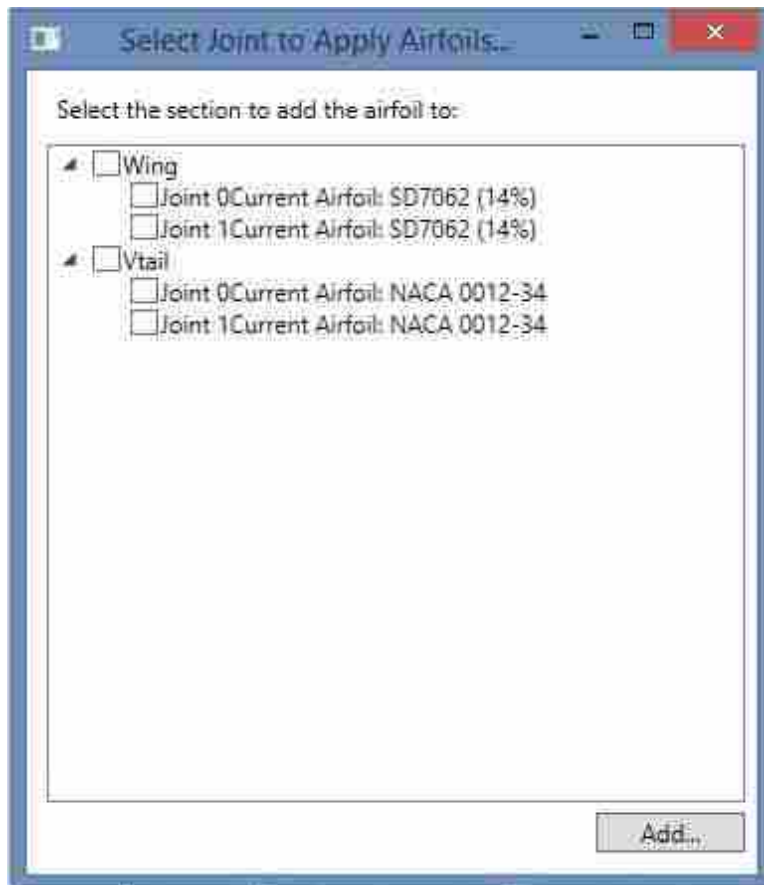
Output	Description
AOA at the required C_L	The AOA, if your UAV were an airfoil flying at the specified mission leg. You can think of it as the inclination of your airfoil which will affect the AOA of your 3D UAV at the mission leg.
AOA at the maximum L/D	The AOA, where you have the highest efficiency of your airfoil. You should try to get this value as close to the AOA @ C_{Lreq} as possible.
C_{Lmax}	The coefficient of lift just before stall. This is the maximum lift you can get out of the airfoil.
AOA at C_{Lmax}	The AOA just before stall.
Stall velocity	The velocity just before stall. Reducing velocity beyond this will cause the airfoil to stall. This is calculated using the take-off weight from the sizing tab.

Once you have decided on an airfoil to use in your UAV, make sure it is selected in (1), and then click (7). This will bring up a separate window shown below. This window shows a tree listing as a hierarchy:

- Each lifting component
 - Each joint between wing sections in the lifting component

The components are listed by name. The joints are labeled from the root span, 0, to the tip of the wing. The current airfoil, if you have previously applied an airfoil to the joint, is shown next to the joint number. Simply check all the joints you wish to apply the airfoil cross-section to. Checking the box next to the component will automatically select all the joints contained in that component. When you are done, click "Add". Once all wing joints have been assigned an airfoil, then you may move on to the aerodynamics tab. You CANNOT move on before ALL wing joints have an airfoil assigned.

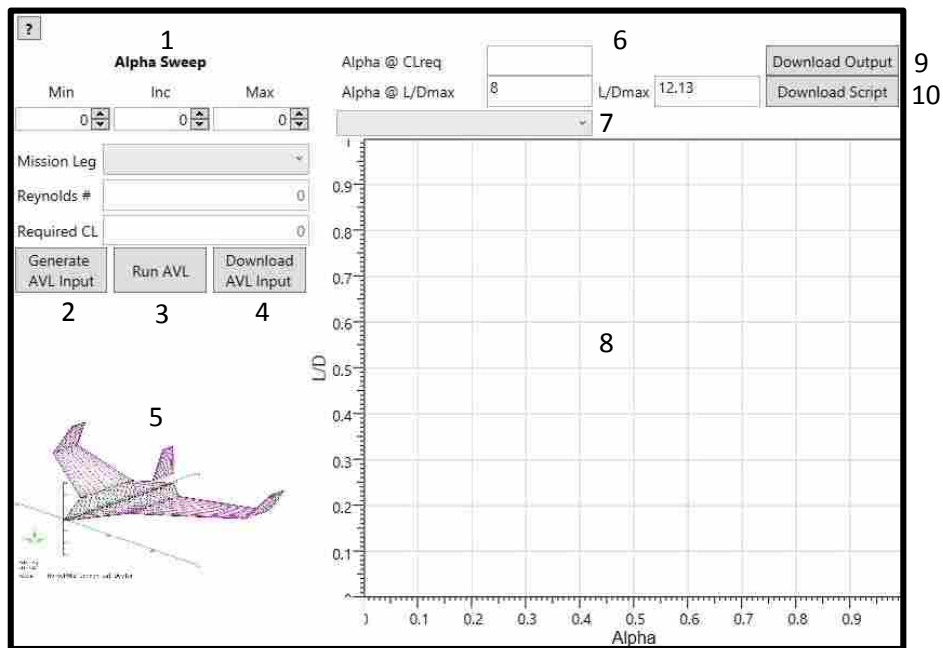
⁵ The green plot represents the transition to turbulence on the upper surface (suction side) of the airfoil. The red plot shows the transition of the lower surface (pressure side) of the airfoil.



Independent Analyses

The previous analyses and design inputs, Mission Requirements, Initial Sizing, Geometry Selection, and Airfoil Selection, are required for all the subsequent analyses. Do not attempt to run other analyses without first completing these. The aerodynamic, stability and control, propulsion, and structures analyses are essentially independent in the sense that they can be completed in any order. The exception to this is the structures module, which requires a lift distribution to calculate the forces on the wing. Focus on getting at least one good aerodynamic analysis complete before working simultaneously on the other disciplines of analysis. Then they might all be worked (in addition to the previous analyses) independently.

Aerodynamics



The Aerodynamics tab is used with the airfoil data you just defined, in conjunction with the geometry you uploaded, to analyze the 3D wing geometry. There will be differences in the airfoil results to even a straight wing with a single airfoil section because of the finite length of the wing. The inputs (1) are identical to those required for the airfoil selection analysis, except that the airfoil sections have been previously defined. Please refer to the previous section for an explanation of these inputs. One difference is that the Reynolds number is no longer variable, but defined by the mission leg selected.

Once the inputs are defined, click on (2) to generate the required input AVL file and script. You can download the AVL input file (which contains a definition of the panel geometry) by clicking on (4), and download the AVL script file (which is a log of the AVL commands required to run the analysis CCADE runs for you) by clicking on (10). This will also trigger an update in the visualization of panel geometry in (5). This is an image of the degraded geometry defined automatically for you per your

OpenVSP geometry. Check this to make sure it reflects your actual geometry before clicking on (3) to begin the AVL execution to run the analysis. You will be able to see the AVL progress on your screen. Once it completes, the raw results are displayed in charts in (8), and CCADE-calculated interpretations of the results in (6). The raw data plots are the same as in the airfoil selection tab, with the exception of the transition to turbulence plot. They are repeated here for completeness:

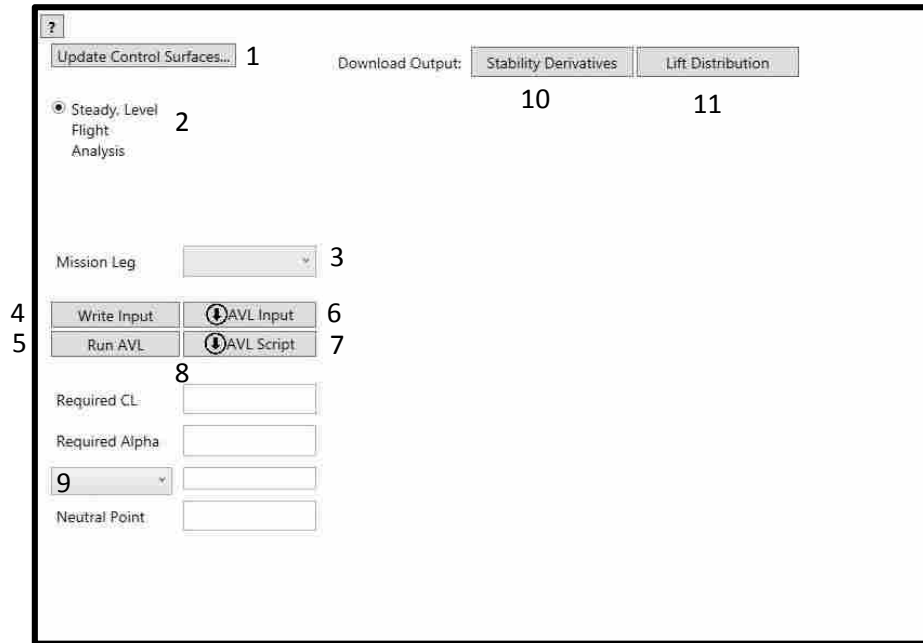
- Coefficient of Lift (C_L) versus AOA
- Coefficient of Drag (C_D) versus AOA
- Coefficient of Moment (C_M) versus AOA
- Lift versus Drag
- Lift-to-drag ratio (L/D) versus AOA

You can switch the plots by using the combo-box (7). The outputs calculated by CCADE are also a subset of those given in the airfoil selection tab. This includes:

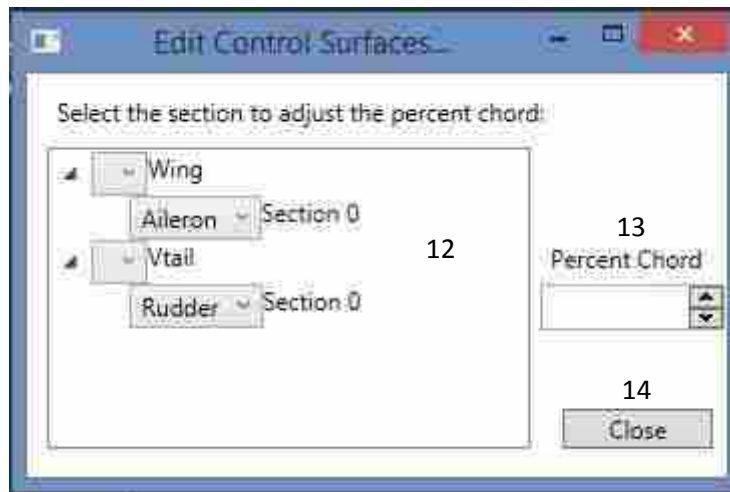
- The AOA at the required C_L at the chosen mission leg
- The AOA at the maximum L/D
- The value of maximum L/D

You may download all the raw data from AVL by clicking on (9).

Stability and Control



The stability and control tab uses AVL to analyze the ability of your aircraft to recover from perturbations in the flow, and the ability to steer your aircraft where you want it to go. First, you will need to add a definition for control surfaces by clicking on (1). This will bring up a separate window shown below.



In this “Edit Control Surfaces...” window, there is a list of the lifting components of your aircraft (ones defined as a wing in OpenVSP) in (12). Under each component are listed the individual sections that make up the wing, numbered from the root section, 0, to the final section which includes the wing tip. The combo-box next to each section is used to define the type of control surface to be applied to it. The various types are describe in Table 5, showing normal methods of deflecting the control surface

(either up/positive or down/negative, which might be violated by AVL), whether symmetric control surfaces are moved in the same or opposite directions (symmetric or anti-symmetric, respectively), the target moment which the control attempts to zero, and popular locations for application of the control surface. You cannot assign a control surface to only a portion of a section. To adjust the span of the control surface, you must add an additional section in OpenVSP that correctly breaks up the wing. When you click on the section so it is highlighted in (12), the percent chord field becomes active (13) and you can modify the percent of the total wing chord that is allocated for the control surface. Once you are done assigning control surfaces, click (14) to close the window.

Table 5: The types of control surfaces offered by CCADE, with how their symmetry is represented in AVL, which moment the scripts direct it to target, and normal deflections and locations.

Control Surface Type	Normal Deflection	Symmetric/Antisymmetric	Target Moment	Location
Elevator	Negative	Symmetric	Pitching	Horizontal Tail
Flap	Positive/Negative	Symmetric	None – increases lift	Main wing
Aileron	Positive/Negative	Antisymmetric	Rolling	Main wing
Rudder	Positive/Negative	No symmetry	Yaw	Vertical tail
Flaperon	Positive/Negative	Symmetric/Antisymmetric	Rolling – increases lift	Main wing
Elevon	Positive/Negative	Symmetric/Antisymmetric	Pitching/Rolling	Main wing – flying wings
Ruddervator⁶	Positive/Negative	Symmetric/Antisymmetric	Pitching/Yaw	“V”-Tail

Once you have the control surfaces defined, you can move on to define the mission leg to perform the analysis at with the combo-box (3). Although there are more types of stability analyses planned for future versions of CCADE, the only one available currently in the radio-button list at (2) is a steady, level flight analysis. This analysis iterates on the angle of attack and deflections of control surfaces until the pitching, rolling, and yaw moments come within an acceptable tolerance of zero. Since the UAV is assumed level, and no side-slip is applied, the yaw moment automatically defaults to zero (assuming a symmetric body and lift distribution), so other methods must currently be applied to size the rudder if one exists.

Click on (4) to generate the required AVL panel geometry file and script to run the analysis. You may click on (6) to download the former and (7) for the latter after this has been done at least once. Then, click on (5) to begin running the analysis in AVL. You will see AVL running, iterating until the constraints are met. If they successfully converge, the reduced output will be shown in the fields below (8). These outputs were chosen because they are especially important to the designer when sizing control surfaces and assessing the stability of the aircraft, and are explained in Table. The raw AVL output containing aerodynamic data as well as the complete set of stability derivatives is available by clicking (10). By clicking (11), you can also download the spanwise lift distribution of the aircraft.

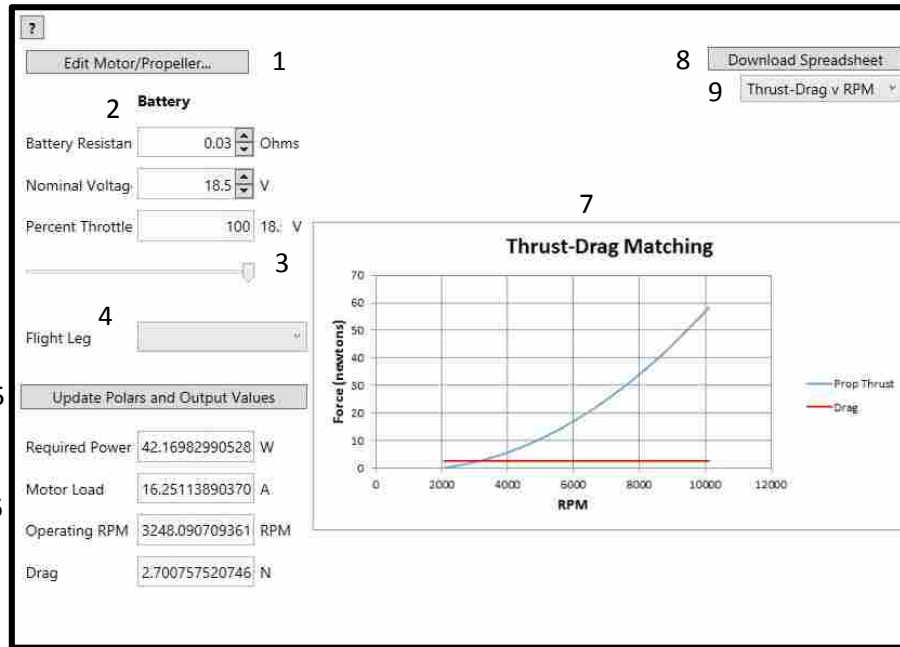
⁶ Not implemented in current version of CCADE, but planned for future versions.

Table 6: Stability and control outputs reduced from raw AVL data.

Output	Description
Required angle of attack	The angle of attack required to achieve the required lift coefficient, defined by the selected mission leg.
Control Surface Deflection	The required control surface deflection to achieve steady, level flight.
Neutral point	The position of the center of gravity where the aircraft is neutrally stable. To be positively stable (inherently return to steady, level flight after a perturbation), the center of gravity must be forward of this point.

By using the combo-box (9), you can alternate the display of control surface deflection between each control surface you have defined on your aircraft. They are named after the pattern: “<Control Type> Deflection.”

Propulsion

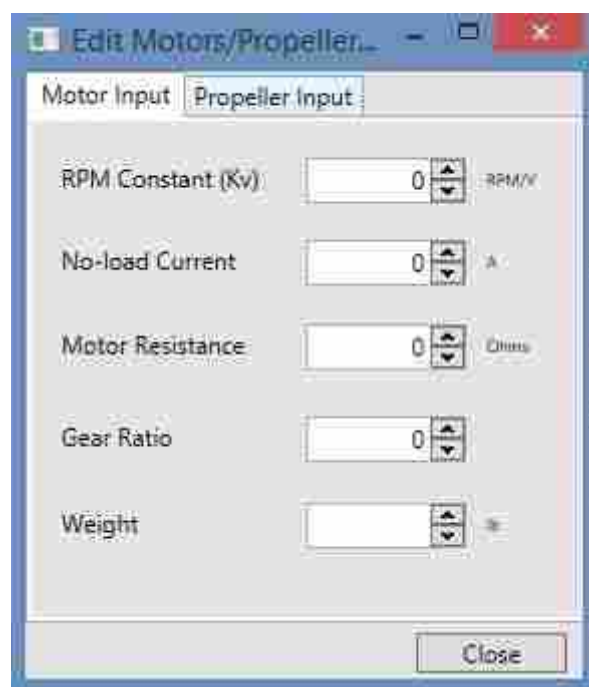


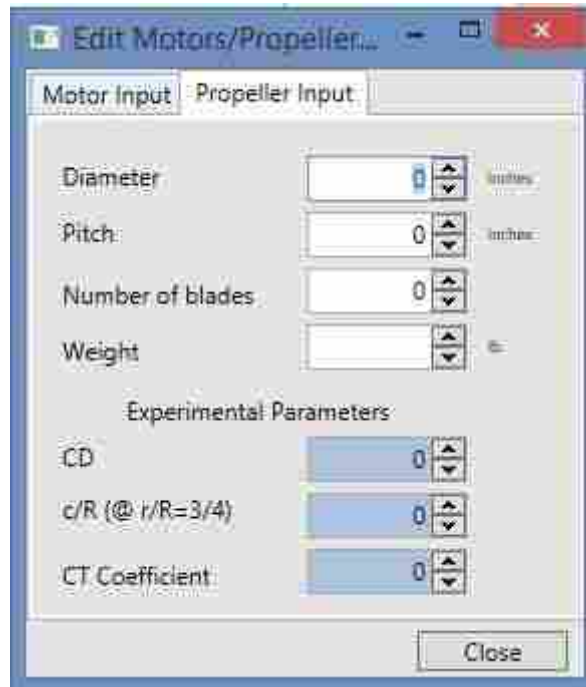
The Propulsion tab enables you to define motor, propeller, and battery properties and analyze the power system of the UAV. The first task to accomplish is to click on (1) which opens a new window (shown below) for you to begin to define the motor and propeller of your system. These values can come from manufacturer data on a motor and propeller you already have, or they can come from data you find on the internet on potential motors and propellers for the same scale aircraft as the current design concept. The window for inputting the motor and propeller data is simply a list of fields that must all be filled out for the analysis to run successfully. These inputs are described in Table 7.

Table 7: Description of motor and propeller inputs.

Motor Input	Description
RPM Constant (Kv)	The RPMs obtained per volt applied via the speed controller to the motor.
No-load Current	The current the motor draws when there is no propeller or other load attached. Additional current is drawn linearly with respect to the load placed on the motor.
Motor Resistance	The electrical resistance of the motor, which increases the required current draw.
Gear Ratio	The ratio of angular velocity input from the motor to angular velocity output to the driven load.
Weight	The weight of the motor
Propeller Input	Description
Diameter	The diameter of the area swept out by the propeller blades.

Pitch	The pitch or degree of twist in the propeller blade. Defined as the distance a propeller would move in one revolution if it were moving through a soft solid.
Number of Blades	The number of propeller blades, placed symmetrically about the center.
Weight	The weight of the propeller.
CD	The drag coefficient of the propellers. A good estimate is 0.03.
c/R, chord-to-radius ratio at $\frac{3}{4}$ radius	The chord width at the $\frac{3}{4}$ of the radius of the propeller blade. This can be measured on a propeller, or estimated based on the type of flight the propeller is designed for.
CT Coefficient	Thrust coefficient of the propeller. This is empirically derived; search for performance charts of the propeller you are researching.





Once the motor and propeller have been defined, you must define the battery properties back in the main design window (2). The key specifications are the battery resistance and voltage, which can be found in manufacturer specifications. The percent throttle will be adjusted during the course of the propulsion study to determine what you need to run the throttle at for, say, the main cruise condition. You can also set the percent throttle to 100% and ensure that your performance is sufficient to take-off. Use the slider (3) to quickly adjust the percent throttle. The actual voltage being applied will be shown to the right of the slider.

After you select a mission leg to analyze performance at, the required power and drag fields will be updated, which you can use to check that you have sufficient power to overcome drag (this is also shown in the charts. You can then click (5) to update the two remaining output values (6) and plots (7). The remaining two outputs are:

- Motor Load – The Amps that the motor pulls from the battery in order to drive the load.
- Operating RPM – The speed of the motor using a propeller-thrust-to-drag matching method to obtain the required advance ratio.

The charts available for viewing are described in Table 8:

Table 8: Types of plots available in the propulsion tab for assessing the power system and matching motor to propeller.

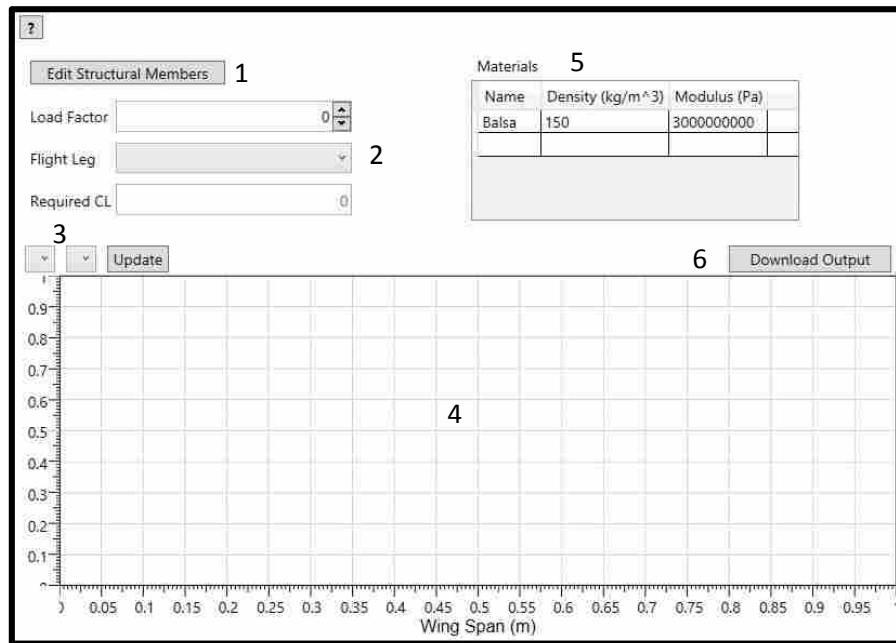
Plot Type	Description
Thrust & Drag v. RPM	Propeller thrust is plotted against UAV drag, so you are able to see at what RPM the propeller will produce enough thrust to overcome the drag.
Power v. RPM	Propeller power and motor power are plotted together, and the operating point (to overcome drag) shown. Since the power must

	always be equal, you can adjust the percent throttle until both curves meet at the operating point.
Efficiency v. RPM	The efficiency of both motor and propeller will peak at a certain RPMs (rarely the same RPM). Try to design so that the operating point is at an acceptable efficiency for both.
Power & Thrust v. Airspeed	The propeller power and thrust (at the operating RPM) are plotted against the speed of the aircraft. The thrust will obviously match at the speed of your UAV at the selected flight leg, and will equal the required power.
Efficiency v. Airspeed	The efficiency of your propeller plotted against airspeed. A good design will maximize efficiency at the critical flight velocity.

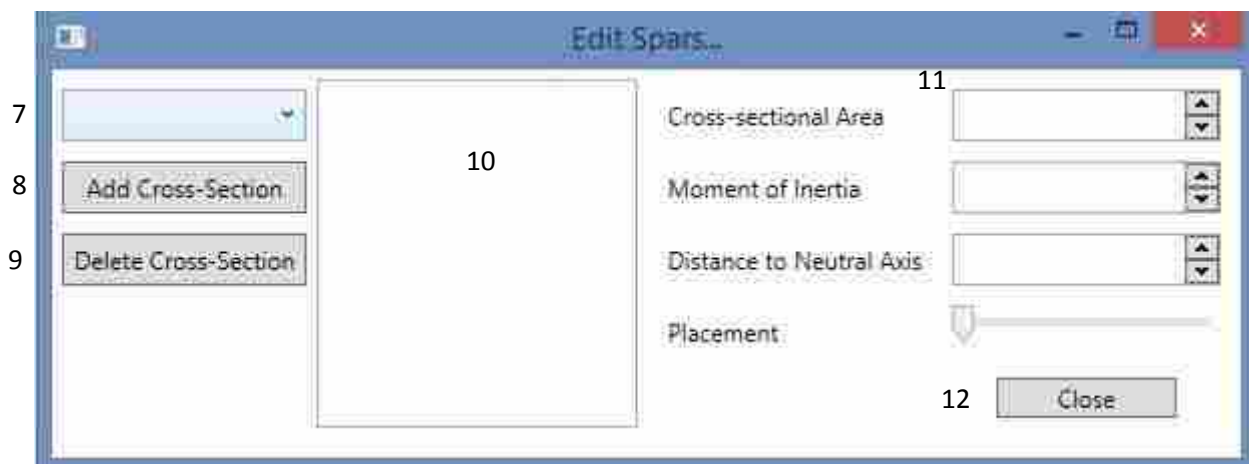
The calculations for the Propulsion tab are conducted with an Excel spreadsheet developed by Dr. Sullivan of Purdue University, edited to include plots of power, thrust, and efficiency against airspeed. This hopefully mimics much of the functionality of MotoCalc. If you download the spreadsheet (8), you can also have access to a database of motors and propellers borrowed from the one made available by the authors of DriveCalc⁷. These may not have currently available motors and propellers, but should at least give you some values to use during the initial stages of your design so you can learn what sort of properties you are looking for in a motor-propeller combination.

⁷ Giese, G., & Persson, C. (2010). Drive Calculator. Retrieved March 2, 2015, from <http://www.drivecalc.de/>

Structures



The Structures tab will help you size the structural components in your UAV, based off the lift distribution on the wing and a load factor applied to that lift. As of this version of CCAD, there is only capability to analyze a single spar per wing component using simple beam theory. The analysis assumes that the spar is the sole load-bearing member in the wing. To begin, an initial guess at spar geometry must be guessed. This guess must be iterated on during the sizing procedure. To do so, click on (1). A new window will be opened, shown below:



This window allows you to select a wing component in the drop-box (7). Once a component is selected, the current spar cross-sections, organized span-wise root-to-tip in (10). Each component will start with two cross-sections (root and tip). You can add a new cross-section by clicking (8), which will place a new (numerically labeled) cross-section in the middle of the furthest-spaced cross-sections. To delete a cross-section, select the cross-section you wish to delete in (10) and click (9). Each cross-section must have the inputs under (11) completely defined. This inputs are described in Table 9. You can close the window after finishing the spar definition by clicking (12).

Table 9: Description of the structural component (spar) inputs.

Cross-section input	Description
Cross-sectional Area	CCADE does not specify fixed cross-sectional shapes to use. You can select a shape, or none at all, and simply vary cross-sectional area.
Moment of Inertia	Just as cross-sectional area, you can independently define a moment of inertia. It may be helpful first to assume a square or circular cross-section and coordinate changes in cross-sectional area with changes in moment of inertia.
Distance to Neutral Axis	The neutral axis is the axis along which there is no bending moment in the beam. This can likewise be derived from the cross-sectional area after selecting a cross-section shape.
Placement	This is the span-wise distance from the root of the cross-section. Use the slider to arrange the cross-section. To assist, the slider will automatically lock onto locations where there is a change in the wing section.

Once the spar is defined, you must set a load factor for the analysis, the first field by (2). Then select the mission leg to pull required lift from, which is shown in the field below the combo-box by (2). The materials section by (5) is optional for basic analyses without deflection, but in order to calculate deflection, it is required to have the Young's modulus (or effective Modulus) of the material. The density is used in calculating weight. The results are calculated when an output to be plotted is selected in the combo-boxes by (4). The first combo-box is to select the component whose spar output will be plotted. The second combo-box contains the available outputs that can be plotted, and are described in Table .

Table 10: The spar outputs available to be plotted and downloaded.

Spar Output	Description
Lift	The span-wise lift distribution calculated from the aerodynamic analysis, multiplied by the load factor from (2). This is the load on the wing. Note that the plot is currently a 6 th order polynomial fit of the lift, and thus there are some unrealistic effects of waviness included. This is planned to be fixed in future versions with a Taylor-Series approximation (this is to obtain a distribution which can more easily be integrated).
Shear	This is the shear force in the spar, derived by integrating the lift distribution with respect to span-wise distance.
Moment	This is the bending moment in the spar, derived by integrating the shear distribution with respect to the span-wise distance.

Bending Stress	This is the bending stress of the spar, by using the bending moment in the equation for bending stress: $\sigma_b = \frac{My}{I}$ where y is the distance from the neutral axis to the furthest extent of the cross-section, and I is the moment of inertia.
Shear Stress	This is the shear stress in the spar using the equation for normal stress: $\sigma = \frac{V}{A}$ where V is the shear force, and A is the cross-sectional area.
Deflection	This is the deflection of the spar assuming a linearly elastic, isotropic material operating according to beam theory. Maximum deflection is $\delta = \frac{PL^3}{EI}$, where δ is the deflection, P the load on the wing, L the length of the beam to that point, E is the modulus of the material, and I the moment of inertia of the cross-section.

You can check multiple outputs to chart them on the same graph, but be warned that the chart will automatically scale. Deflection, for example, will be much less (normally) than the bending stress in scale. Generally speaking, the lift, shear, and moment are nicely plotted together; the bending stress and shear stress plot well together; and the deflection is plotted well on its own.

Since the calculations made in this module are so simple, the input is not available for download. You can download the output by clicking (6), which gives you the points used in plotting each of the outputs available.

Conclusion

CCADE is a great tool for beginning to evaluate the many concepts you may have for accomplishing the mission which satisfies your RFP for AerosPACE. As a collaborative conceptual design tool, CCADE can help your team learn better from each other and facilitate transfer of information via a structured database system. The software aims to significantly increase the number of design configurations analyzed by a team during the conceptual design phase and allows students to accomplish a fuller, more detailed exploration of the design space in a shorter amount of allotted time. The user interface of CCADE takes advantage of the distributive expertise of the design team and allows for both concurrent and independent design. This interactive, immersive design environment will eventually lead to a better overall UAV design to advance to preliminary and detailed design.

APPENDIX C. DESIGN EXPLORATION CODE

C.1 Optimus Configuration File

Optimus-CCADE Configuration, Design: test, Date: 2/3/2015 6:21:17 PM

Flight Parameters:

Required Cl: 0.3

Parasite Drag, CD0: 0.035

Component-Level Parameters:

Component	Group	Parameter	Min	Max	Nominal
-----------	-------	-----------	-----	-----	---------

Section-Level Parameters:

Component	Section	Parameter	Min	Max	Nominal
0	0	Sweep	10	50	30
0	1	Aspect	3	7	4
0	1	Dihedral	50	90	75

Control-Surface (Percent Chord) Parameters:

Component	Section	Control	Min	Max	Nominal
0	1	Elevon	10	50	30

Airfoil Parameters:

Component	Airfoil	Joint	Airfoils
0	0	0	cj25209.dat,e186.dat,e325.dat,e330.dat,e340.dat,eh1590.dat,eh2010.dat,mh61.dat,mh78.dat,phoenix.dat,rs001t10.dat,rs004a.dat,s5010.dat,s5020.dat,sd7003.dat
0	1	1	cj25209.dat,e186.dat,e325.dat,e330.dat,e340.dat,eh1590.dat,eh2010.dat,mh61.dat,mh78.dat,phoenix.dat,rs001t10.dat,rs004a.dat,s5010.dat,s5020.dat,sd7003.dat
0	2	2	cj25209.dat,e186.dat,e325.dat,e330.dat,e340.dat,eh1590.dat,eh2010.dat,mh61.dat,mh78.dat,phoenix.dat,rs001t10.dat,rs004a.dat,s5010.dat,s5020.dat,sd7003.dat

Current Controls:

Component	Section	Control
0	1	Elevon
1	0	Rudder

Current Airfoils:

Component	Airfoil	Joint	Airfoil
0	0	0	e325.dat
0	1	1	sd7003.dat
0	2	2	sd7003.dat
0	3	3	sd7003.dat
1	0	0	naca0021.dat
1	1	1	naca0021.dat

Constraints:

	Constraint	Min	Max	
	Static Margin	10	15	-
	Cm_alpha - pitching moment coefficient	-	0	-
	Cl_beta - Lateral stability derivative	-	0	-
	Cn_beta - Directional derivative coefficient	0	-	-

Objective	Type
L/D @ cruise alpha	Max

C.2 Parametric Workflow Construction Script

```
import os
import sys
import shutil
import optimus_extended as optex
from opt_ccade_utils import *

mainPath = os.path.join("C:\\", "Apps", "byupaper", "NafemsGui")
airfoilsPath = os.path.join(mainPath, "Build2", "Resources", "airfoils")

# INPUT FILES
optConfigFile = os.path.join(mainPath, "nafems_paper.optconfig")
vspFileIn = os.path.join(mainPath, "input.vsp3")
optFileIn = os.path.join(mainPath, "optcade.opt")

# OUTPUT FILES
scriptFile = os.path.join(mainPath, "optcade.vspscript")
vspFileOut = os.path.join(mainPath, "output.vsp3")
compGeomFile = os.path.join(mainPath, os.path.splitext(os.path.basename(
    vspFileOut))[0] + "_CompGeom.txt")
massPropFile = os.path.join(mainPath, "massprop.txt")
optFileOut = os.path.join(mainPath, "optcade_mod.opt")

print "
*****
"
print "* OPTIMUS - CCADE Interface script *"
print "
*****
"
print "* Settings: *"
```

```

print "* +-> Main Path.....: " + mainPath
print "* +-> Config file.....: " + optConfigFile
print "* +-> VSP file Input....: " + vspFileIn
print "* +-> OPTIMUS template..: " + optFileIn
print "* | *"
print "* +-> Output script file: " + scriptFile
print "* +-> VSP file Output...: " + vspFileOut
print "* +-> OPTIMUS out.....: " + optFileOut
print "
*****
"

try:
    configFile = []
    if (os.path.exists(mainPath) and os.path.exists(optConfigFile) and os.path.
        exists(vspFileIn)):
        print "* +--> Opening config file..."
        with open(optConfigFile) as f:
            configFile = f.readlines()
    print "* | +-> OK - got " + str(len(configFile)) + " lines."

#***** CREATION OF VSP SCRIPT FILE *****
print "* +--> Writing VSP script file..."
outfile = open(scriptFile, "w+")
print "* | |-> Writing header..."
outfile.write(getCCADEheader(vspFileIn))

optprj = optex.openExistingProject(optFileIn)
prjgraph = optprj.getGraph("Graph1")
for idx in range(prjgraph.numberInputVarArrays()):
    curArray = prjgraph.getInputVarArray(idx)
    if curArray.getName()=="OpenVSPInputs":

```

```

    OpenVSPInputs = prjgraph.getInputVarArray(idx)
    break
print "Got Input var array: "+ OpenVSPInputs.getName()

print "* | |-> Writing components..."
outfile.write(getCCADEcomponents(configFile, OpenVSPInputs))

print "* | |-> Writing footer..."
outfile.write(getCCADEfooter(vspFileOut, compGeomFile, massPropFile))
print "* | |-> Closing..."
outfile.close()
print "* | +-> Done!"
print "* +-> Job Finished!"

#***** Create AIRFOIL INPUTS *****
# TODO

#***** Create CONTROL SURFACES INPUTS *****
for idx in range(prjgraph.numberAnalyses()):
    curArray = prjgraph.getAnalysis(idx)
    if curArray.getName()=="AVLWriter":
        AvlWriterAction = prjgraph.getAnalysis(idx)
        break
print "Got action item: "+ AvlWriterAction.getName()
setAvlWriterAction(configFile, AvlWriterAction)

#***** Save the MODIFIED Optimus Project *****
optex.saveProject(optFileOut, True)

# Replace VSP SCRIPT template in Optimus
prjTplPath = os.path.join(os.path.splitext(optFileOut.replace('\\',r'\\'))[0]+'
    .optdir',"Graph1")

```

```

shutil.move(scriptFile, os.path.join(prjTplPath, os.path.basename(scriptFile)+'
    .tpl'))
shutil.copy(optConfigFile, os.path.join(prjTplPath, os.path.basename(
    optConfigFile)+' .tpl'))

# Close the Optimus project
optex.closeProject(optprj)

except Exception as e:
    print "* " + e.message
    sys.exit('* Error getting optimus-CCADE config file. Project dir may not exist
        or file is not existing!\n')

print "
    *****
    "

```


APPENDIX D. OPTIMIZATION AIRFOILS

This appendix contains the airfoil profiles and XFOIL results for the airfoils considered in the proof-of-concept optimization detailed in Section 5.3. Each airfoil has its own section. First a plot of the top and bottom surfaces of the airfoil is shown to illustrate the shape of the airfoil. In order to exaggerate the shape of the airfoil, they are not plotted to scale. Then, XFOIL-generated results are plotted for coefficient of lift versus angle of attack, coefficient of drag versus angle of attack, coefficient of pitching moment about the aerodynamic center versus angle of attack, and the drag polar (coefficient of lift versus coefficient of drag). The majority of these plots were obtained using CCADE, however, convergence for the Eppler 340 airfoil could not be obtained. The results displayed for this airfoil were obtained from the Airfoil Tools website [60].

D.1 CJ 25209

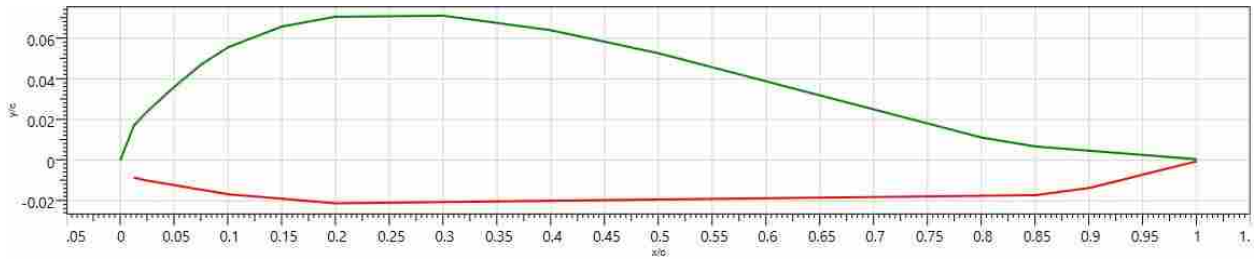
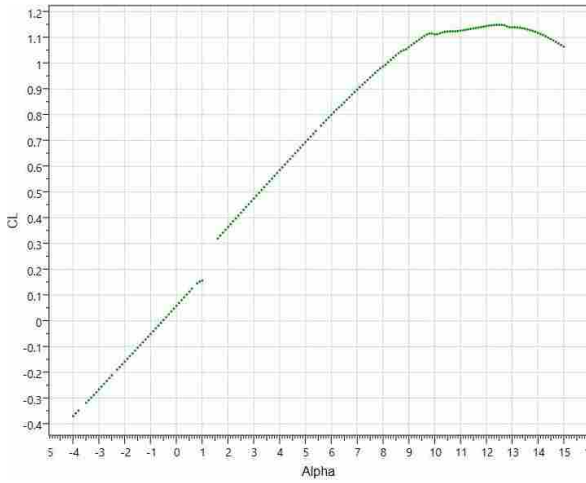
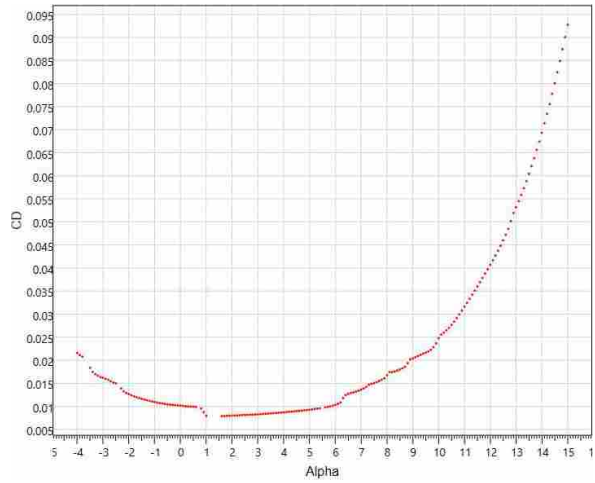


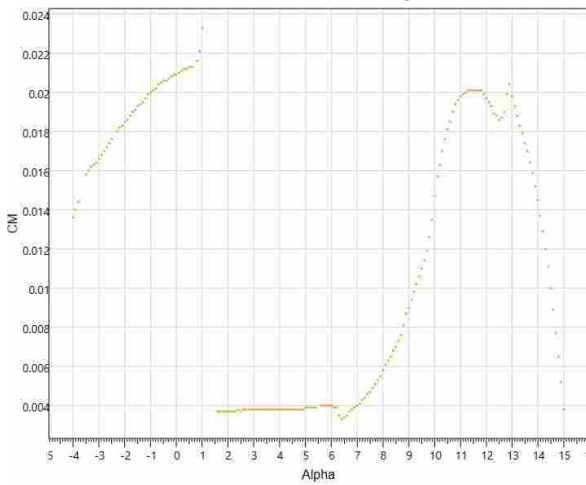
Figure D.1: CJ 25209 airfoil profile.



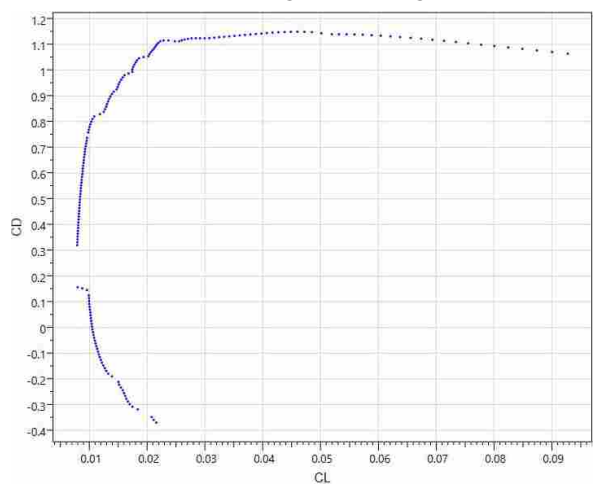
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.2: CJ 25209 airfoil performance at a Reynolds number of 500,000.

D.2 Eppler 186

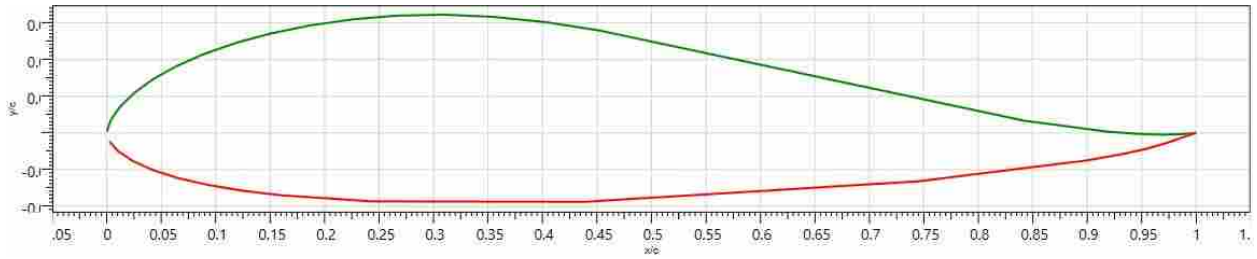
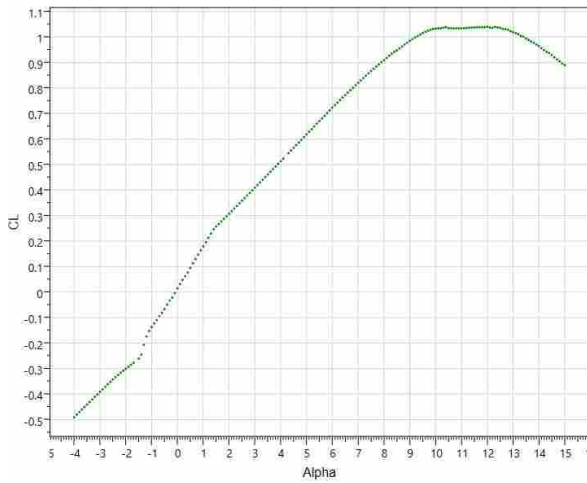
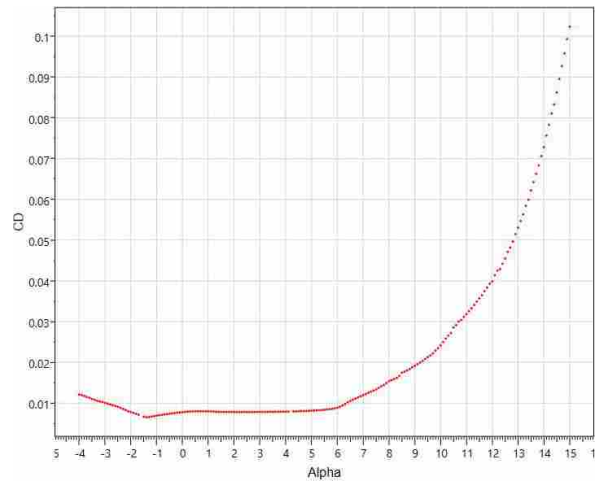


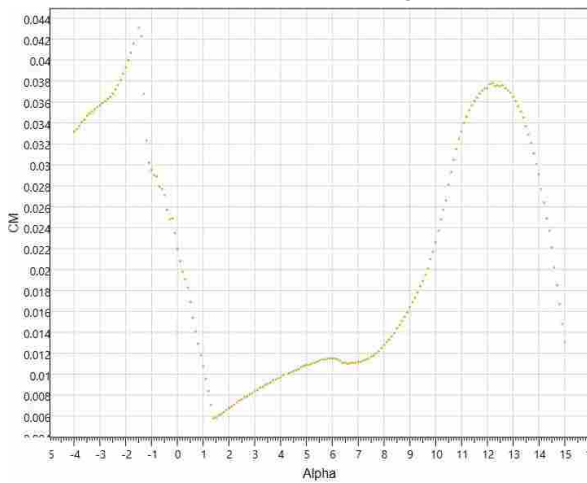
Figure D.3: Eppler 186 airfoil profile.



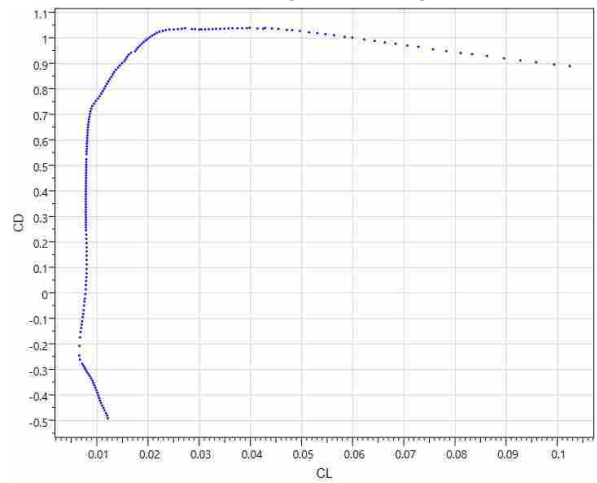
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.4: Eppler 186 airfoil performance at a Reynolds number of 500,000.

D.3 Eppler 325

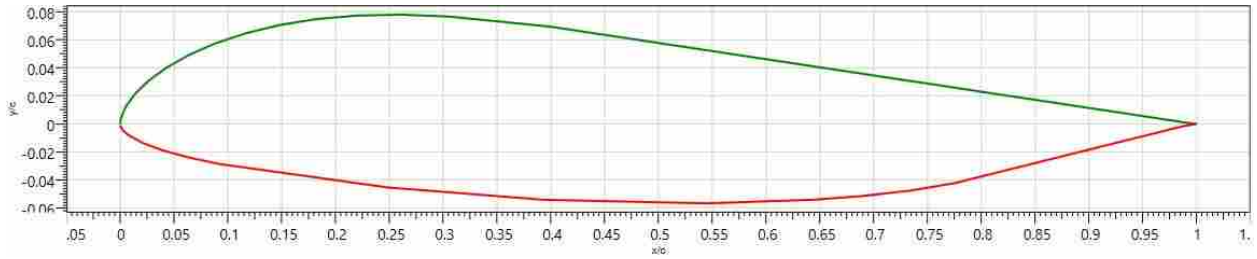
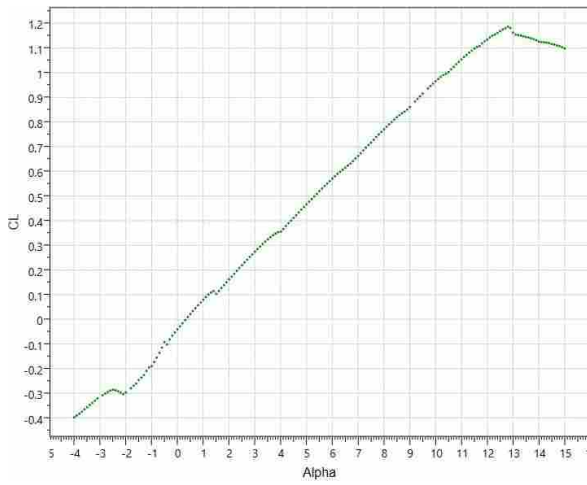
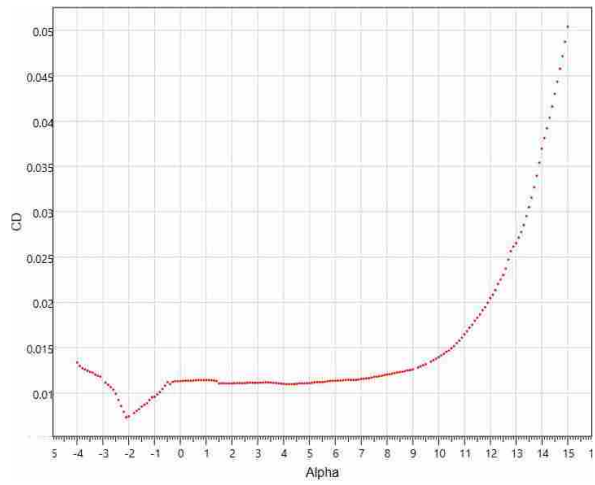


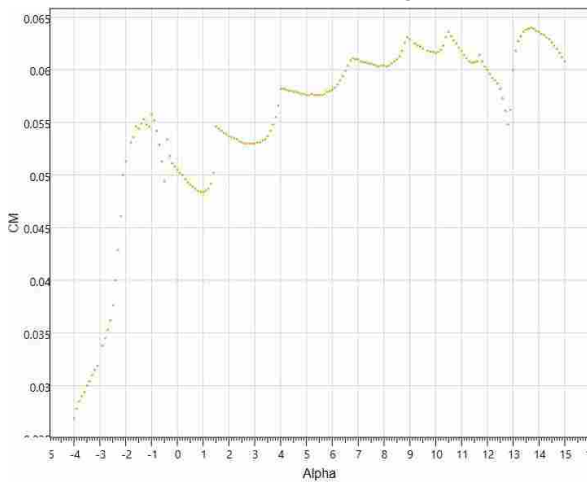
Figure D.5: Eppler 325 airfoil profile.



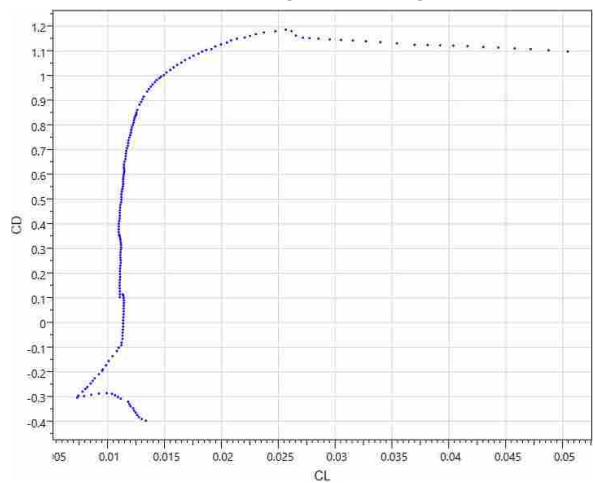
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.6: Eppler 325 airfoil performance at a Reynolds number of 500,000.

D.4 Eppler 330

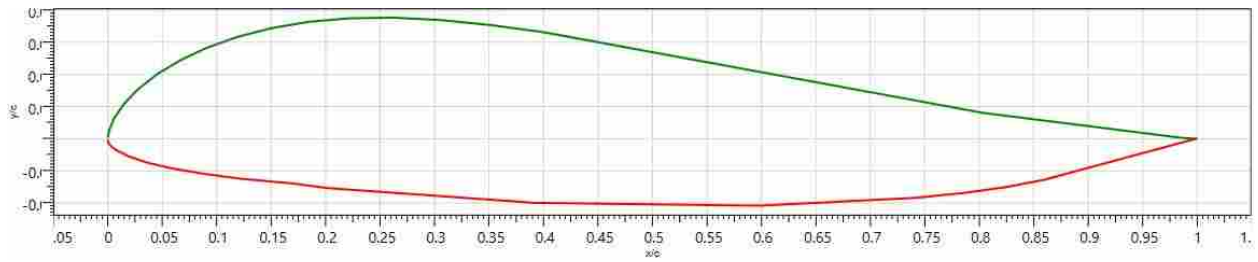
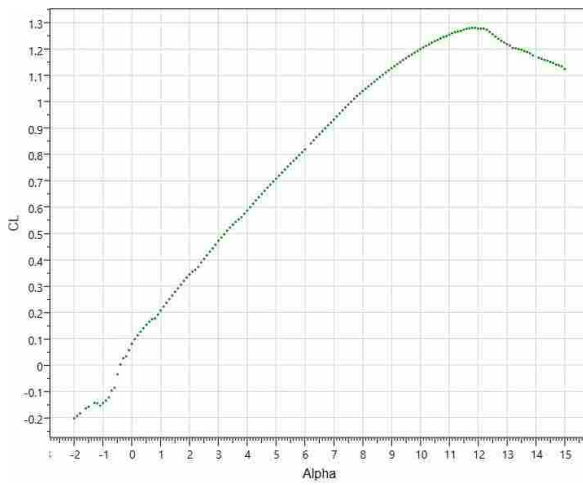
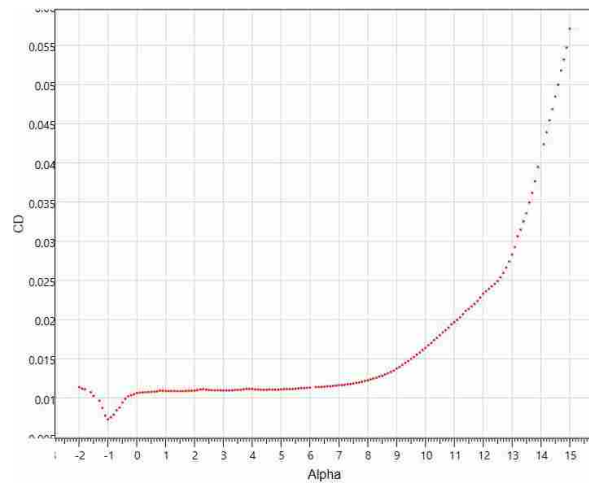


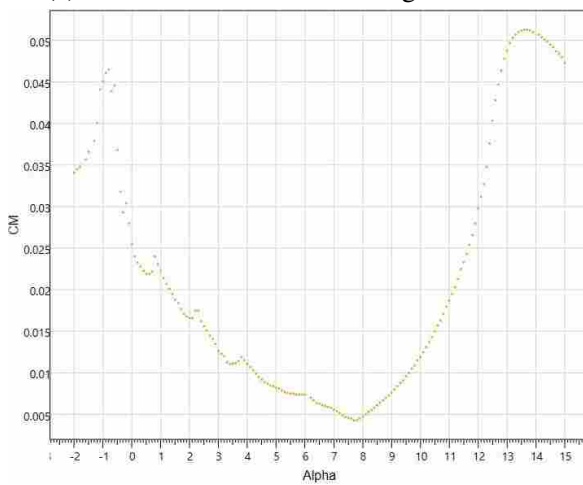
Figure D.7: Eppler 330 airfoil profile.



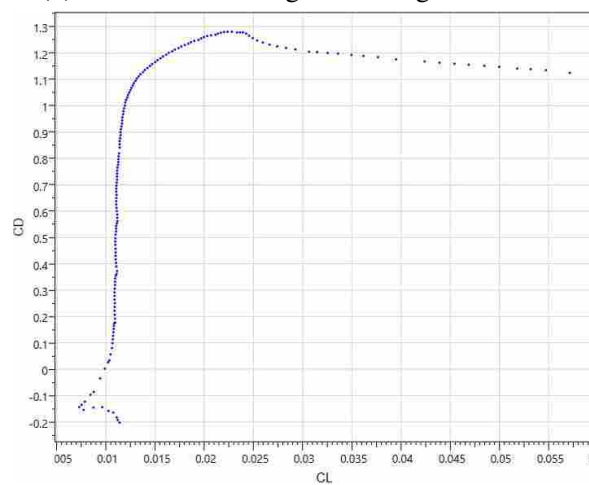
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.8: Eppler 330 airfoil performance at a Reynolds number of 500,000.

D.5 Eppler 340

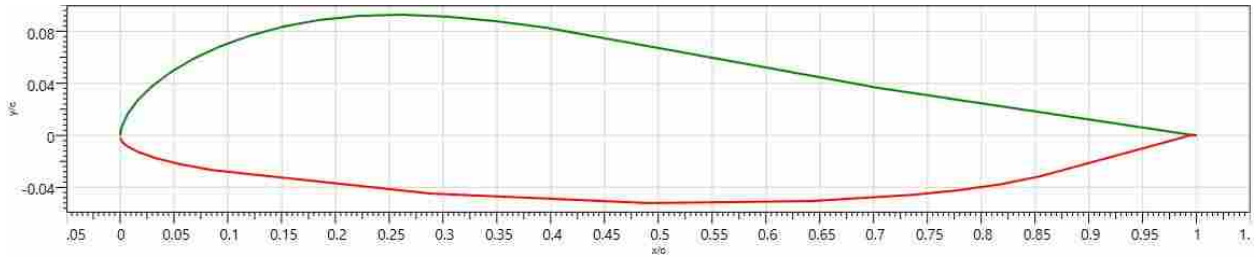
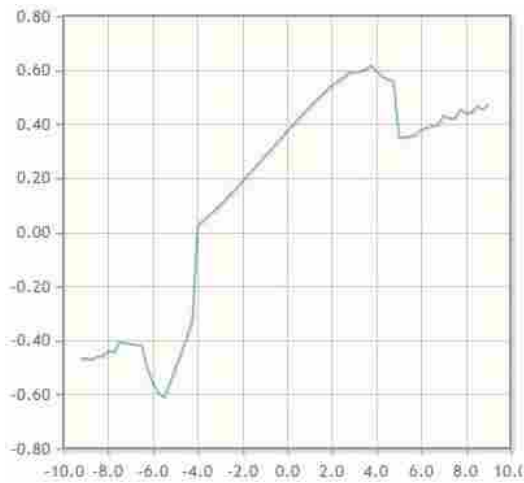
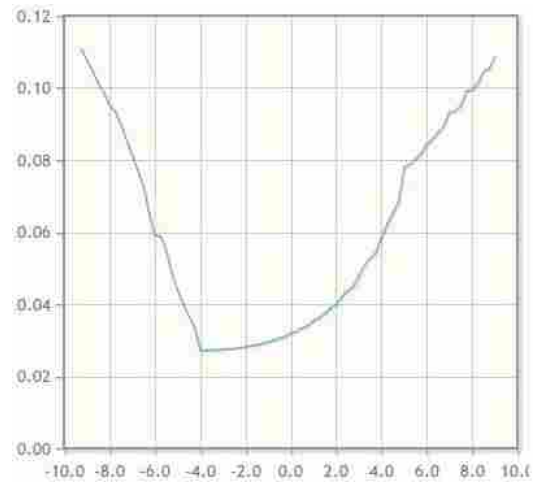


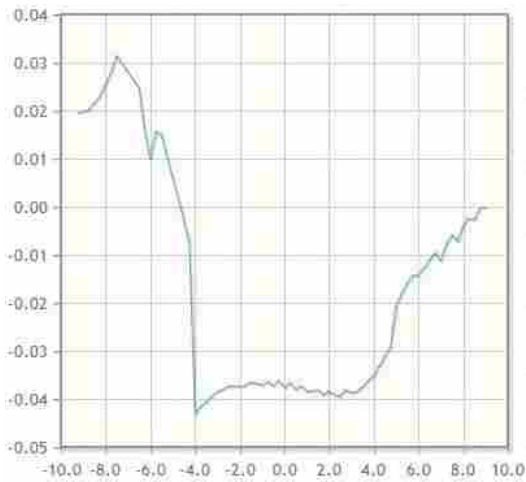
Figure D.9: Eppler 340 airfoil profile.



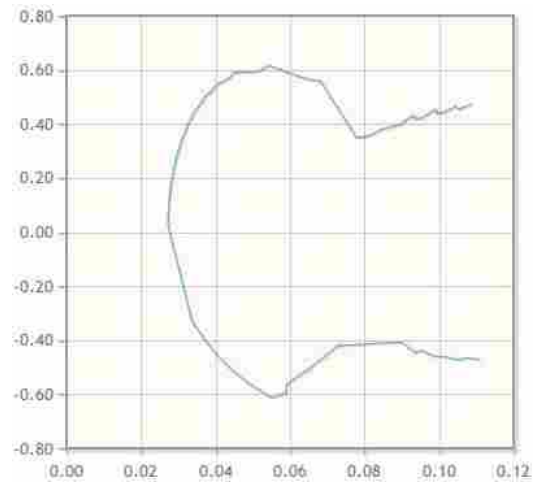
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.10: Eppler 340 airfoil performance at a Reynolds number of 500,000.

D.6 EH 1.5/9.0

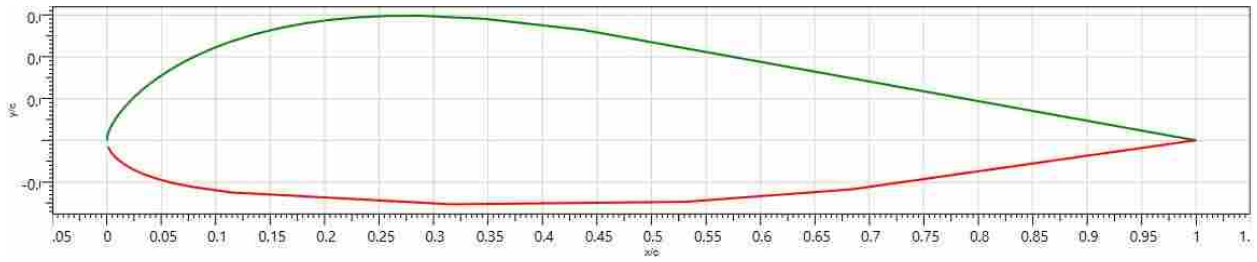
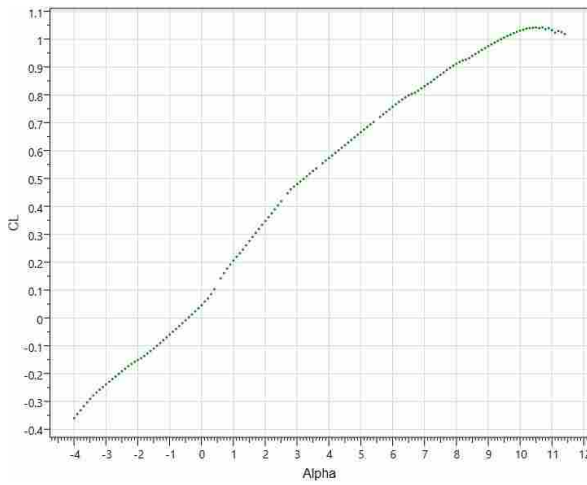
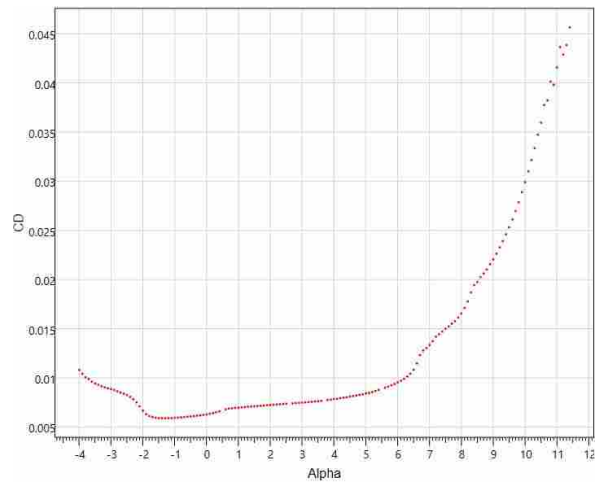


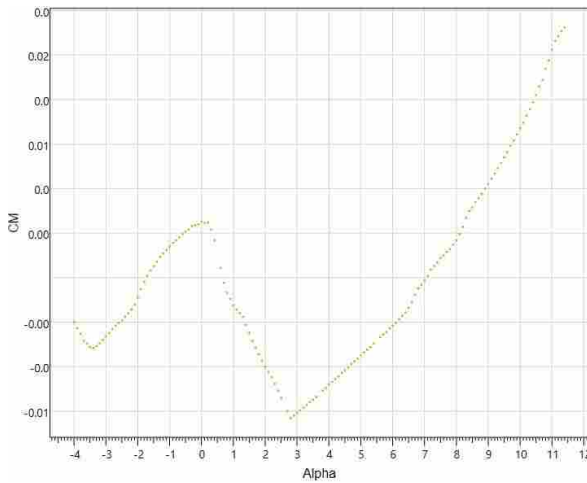
Figure D.11: EH 1.5/9.0 airfoil profile.



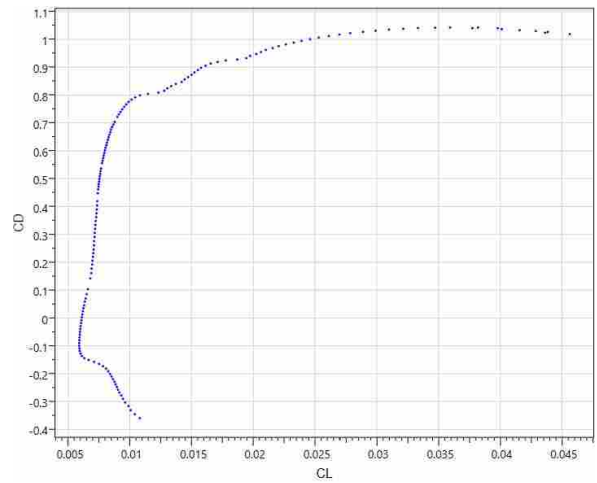
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.12: EH 1.5/9.0 airfoil performance at a Reynolds number of 500,000.

D.7 EH 2.0/10.0

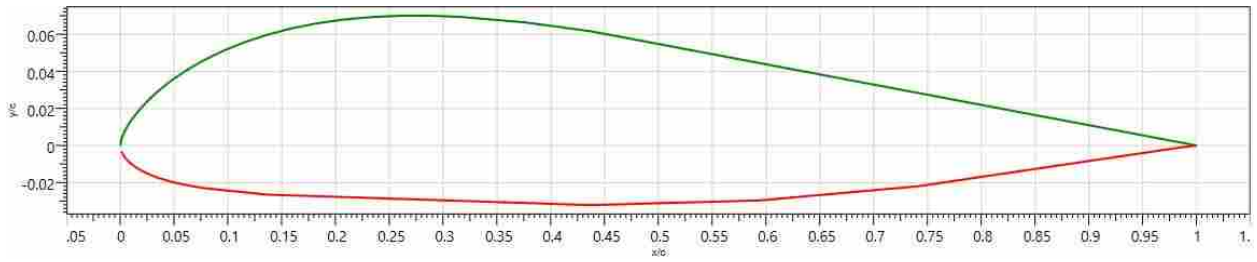
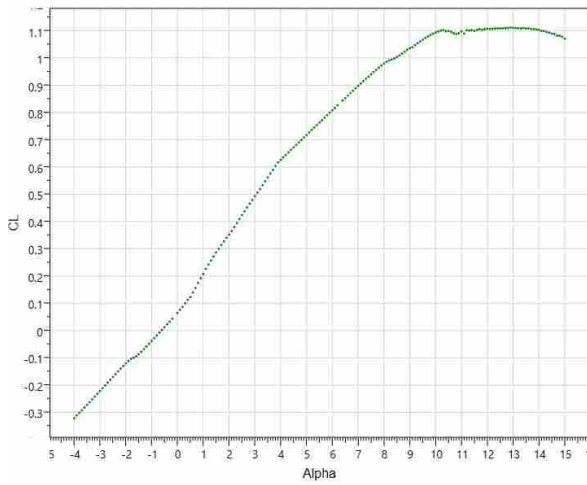
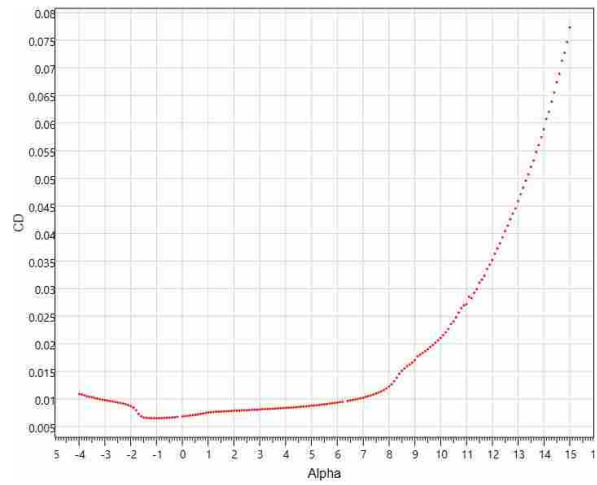


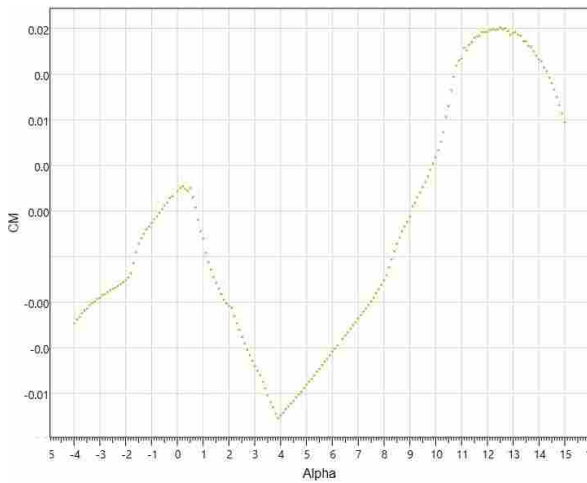
Figure D.13: EH 2.0/10.0 airfoil profile.



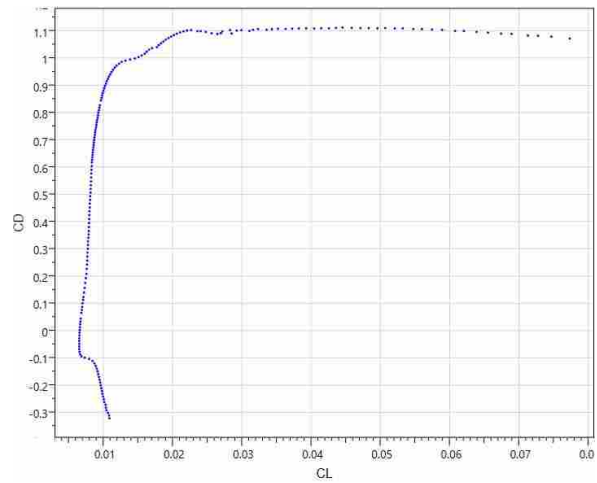
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.14: EH 2.0/10.0 airfoil performance at a Reynolds number of 500,000.

D.8 MH 61

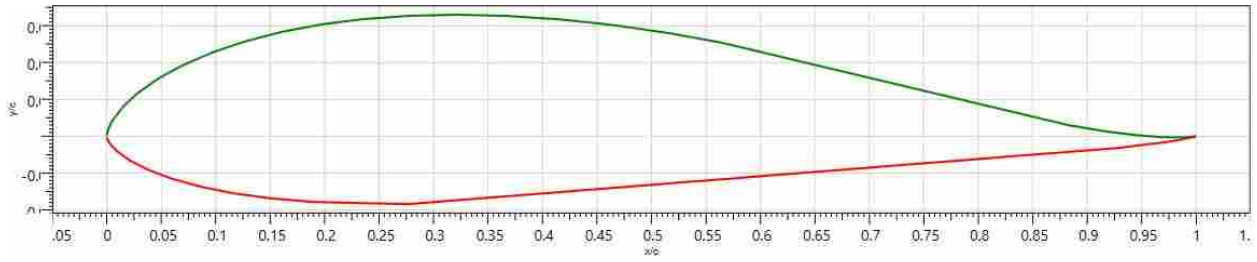
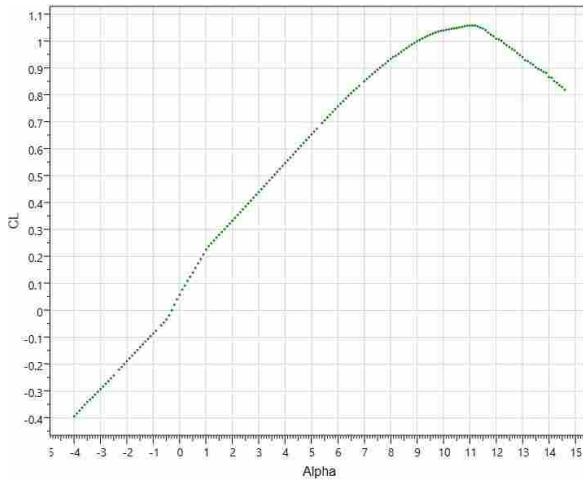
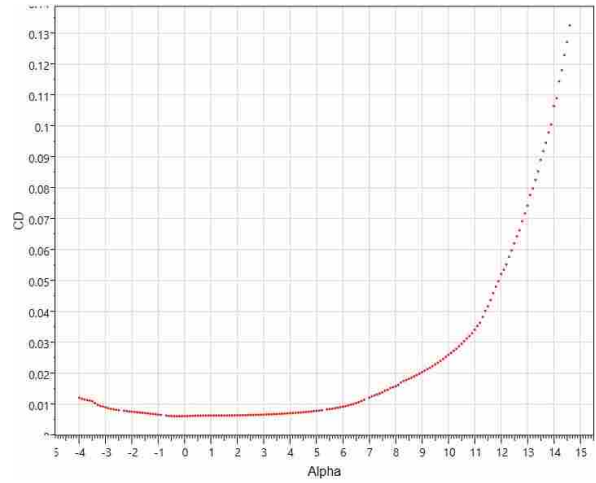


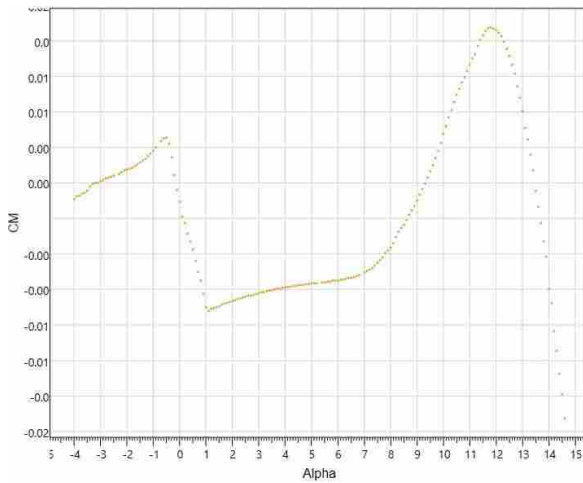
Figure D.15: MH 61 airfoil profile.



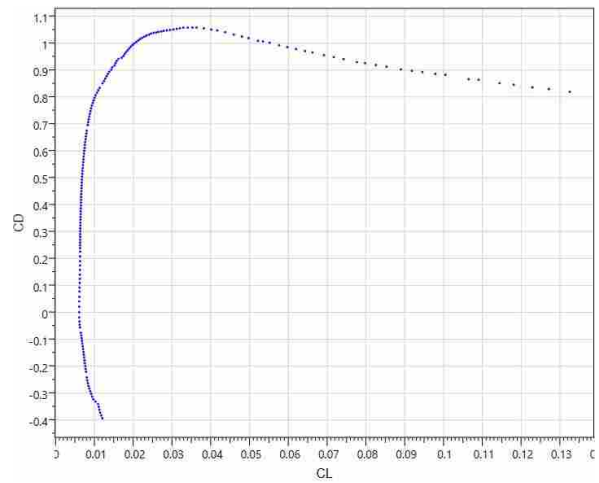
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.16: MH 61 airfoil performance at a Reynolds number of 500,000.

D.9 MH 78

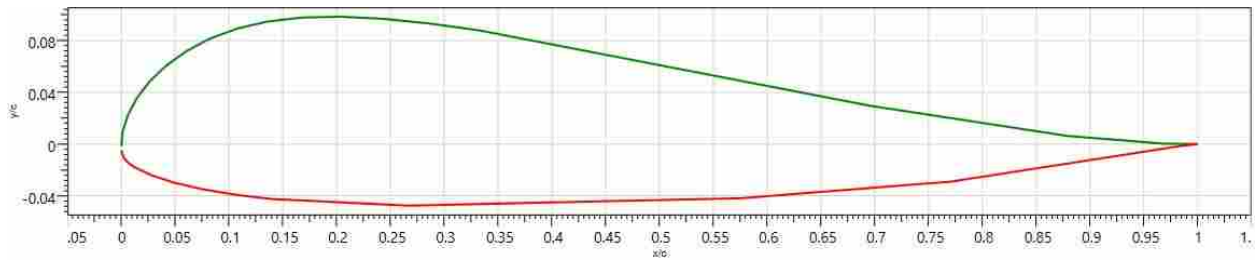
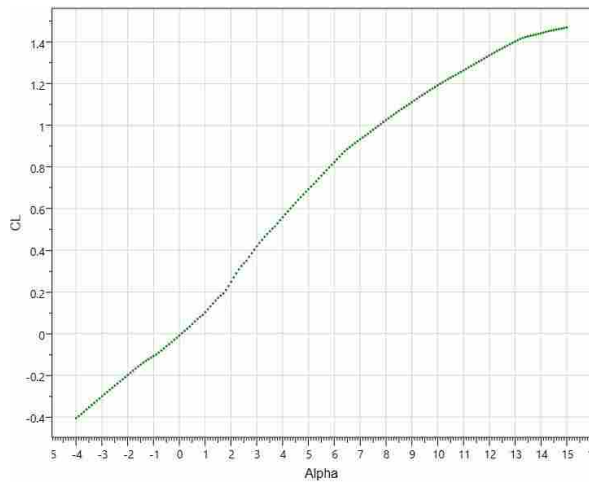
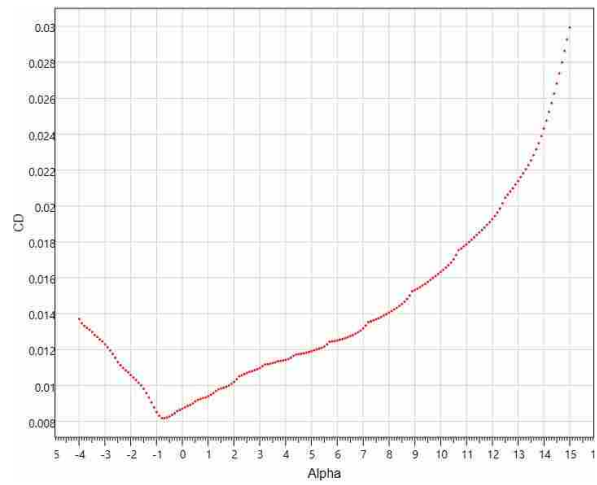


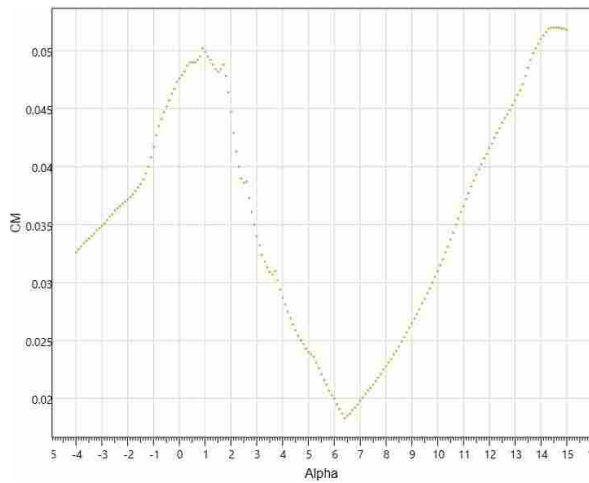
Figure D.17: MH 78 airfoil profile.



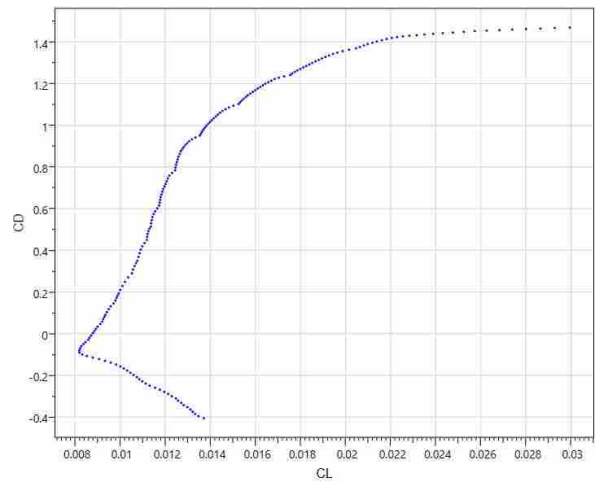
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.18: MH 78 airfoil performance at a Reynolds number of 500,000.

D.10 Phoenix

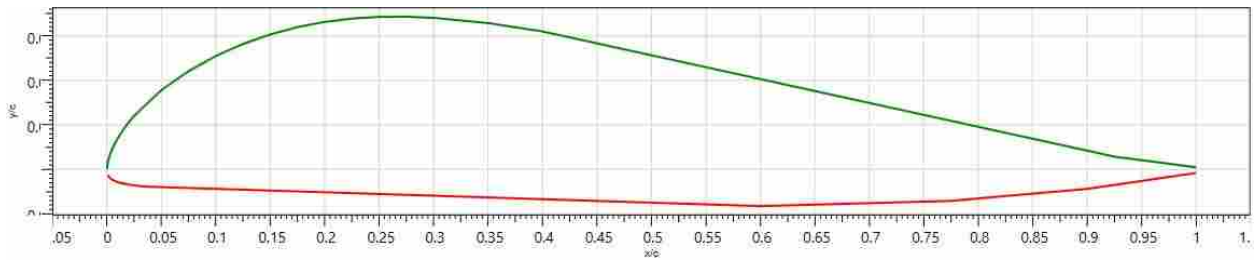
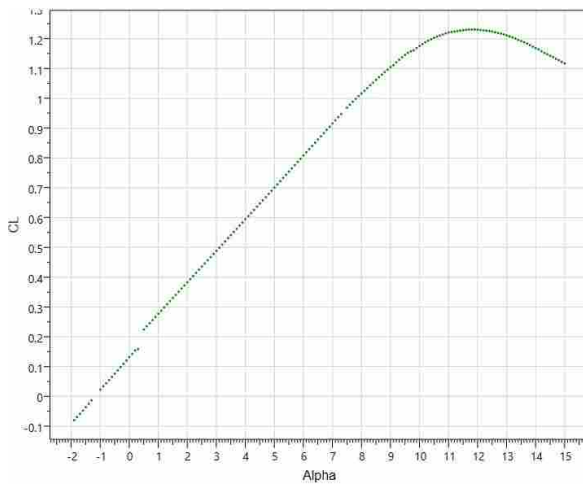
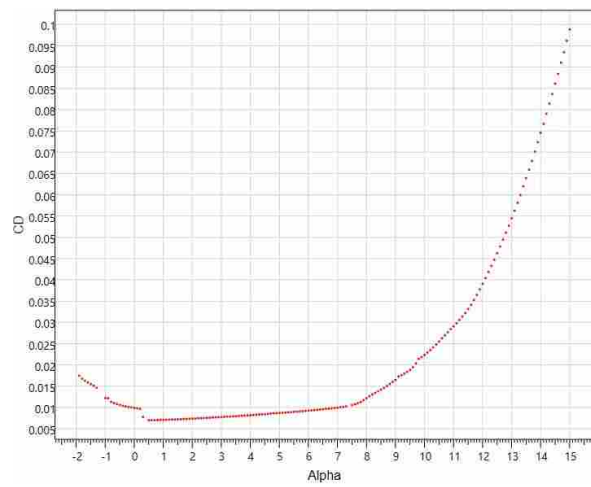


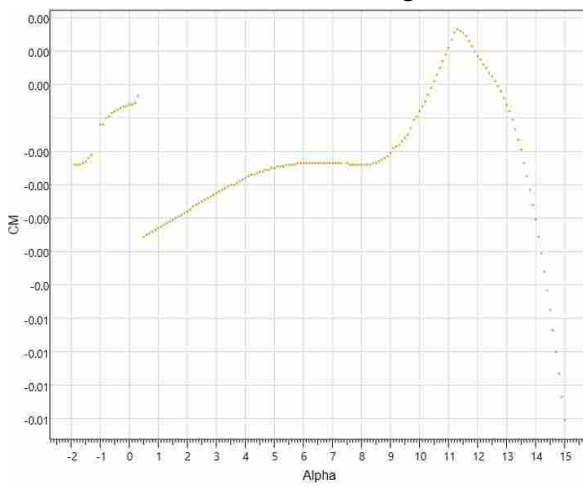
Figure D.19: Phoenix airfoil profile.



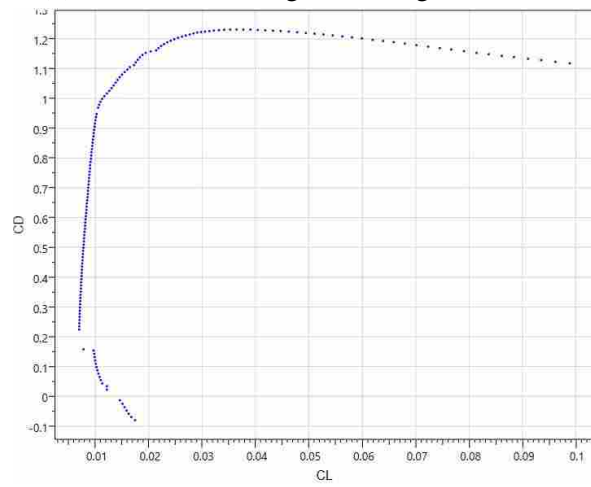
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.20: Phoenix airfoil performance at a Reynolds number of 500,000.

D.11 RS-001 T10

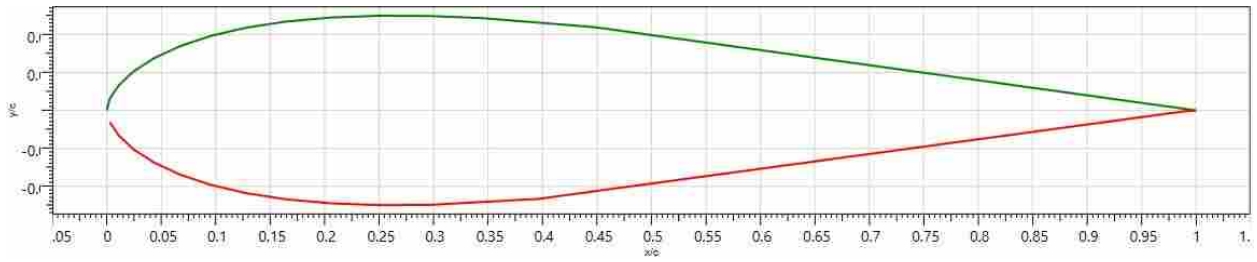
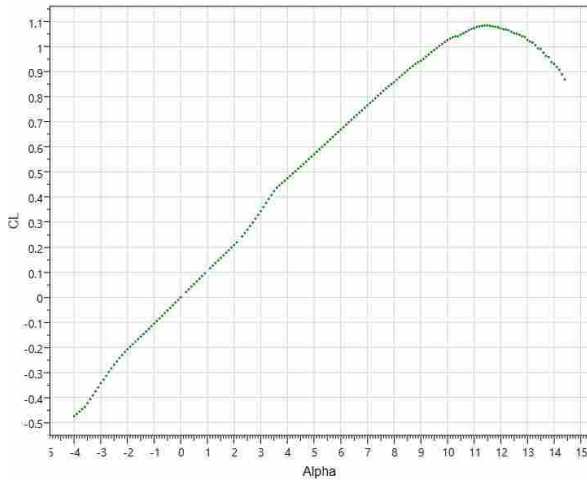
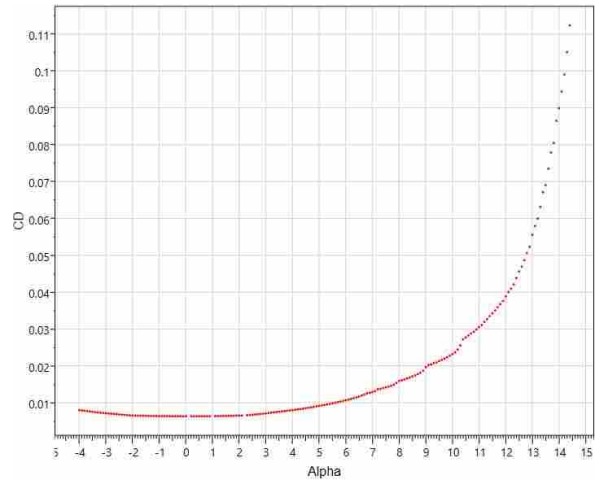


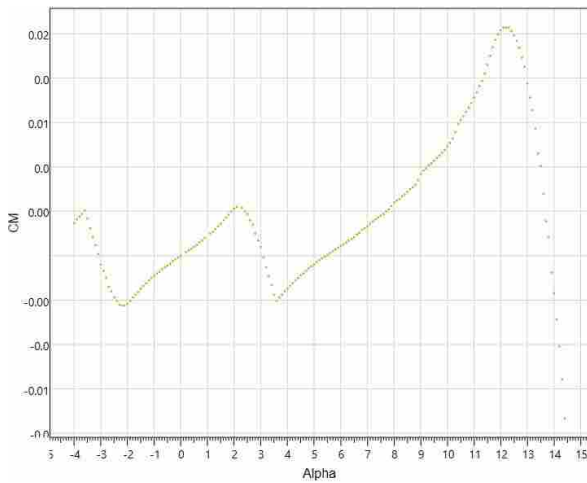
Figure D.21: RS-001 T10 airfoil profile.



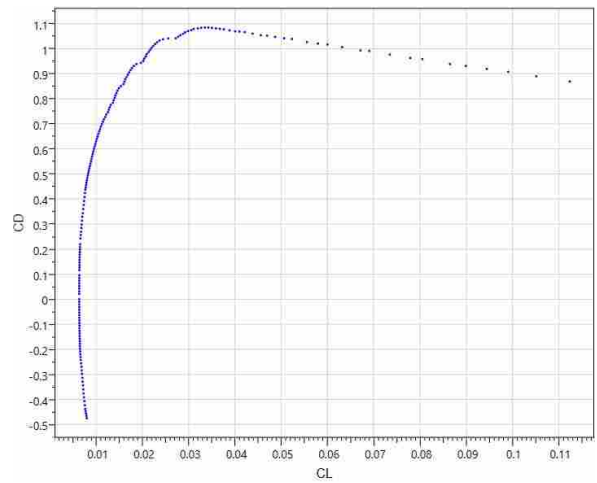
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.22: RS-001 T10 airfoil performance at a Reynolds number of 500,000.

D.12 RS-004 A

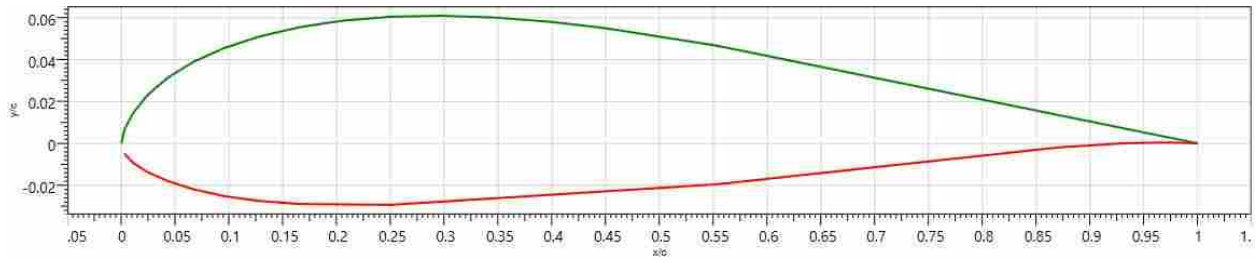
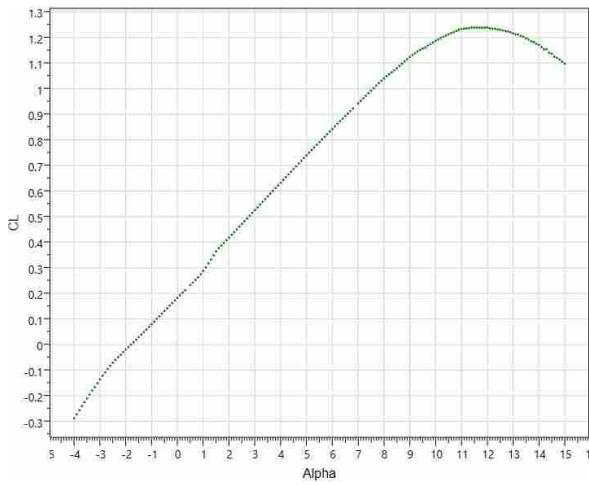
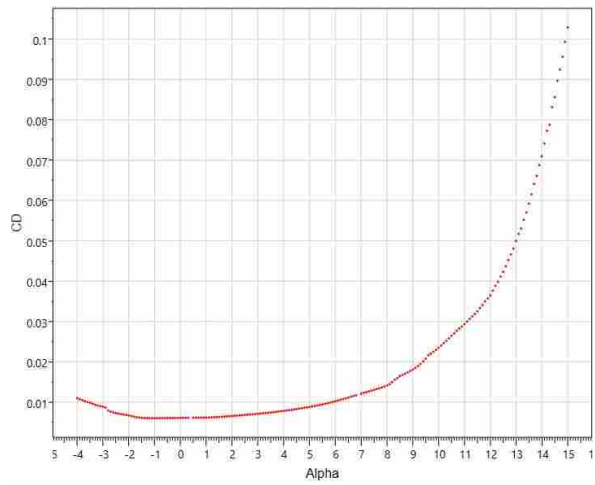


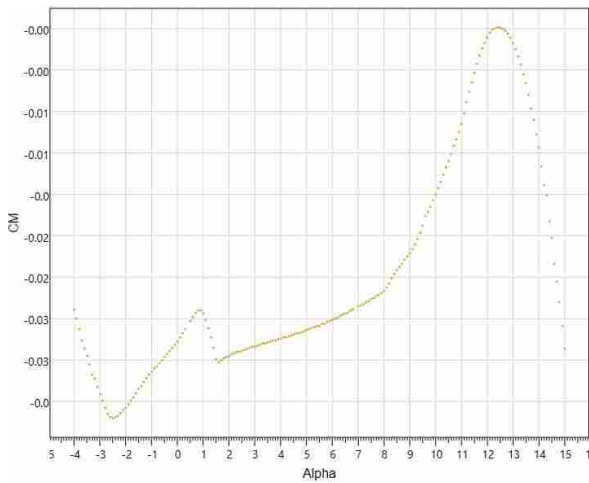
Figure D.23: RS-004 A airfoil profile.



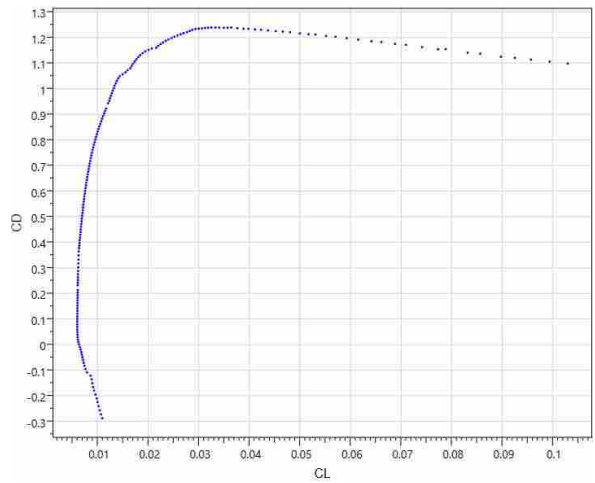
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.24: RS-004 A airfoil performance at a Reynolds number of 500,000.

D.13 Selig 5010

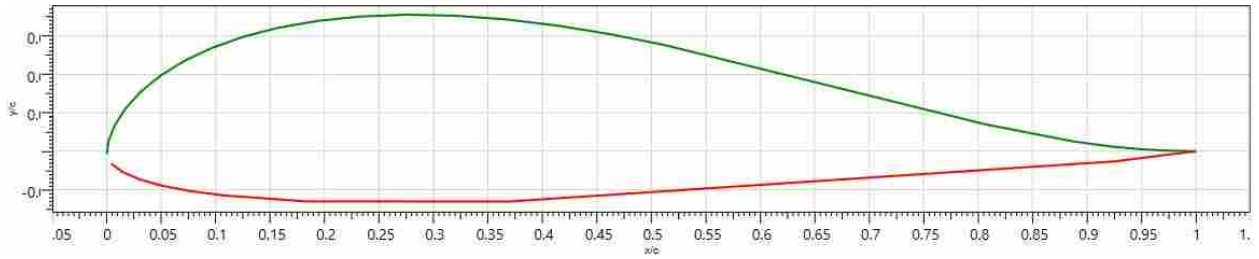
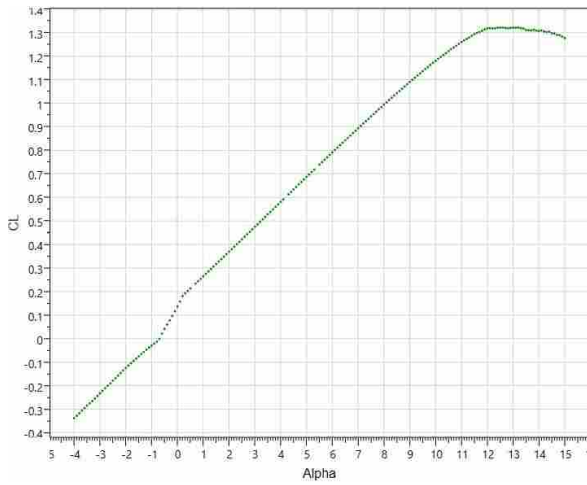
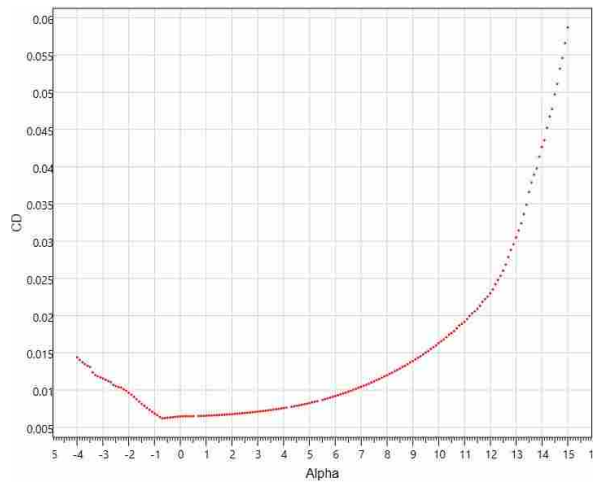


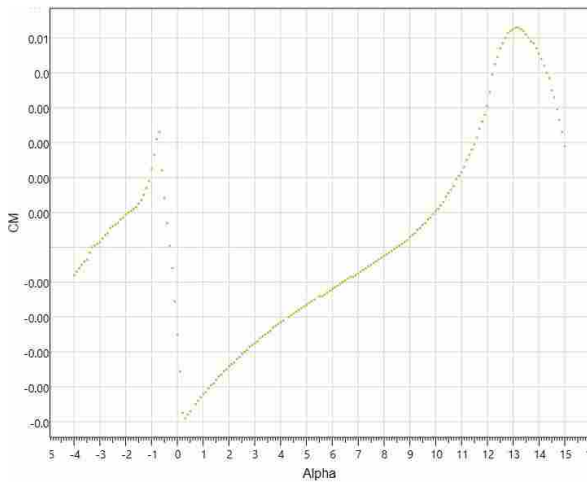
Figure D.25: Selig 5010 airfoil profile.



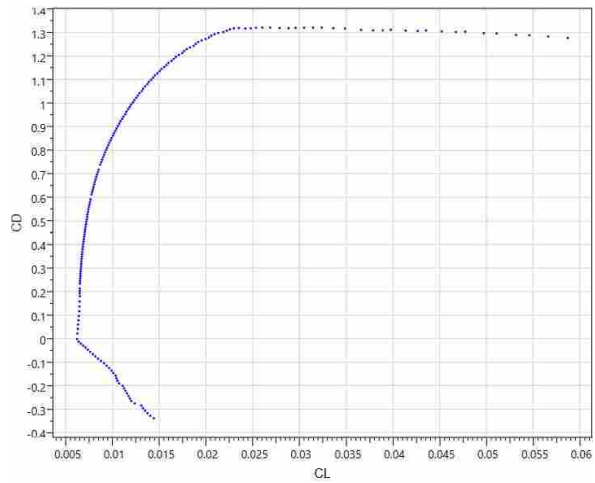
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.26: Selig 5010 airfoil performance at a Reynolds number of 500,000.

D.14 Selig 5020

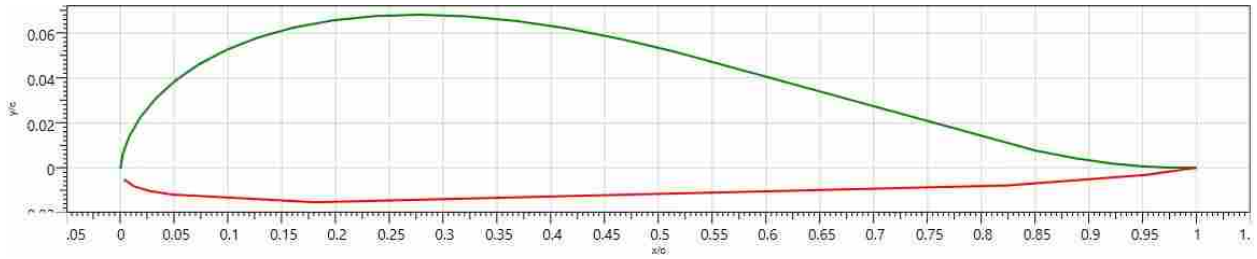
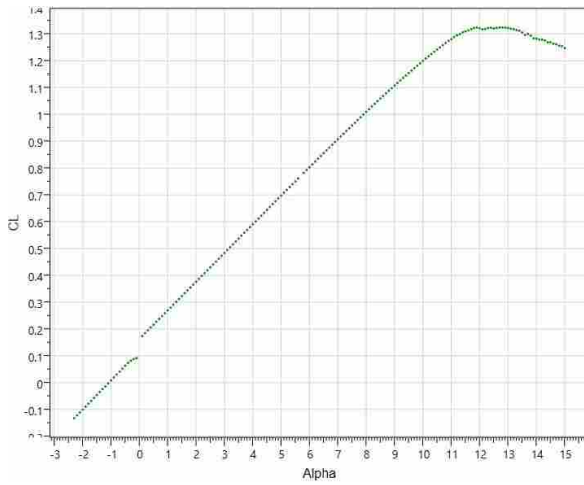
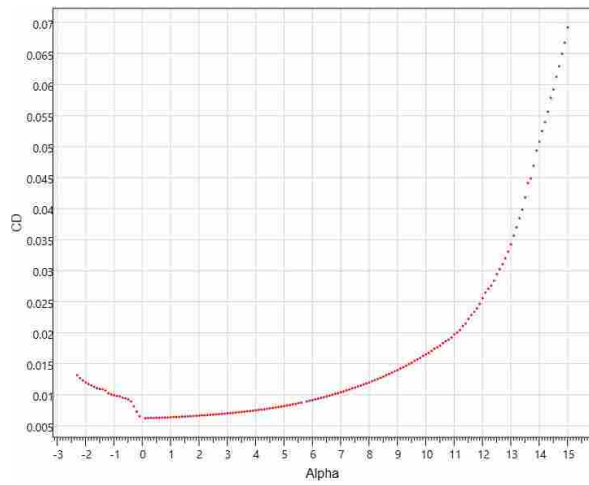


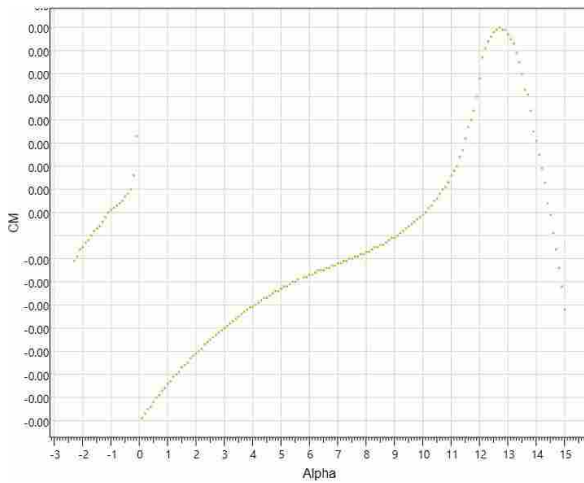
Figure D.27: Selig 5020 airfoil profile.



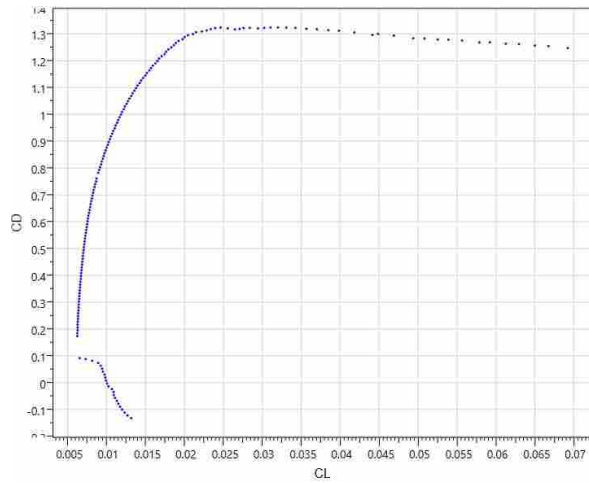
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.28: Selig 5020 airfoil performance at a Reynolds number of 500,000.

D.15 SD 7003

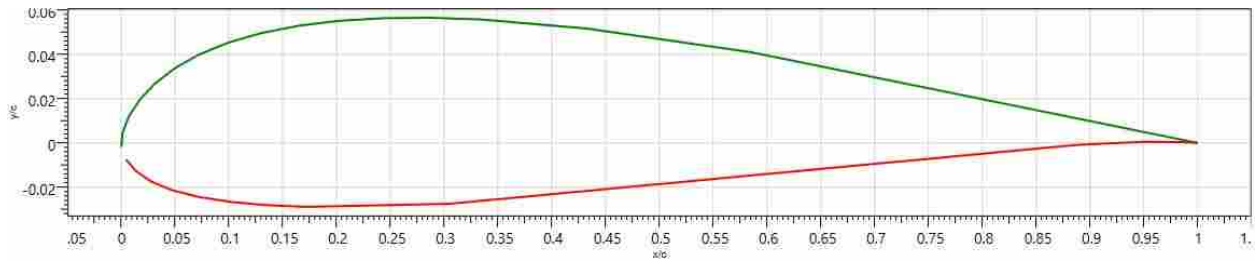
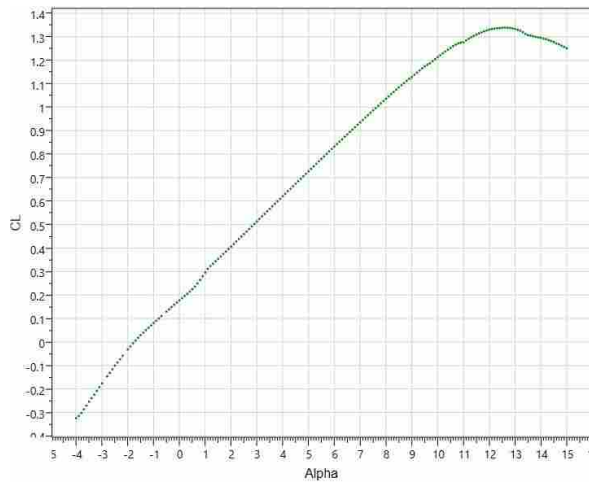
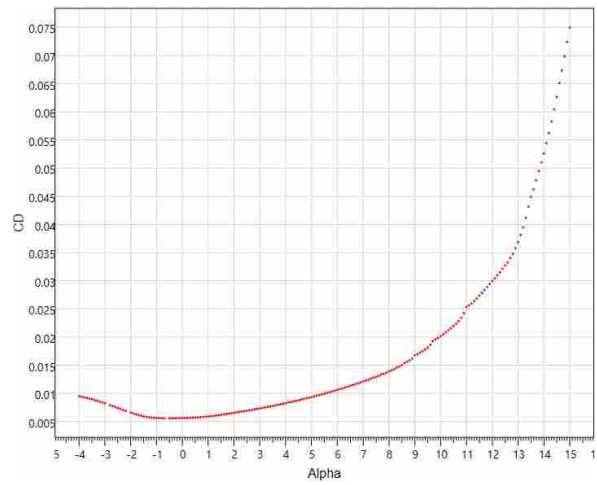


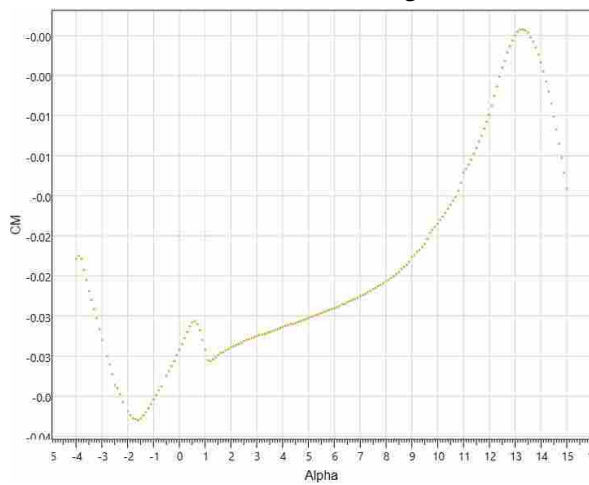
Figure D.29: SD 7003 airfoil profile.



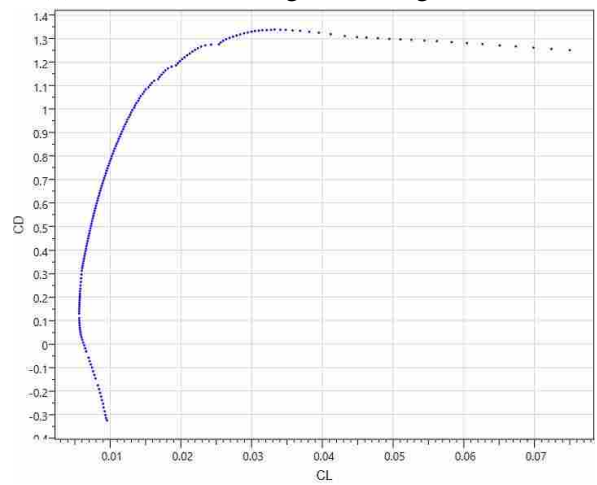
(a) Coefficient of lift versus angle of attack.



(b) Coefficient of drag versus angle of attack.



(c) Coefficient of moment versus angle of attack.



(d) Coefficient of lift versus coefficient of drag.

Figure D.30: SD 7003 airfoil performance at a Reynolds number of 500,000.