

2016

A high-fidelity approach to conceptual design

John Thomas Watson
Iowa State University

Follow this and additional works at: <http://lib.dr.iastate.edu/etd>



Part of the [Aerospace Engineering Commons](#), and the [Art and Design Commons](#)

Recommended Citation

Watson, John Thomas, "A high-fidelity approach to conceptual design" (2016). *Graduate Theses and Dissertations*. 15183.
<http://lib.dr.iastate.edu/etd/15183>

This Thesis is brought to you for free and open access by the Graduate College at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

A high-fidelity approach to conceptual design

by

John T. Watson

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Major: Aerospace Engineering

Program of Study Committee:
Richard Wlezien, Major Professor
Thomas Gielda
Leifur Leifsson

Iowa State University

Ames, Iowa

2016

Copyright © John T. Watson, 2016. All rights reserved.

TABLE OF CONTENTS

	Page
LIST OF FIGURES	iii
LIST OF TABLES	v
NOMENCLATURE	vi
ABSTRACT	vii
CHAPTER I INTRODUCTION: PROJECT BACKGROUND	1
CHAPTER II METHODOLOGY: HIGH FIDELITY CONCEPTUAL DESIGN APPROACH.....	19
Computer-Aided Drafting using Solidworks	20
Aerodynamic Analysis Using STAR-CCM+	31
Post-Analysis using MATLAB.....	41
Meta-Analysis using DSM.....	44
CHAPTER III RESULTS	47
CHAPTER IV DISCUSSION	53
Meta-Analysis	53
Design Candidate Variations	55
Included Preliminary Design Factors.....	55
CHAPTER V SUMMARY AND CONCLUSIONS	58
Summary	58
Conclusions.....	59
REFERENCES	60
APPENDIX A: JAVA SOURCE CODE FOR STAR-CCM+	63
APPENDIX B: VBA SOURCE CODE FOR SOLIDWORKS	74

LIST OF FIGURES

	Page
Figure 1.1. Processing power over time.....	1
Figure 1.2. SCCT Destination Range Distribution	4
Figure 1.3. Cabin Altitude vs. Flight Altitude for selected cabin pressures	5
Figure 1.4. Travel time vs. Cost per Seat-Mile for selected trip distances	6
Figure 1.5. Lockheed F-104 Starfighter.....	7
Figure 1.6. Aerospatiale/BAC Concorde	8
Figure 1.7 Lockheed-Martin F-22.....	9
Figure 1.8. Aerion AS2.....	10
Figure 1.9. Spike S-512	10
Figure 1.10 SAI QSST	11
Figure 1.11. Aircraft Design Phases	12
Figure 1.12. Single-Dimension Stick Airplane.....	14
Figure 1.13. Panel Aircraft representation.....	15
Figure 1.14. 3D CAD Aircraft Exterior	16
Figure 1.15. Design Structure Matrix terminology and format	18
Figure 2.1. CAD Construction Process.....	20
Figure 2.2: Un-dimensioned sketch	22
Figure 2.3: Fully Defined sketch	22
Figure 2.4: Angle of Incidence dimension for airfoil showing +3 and -2 degrees	23
Figure 2.5: Comparison of Sketch Construction; 3 dimensions vs. 9 dimensions	24
Figure 2.6: CAD Wing Loft Progression.....	25

Figure 2.7: Inlet geometry generated by NASA SUPIN.....	26
Figure 2.8: CAD Placement of Imported Engine Geometry.....	27
Figure 2.9: Complete Fluid Domain CAD.....	28
Figure 2.10: Design Table Excerpt.....	29
Figure 2.11: CAD Rebuild Failure.....	31
Figure 2.12. STAR-CCM+ Evaluation Workflow Schematic.....	32
Figure 2.13. STAR-CCM+ Simulation Topology Schematic.....	32
Figure 2.14. STAR-CCM+ Part Tessellation.....	33
Figure 2.15. STAR-CCM+ Surface Mesh.....	34
Figure 2.16. STAR-CCM+ Volume Mesh, Intake Detail.....	36
Figure 2.17. Supercruise Analysis DSM.....	45
Figure 3.1. Sensitivity of Range to STAR-CCM+ mesh resolution.....	47
Figure 3.2. Evaluation Time comparison for Solidworks analysis.....	48
Figure 3.3. Evaluation Time comparison for STAR-CCM+ analysis.....	48
Figure 3.4. Histogram of Candidate Population CAD export file sizes.....	49
Figure 3.5. Histogram of Candidate Population Evaluated Cruise Range.....	50
Figure 3.6. Comparison feasible candidates from multiple configuration options....	51
Figure 3.7. Design Study Best Performing Candidate.....	52
Figure 4.1. Wing Beam Bending Schematic.....	57

LIST OF TABLES

	Page
Table 1.1. SCCT Mission Requirements	2
Table 1.2. Typical Cabin Pressurization for Passenger Aircraft.....	6
Table 2.1 STAR-CCM+ Mesh Parameters	35
Table 2.2. STAR-CCM+ Physics Models used for SCCT Cruise Evaluation.....	38
Table 2.3. STAR-CCM+ Solver Parameters used for SCCT Cruise Evaluation.....	39
Table 2.4. STAR-CCM+ Reported Values used in SCCT Cruise Evaluation.....	39
Table 2.5. SCCT Candidate Feasibility Criteria	43

NOMENCLATURE

α	Angle of attack
ρ_{min}	Density at maximum altitude
ρ_{eval}	Density at evaluated altitude
$\hat{\mu}$	Estimated mean
$\hat{\sigma}$	Estimated standard deviation
b	Wing span
C	Cruise specific fuel consumption
CAD	Computer Aided Drafting
C_{DR}	Dutch Roll coefficient
L/D	Lift-to-Drag ratio
M_x	Yaw moment
M_y	Pitch moment
MDO	Multidisciplinary Design Optimization
V	Flight velocity
R	Cruise range
S	Planform area
W_0	Gross takeoff weight
W_3	Cruise initial weight
W_4	Cruise final weight
W_e	Empty weight
W_f	Fuel weight
W_p	Payload weight
X_{ac}	Aerodynamic Center location
X_{np}	Neutral Point location

ABSTRACT

The earliest conceptual phases of aircraft design present a challenging problem to engineers, due to the breadth of potential designs available and the depth of analysis required to choose between them. In this research, a new methodology was created to perform conceptual design analysis on aircraft, using off-the-shelf, high-fidelity software tools to explore the project design space, including important preliminary design factors and thereby producing a more robust result which is less subject to compromise at later design stages. We claim that this analysis can be performed in one hour with commonly available computation resources, and therefore is applicable to conceptual design. A case study was created to develop the method, particularly, the conceptual design of a supersonic transport jet. For this application, Solidworks was used to create a parameterized three-dimensional CAD solid to define the exterior geometry of the aircraft, and create populations of design candidates. The method also used STAR-CCM+ to perform an automated fluid flow analysis of these candidates, using three-dimensional, viscous, turbulent finite volume analysis and incorporating internal engine performance characteristics. Finally, MATLAB was used to collect the data produced by these analyses, compute additional results of interest, and quantify the design space represented by a population of candidates. We heavily automated the steps of this process, to allow large studies or optimization frameworks to be implemented. Our results show that the method produces a data set which is much richer than conventional conceptual design techniques. The method captures many interactions between aircraft systems which are normally not quantified until later phases of design: aerodynamic interactions between external lifting surfaces and between the external body and internal engine performance, and how structural constraints affect wing performance. We also produce detailed information about the aircraft static stability. Further, the method is able to produce these results with commonly available computer hardware within the one-hour timeframe we allow for a conceptual design analysis.

A project case study was formulated to lend a basis for the research project, and this study will be used in discussion at hand. The case study is an aircraft conceptual design project, where our methods were developed, tested and compared. As this development proceeded, valid conceptual designs were produced.

The case used for this purpose is the design of a small transport aircraft, capable of long-range supersonic cruise, called the SCCT Project. This project was driven by a group of industry leaders (SCCT Committee), who developed the requirements for the mission in accordance with their knowledge of the aircraft market. The committee is also responsible for making final design decisions regarding this aircraft; the research described in this thesis is used by the committee to understand the feasible design space of the intended vehicle, as well as the layout and configuration of the aircraft.

Table 1.1. SCCT Mission Requirements.

Total Range	5000 nmi
Passenger Count	20-30
Maximum Takeoff Weight	121,000 lb _f
Maximum Altitude	55000 ft
Cruise Mach	1.6

The requirements for the project are shown above in Table 1.1. The set of requirements was developed by the SCCT committee based on their analysis of the global aviation market, as well as the practical and regulatory limitations of passenger transport. However, it is necessary for the designer to understand the requirements in great detail, as the requirements must always be interpreted and interpolated to arrive at a complete design [3].

The gross takeoff weight of the aircraft should be less than 121,000 pounds. The general reason for this requirement is to keep the takeoff weight low, in order to drive down the vehicle cost. Nicolai shows that the takeoff weight is a reasonable predictor of unit cost for various aircraft types, and therefore can be used as a proxy during the early design stages, when it is infeasible to calculate a detailed bill of materials for the aircraft [4]. The specific value for the weight limit corresponds to noise regulation. Starting in 2017, new aircraft will be subject to

ICAO Chapter 14 regulations concerning takeoff and landing noise; the regulation progressively limits the noise for aircraft below 121,000 pounds [5][6]. Noise for supersonic aircraft has historically been a difficult challenge, and as such its constraints are being considered at the earliest possible phase of this design.

The target of 20-30 passenger seating is based on the market size of the aircraft, and the desired price point of the flight when finished. The desired cost of the flight is expected to be only slightly higher than contemporary first-class airline flights, i.e. less than \$20,000 USD [7]. The market for airfare of this price is expected to be small, and so the seat count is fairly minimal. The seating in the aircraft is considered to be typical of business class on a modern airline flight, with seat pitch roughly three feet. Since the aircraft cruises at supersonic velocity, the cruise phase is significantly shorter than aircraft of similar size; provisions for extended sleep are thus not required.

The SCCT committee created a matrix of the intended destinations of the vehicle, containing 24 major cities, to set a target for the range of the aircraft. For each possible trip between these destinations, the great-circle distance was computed using the airport latitude and longitude and the following form of the Vincenty formula [8]:

$$\Delta\sigma = \arctan \left(\frac{\sqrt{(\cos \phi_2 \sin \Delta\lambda)^2 + (\cos \phi_1 \sin \phi_2 - \sin \phi_1 \cos \phi_2 \cos \Delta\lambda)^2}}{\sin \phi_1 \sin \phi_2 + \cos \phi_1 \cos \phi_2 \cos \Delta\lambda} \right). \quad (1)$$

Then, an interval of potential vehicle range distances was evaluated. For each range, the proportion of feasible trips was computed. The range target was selected where a small peak in this distribution occurs; this corresponds to a range that gives a large number of possible destinations without excess capability. This decision was also heavily informed by the historic knowledge of the committee members. The distribution is shown in Figure 1.1 below.

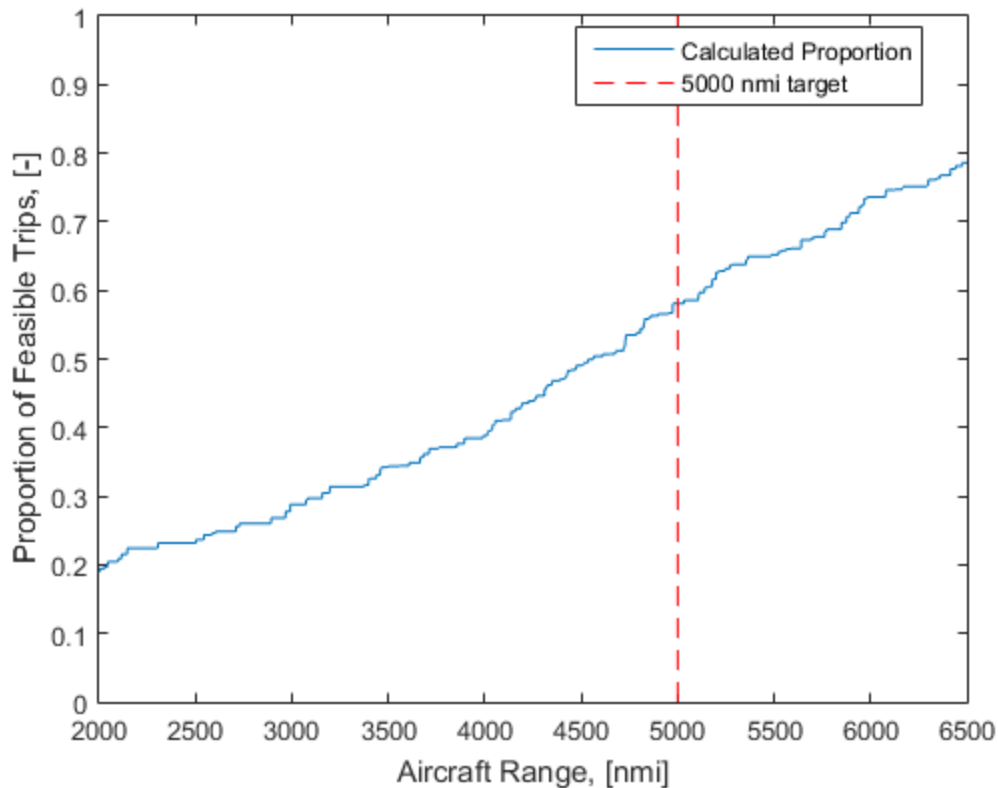


Figure 1.2. SCCT Destination Range Distribution.

Another important constraint on the SCCT design is the maximum cruising altitude. Generally, increased cruise altitude corresponds to better cruise performance, but this comes at the expense of weight in the passenger pressure vessel structure as well as human safety in the event of decompression. The United States Federal Aviation Administration (FAA) mandates that the cabin altitude (the altitude corresponding to the pressure in the cabin) must be maintained at 8000 feet or lower. At higher altitudes, hypoxia compromises the health and safety of crew and passengers. There is a trend among corporate aircraft manufacturers to fly at even lower cabin altitudes for reasons of passenger comfort, but attempts to study this have proven inconclusive [9]. The relationship between the flight altitude and cabin altitude is shown in Figure 1.2 below; some typical pressure values are shown in Table 1.2. To maintain the required cabin altitude for our case, the cabin must be pressurized to 9.64 pounds-force per square inch (psi). This corresponds well with similar aircraft.

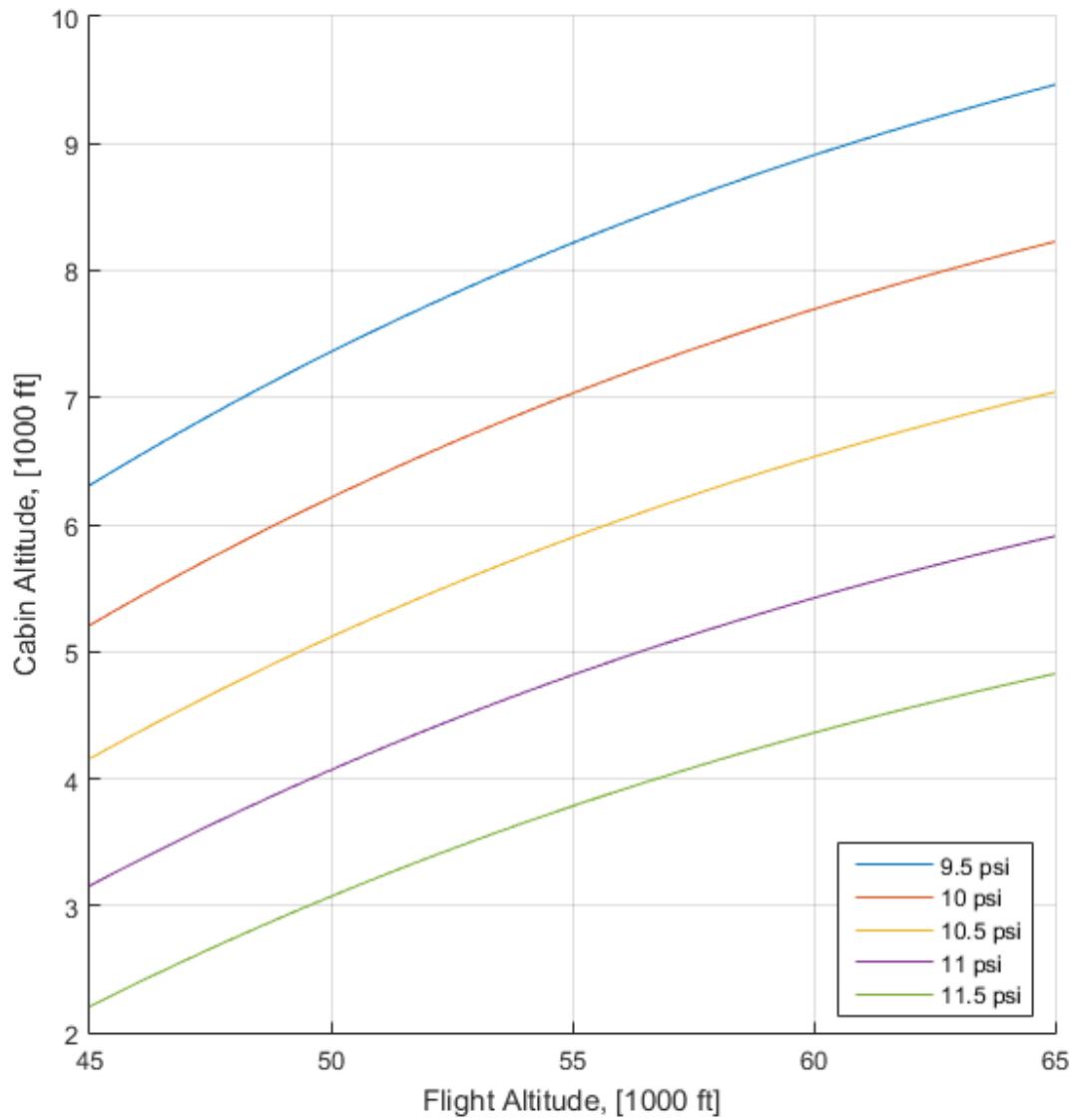
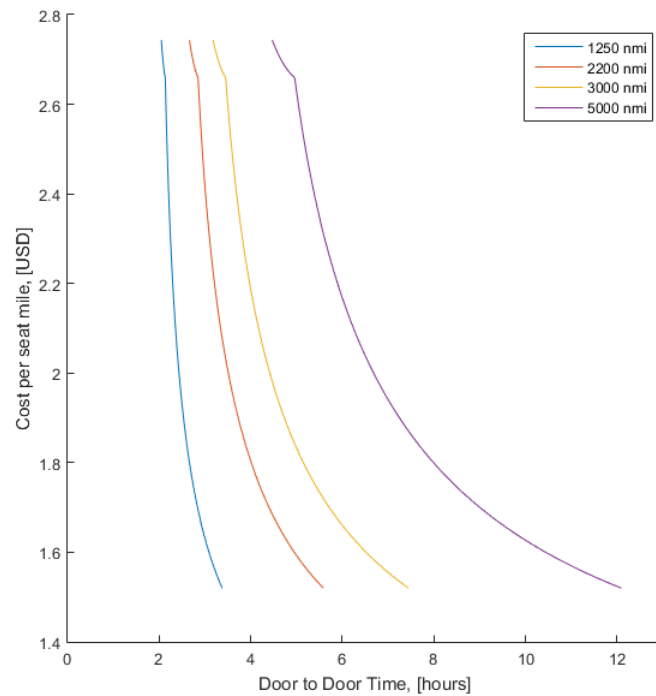


Figure 1.3. Cabin Altitude vs. Flight Altitude for selected cabin pressures.

Table 1.2. Typical Cabin Pressurization for Passenger Aircraft.

Aircraft	Cruise Altitude, [ft]	Pressurization, [psi]
BAE Concorde	60000	10.5
Bombardier Global Express	41000	9.7
Boeing 767	39000	8.5
Boeing 787	39000	8.9
Syberjet SJ30	41000	12.1

The choice of desired cruise Mach number was made by the SCCT committee based on flight time for the desired trips. To inform this decision, an analysis of the cost per seat-mile and the total travel time was created. The operating point was then selected to give the maximum improvement in flight time while staying within the desired operating cost. The resulting operating point was in the vicinity of Mach 1.6; the trade analysis is shown below in Figure 1.3.

**Figure 1.4. Travel time vs. Cost per Seat-Mile for selected trip distances.**

Some comparison to historic supersonic aircraft is useful to provide additional background for the SCCT project. One of the first designs built specifically for supersonic performance was the F-104 Starfighter. Its design was a response to USAF pilots' need for a lighter, more maneuverable replacement for the F-86 and F-100 aircraft, and was optimized for transonic performance [10].

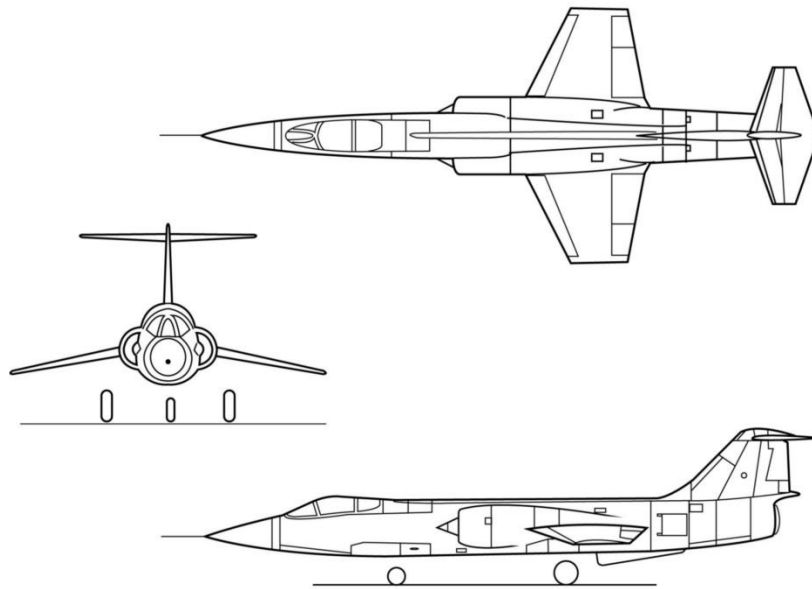


Figure 1.5. Lockheed F-104 Starfighter [2].

The performance requirements drove several notable features of this design. First, the wing planform is extremely small. This raised the maneuvering speed of the aircraft to the desired regime without excessively increasing drag. The wing airfoil section was very slender and sharp to reduce wave drag. These features provided the desired performance, but with the side effect of raised landing speed; a blown flap was required to reduce the speed at touchdown from 220 knots to 170, and a drogue chute was used on a normal landing. The use of a large turbojet engine with blown flaps also resulted in extreme ground noise. [10].

The 1960's brought the design of the Concorde, the first supersonic transport aircraft. This aircraft marked several key advancements in the field. The chief improvement was the use

of a slender delta wing to provide acceptable performance in supersonic flight while retaining a low landing speed. The tradeoff for this low landing speed was a high angle of attack, requiring a complex and heavy droop nose to give the pilot visibility. The design also used a highly advanced engine installation, with variable inlet geometry and a high-temperature compressor. The overall pressure ratio of the engine at cruise was 82:1, and the high thrust of the engine allowed the aircraft to cruise without afterburner use [11].

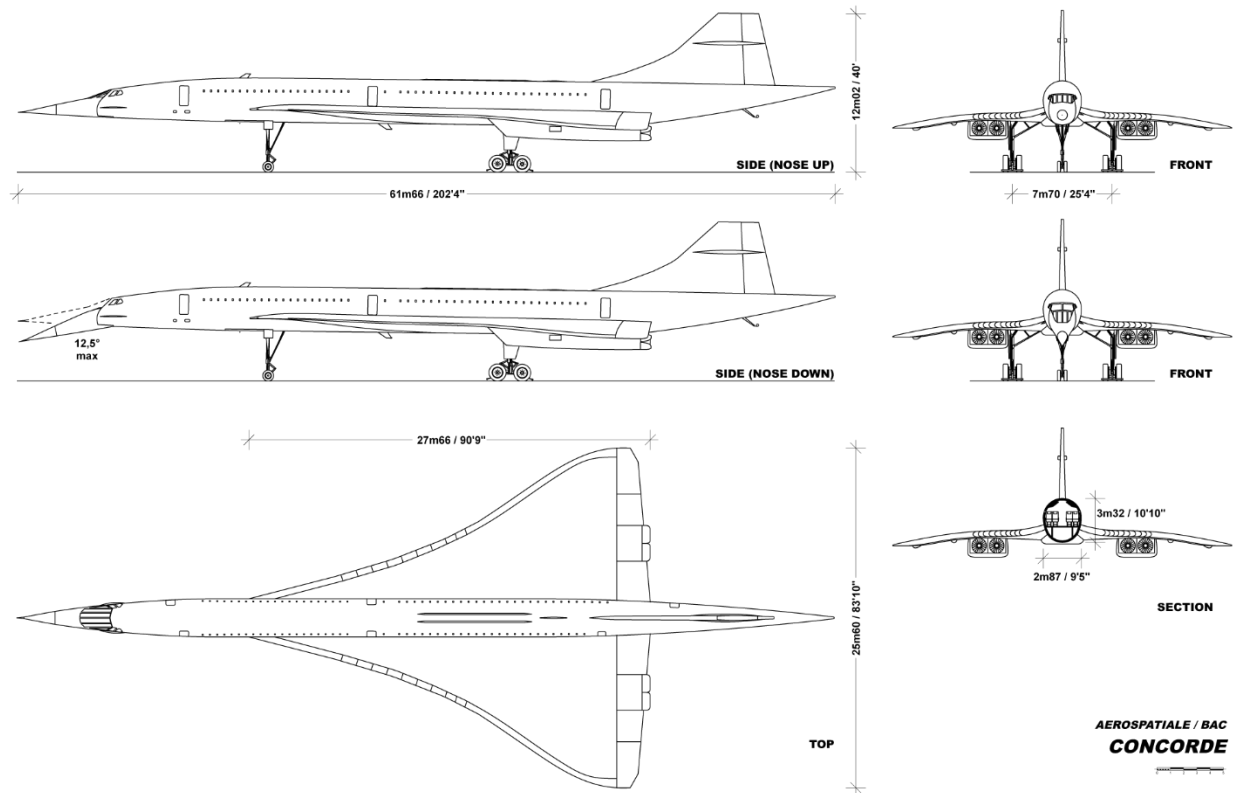


Figure 1.6. Aerospatiale/BAC Concorde [2].

Another important design feature of the Concorde was the overall sizing. The desire to increase the lift-to-drag ratio demanded an extremely slender fuselage, and a large wing. This slender fuselage still had to accommodate standing passengers and the pass-through wing structure, thereby fixing the minimum dimension of the aircraft [12].

The Lockheed-Martin F-22 is an example of a modern aircraft capable of supersonic cruise. In this aircraft however, the ability to cruise at Mach 1.8 without afterburner is due in

large part to the sheer thrust-weight ratio; the dry (non-afterburning) thrust-weight ratio of the aircraft is typically 0.77 [13]. Other features of the aircraft actually compromise the supercruise performance. The engine inlets have fixed geometry to reduce weight, and are sized for subsonic performance; this results in spillage drag during supercruise. The inlet edges are rounded to accommodate this spillage [13].

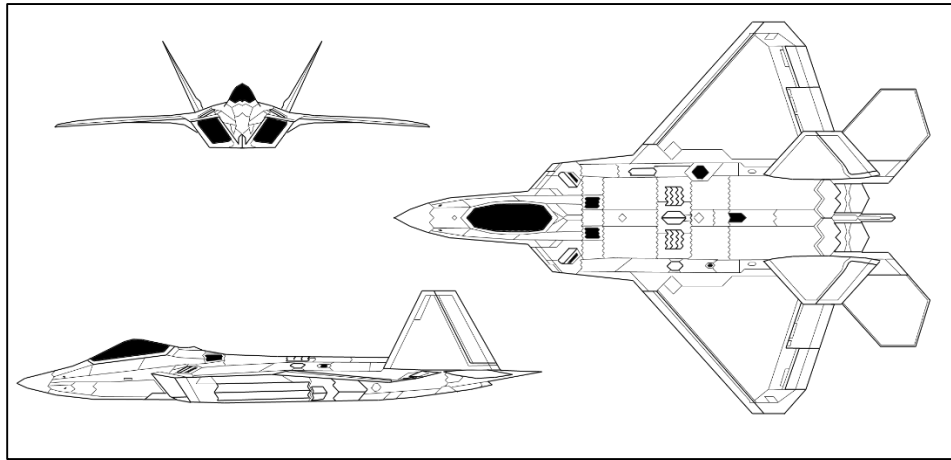


Figure 1.7 Lockheed-Martin F-22 [2].

The SCCT project is part of a renewed interest in supersonic passenger transport since the late 1990s. Several private-sector projects are underway, all in early phases of design. These may be regarded as direct competitors to the SCCT project, targeting very similar business markets.

Chief among these competitor designs is the AS2 aircraft, being developed by Aerion. The current published specifications of this aircraft include a gross takeoff weight of 95,000 pounds, cruise Mach of 1.6 and a three-engine design; the model of the engine is not specified. The key aerodynamic feature is the use of a short wing using a “Natural Laminar Flow” airfoil. This airfoil attempts to reduce skin friction drag by delaying the transition from laminar to turbulent flow on the wing; ideally, only the aft 30% of the wing is in the turbulent regime [14].



Figure 1.8. Aerion AS2 [14].

Another competitor design is the Spike Aerospace S-512. This aircraft is being developed toward a target gross-takeoff weight of 84,000 pounds, a target range of 4000 nautical miles, and a fuselage profile intended to minimize sonic boom [15]. The low weight of this aircraft is driven by the intended use of two Pratt & Whitney JT8D low-bypass turbofan engines. The aerodynamic design of this aircraft appears to be in an early phase; numerous changes to the promotional material have taken place and the aircraft currently features a delta wing. Applying the range analysis featured in Chapter 3 of [4], this design requires a lift-to-drag ratio of 11.4 during supersonic cruise.



Figure 1.9. Spike S-512 [15].

A common feature of many current supersonic initiatives is the reduction in sonic boom noise. This is a feature of the Spike design, and the chief feature of the SAI QSST design. Very little data is published regarding the QSST, but the obvious feature is a complex fuselage profile intended to reduce the strength of shockwaves during supersonic flight.

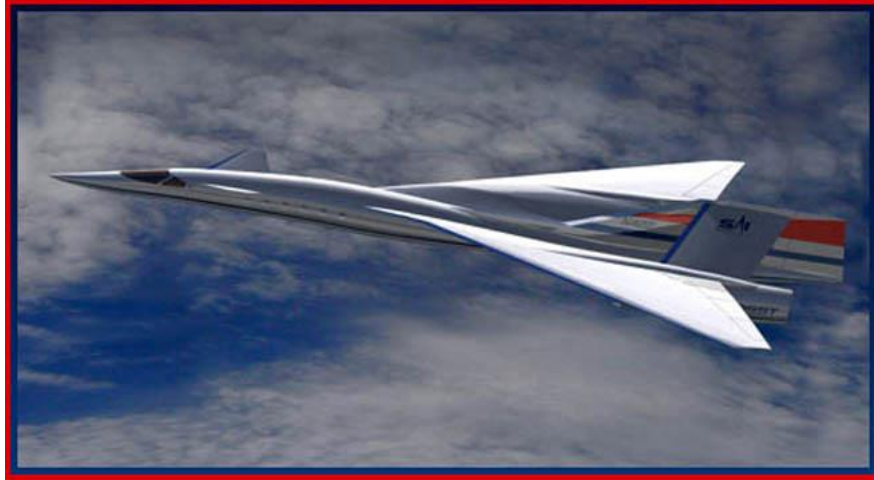


Figure 1.10 SAI QSST [16].

Introduction to Aircraft Design Process

Canonically, the aircraft design process is considered as a process through three phases. The conceptual design process is the first of these. The goal of this phase of design is to arrive at a set of high-level design parameters which satisfy the overall customer requirements of the aircraft. The customer requirements are the mission profile and other constraints established by the customer; this process requires careful analysis and discussion between the customer and engineers, as the requirements often address quantitative customer needs using numerical quantities [3]. The design parameters are those of overall sizing and configuration, such as gross takeoff weight, engine thrust, wing area, wing thickness, taper ratio, and sweep angle. Since conceptual design is the phase considered in this paper, the analysis methods are designed to provide these results at minimum.

The Preliminary design process concerns mid-level parameters of the design, such as wing camber and twist, basic internal packaging, and major structural elements. During this phase, the requirements are taken to be the result parameters of the conceptual phase. For example, a wing structure is designed to support the shape and loading determined in the previous phase.

Detailed design is the final phase of the design process. Continuing in the same fashion of the previous steps, this phase attempts to design detail elements which satisfy the general component requirements laid out in the preliminary stage. Joining of structural elements, specific component locations and mounting, wiring/plumbing, and surface finishing are all finalized at this phase [4].

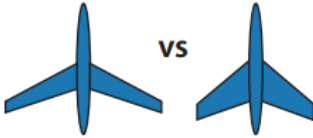
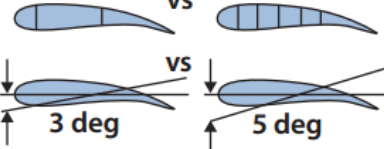
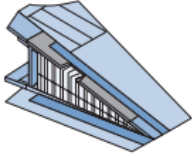
	Phase 1 Conceptual Design	Phase 2 Preliminary Design	Phase 3 Detail Design				
							
Known	Basic Mission Requirements Range, Altitude, & Speed Basic Material Properties σ/ρ E/ρ $\$/lb$	Aeroelastic Requirements Fatigue Requirements Flutter Requirements Overall Strength Requirements	Local Strength Requirements Producibility Functional Requirements				
Results	<table border="1"> <tr> <th>Geometry</th> <th>Design Objectives</th> </tr> <tr> <td>Airfoil Type R t/c λ Δ</td> <td>Drag Level Weight Goals Cost Goals</td> </tr> </table>	Geometry	Design Objectives	Airfoil Type R t/c λ Δ	Drag Level Weight Goals Cost Goals	Basic Internal Arrangement Complete External Configuration <i>Camber & Twist Distribution</i> <i>Local Flow Problems Solved</i> Major Loads, Stresses, Deflections	Detail Design <i>Mechanisms</i> <i>Joint Fittings and Attachments</i> Design Refinements as Result of Testing
Geometry	Design Objectives						
Airfoil Type R t/c λ Δ	Drag Level Weight Goals Cost Goals						
Output	Feasible Design	Mature Design	Shop Designs				
TRL	2 – 3	4 – 5	6 – 7				

Figure 1.11. Aircraft Design Phases [4].

An important aspect of the canonical design process is that it normally incorporates numerous design “freezes”, where a set of design parameters are considered to be fixed for

subsequent design phases. This normally happens between the three phases listed above, but also can occur within a phase. The reason for these freezes are to make the project more tractable; the number of variables present in the overall design project can lead to virtually endless redesign work when a large-scale change is made after detailed design work has been completed.

This action of design freezing can be hazardous to the design process; it necessarily involves truncating a portion of the available design space. Many details are not considered during the conceptual design phase, and may require significant compromises when included at later phases. This problem is typically addressed through experience; an experienced design team is able to predict the important constraints for a given project. We attempt to improve this prediction by including actual analysis germane to these constraints, and indeed all constraints to which the design is subjected.

Any design process requires some schematic concept of the system being designed, and this is also true for aircraft. Initially, this representation is simply the geometric form of the vehicle; considering the exterior shape and internal arrangement. Many different techniques are used, which vary dramatically in the detail of the representation as well as the time required to draft the design. In each case, a number of variables are assigned to the geometry which are later used to analyze the performance. For example, in the simplest case the aircraft can be represented as a single point. The point takes on variables like mass and moment of inertia, and can undergo simple kinematic analysis based on these properties.

One representation commonly used during the earliest phases of aircraft design is the “stick airplane”. This model extends the point along one dimension, effectively modeling the vehicle as a line segment. The various components of the aircraft can then be placed at points along this line, and the overall properties of the vehicle are derived from the sum of the component contributions. An example of this representation is shown in Figure 1.11. The image can be considered as an aircraft viewed from the port side, with the front facing left. Four components are considered; each is placed at a location along the aircraft length and provides some contribution to force, mass or lift. This analysis can consider both the type and properties of each component individually, as well as their placement in the overall system. This typically

allows the estimation of performance quantities such as pitch stability and rate, and rate of climb [17].

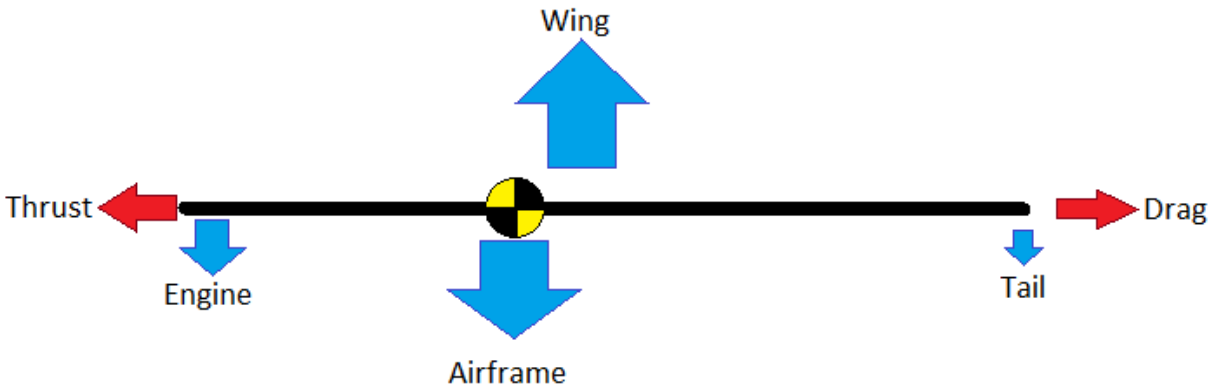


Figure 1.12. Single-Dimension Stick Airplane.

The stick airplane suffers an important drawback, which is that the performance of each component is considered individually. In reality, the performance of all the aircraft components behave in a coupled fashion: the engine thrust affects the flow over the wing, and main wing creates a downwash on the tail. In order to begin to address these couplings, the panel airplane can be used, which begins to represent the aircraft geometry in three dimensions. The exterior geometry of the aircraft is represented as a collection of two-dimensional plane elements, located and oriented in three-dimensional space; a panel airplane is shown in Figure 1.12. This allows the planform shape to be quantified, as well as the three-dimensional mass properties of the vehicle. With this representation, it is possible to estimate the lift distribution, and stability about three axes. It is also possible to begin to analyze the properties of the internal structure, by modeling the vehicle as a collection of simple beams (i.e. the wing and tail spars connected to a central fuselage) [4][18].

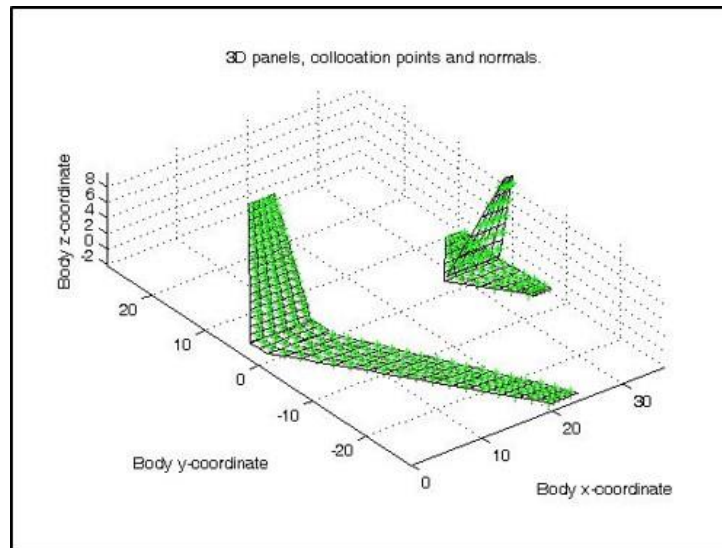


Figure 1.13. Panel Aircraft representation. [19]

At the current apex of geometric representations, 3D CAD (Computer Aided Drafting) tools are used. These methods allow the complete aircraft to be represented in arbitrary detail, using free-form solid volume geometry. The volume of the aircraft components are represented as a set of analytical solids in three-dimensional space. This format is extremely flexible, and introduces many variables to define the form of the solid geometry. With this technique, an analysis of the internal structure and volume is possible, allowing the packaging arrangement of internal components to be considered. CAD is the starting point for many of the most advanced engineering analyses performed today, and also forms the basis of modern manufacturing techniques [20]. A transport aircraft is represented in CAD in Figure 1.13, and shows a typical level of detail used in the conceptual design phase.

Each of the geometric representations described above allow a set of performance analyses to be performed. Some approximation is made of the fluid flow around the body while it is in flight to analyze the aerodynamic performance of a test geometry. The methods used to accomplish this also vary dramatically in cost and detail. The detail of an approach can further be split into its accuracy and precision; an accurate analysis produces answers which closely match the real performance, and a precise analysis is reliably sensitive to small changes in the model variables. It is important to balance these features when creating an analysis suite for a project.

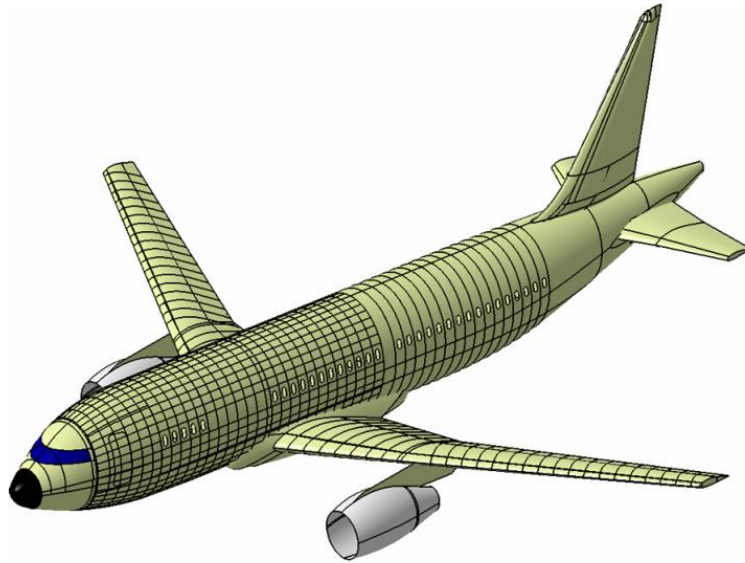


Figure 1.14. 3D CAD Aircraft Exterior.

For the simplest representations, an analytical performance factor called the Oswald efficiency is typically used. This factor is simply a property of any lifting-surface component, and is applied to a point or stick airplane to characterize the three-dimensional shape of the finite wing. Applying this factor is an analytic result, so numerical computation is straightforward; however, the value of the Oswald factor itself is typically an empirical result [21].

Since the 1960s, computational tools have been used to eliminate this experimental dependence and calculate the performance of a design directly. One of the most common tools is the Vortex Lattice method, which is applied to a panel geometry. This approach uses a linear superposition of many analytical solutions of varying strength, with the overall sum constrained to satisfy the body boundary conditions. In order to combine such solutions, they must be linear; this restricts the flow regimes which can be applied to inviscid and irrotational flows. Vortex lattice codes are still widely used for modern design; a popular implementation is Tornado for MATLAB [19]. Computationally, these methods typically require the explicit inversion of a large square matrix.

Modern computational fluid dynamics (CFD) is performed using Finite Volume methods. These methods directly discretize the Navier-Stokes equations on a mesh of volume cells. With

well-posed boundary conditions the problem is tractable to solve numerically. This approach typically uses an implicit numerical solver and requires a high-resolution mesh; the solution therefore requires significantly longer computation time than previous methods. However, when used properly the accuracy and precision of the computation can be arbitrarily chosen by the designer. Many free, commercial and research-grade are currently maintained which implement finite-volume CFD solvers.

Turbulent boundary layers pose a challenge worth mentioning in modern CFD applications, and they strongly affect the flow regimes under consideration in this project. Since the underlying physics of turbulence are not well understood, turbulence simulation typically involves an empirically-based model such as k-epsilon or k-omega [22]. Direct numerical simulation is possible, but it remains cost-prohibitive for high Reynolds number flow and complex shapes [23].

Other analysis tools which are important to aircraft design are those which address the analysis approach used in the project. Since a number of analyses must take place in order to evaluate a design candidate, the type and order of the analysis process can also be studied. Thus, the process of designing an aircraft vehicle also includes the design of the process used to create the vehicle design. Many tools for performing this meta-analysis are provided by Systems engineering, but a more rigorous approach is given by a field called Multidisciplinary Design Optimization (MDO). MDO attempts to formulate a complete design project as a single problem, and in such a way that a true optimization can be performed. Ideally, the project is written as a one-to-one objective function of the design variables, such that minimizing the function results in an optimal design candidate. This approach is an area of active research, particularly by J. Alonso at Stanford University [24]. Our approach differs from true MDO, however; we attempt to use some basic techniques to achieve a fast and inexpensive initial evaluation of a project. Rather than an exhaustive or custom-tailored optimization approach which results in a complete design, we use versatile commercial codes which are automated to provide only the necessary information to assess design concepts against a set profile.

The Design Structure Matrix is a tool used to optimize the analysis itself. Analysis blocks are called Subsystems, and are listed on the main diagonal of a square matrix. The information

flow between subsystems is shown by a binary value in off-diagonal entries, called feed-forward or feed-back. Figure 1.14 shows how the DSM format translates to typical flow-chart diagrams.

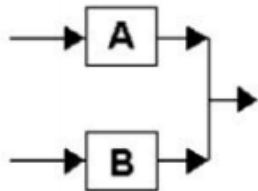

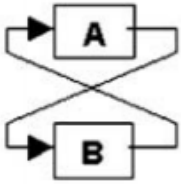
Three Configurations that Characterize a System																														
Relationship	Parallel	Sequential	Coupled																											
Graph Representation																														
DSM Representation	<table border="1" data-bbox="516 737 755 976"> <tr><td></td><td>A</td><td>B</td></tr> <tr><td>A</td><td>■</td><td></td></tr> <tr><td>B</td><td></td><td>■</td></tr> </table>		A	B	A	■		B		■	<table border="1" data-bbox="836 737 1075 976"> <tr><td></td><td>A</td><td>B</td></tr> <tr><td>A</td><td>■</td><td>X</td></tr> <tr><td>B</td><td></td><td>■</td></tr> </table>		A	B	A	■	X	B		■	<table border="1" data-bbox="1156 737 1395 976"> <tr><td></td><td>A</td><td>B</td></tr> <tr><td>A</td><td>■</td><td>X</td></tr> <tr><td>B</td><td>X</td><td>■</td></tr> </table>		A	B	A	■	X	B	X	■
	A	B																												
A	■																													
B		■																												
	A	B																												
A	■	X																												
B		■																												
	A	B																												
A	■	X																												
B	X	■																												

Figure 1.15. Design Structure Matrix terminology and format [25].

The DSM highlights the information flow between portions of the analysis. Coupled portions of the loop require iteration to converge on the output values, so it is useful to attempt to arrange the analysis subsystems to minimize the amount of iteration required. [25][26].

Once an analysis scheme has been optimized, the design of the actual project at hand can take place. Heuristic methods for this optimization are often required for multidisciplinary analysis of large systems, due to the difficulty of calculating the gradient of the objective function with respect to the design variables. The simplest heuristic method is the use of a global random search, in which many design candidates are generated, with design variables taking on random values. Each candidate is analyzed using the same scheme, and the best performing candidate is taken to be the optimum. This approach requires large populations of candidates to be evaluated in order to be successful. An improvement on the simple random search can be provided by a Particle Swarm Optimization, in which a smaller random-sample population is

evolved over several iterations, with each iteration converging toward the population best performer; some randomization is included at each step to avoid local minima [27][28].

Another line of research into analysis optimization involves the use of lower-fidelity analysis tools to predict the best-performing design candidates. This technique is called Surrogate-Based Optimization; its aim is to reduce the computation cost of evaluating large populations. The goal of the analysis tools used here is not to predict exactly the outright performance of each model, but rather to predict the rank of performance of the candidates relative to each other. The top performers can then be evaluated using a small number of high-cost evaluations to obtain the true optimal performance. [24].

It is our claim that the best approach to the design of aircraft balances these goals. By carefully tuning the fidelity of the analysis in the conceptual phase, a more robust optimal design can be delivered by including important preliminary design factors while retaining a low cost of analysis. Using the case study of a supersonic commercial transport, we showcase an analysis technique using commercial software tools, which are automated to provide a high-fidelity analysis at low computation cost. While

CHAPTER II

METHODOLOGY: HIGH-FIDELITY CONCEPTUAL DESIGN APPROACH

Computer-Aided Drafting using Solidworks

A general discussion of CAD is beyond the scope of this thesis, however our problem does require some specific methods and techniques to provide a well-posed problem. The purpose of CAD in our approach is to take a set of real-valued scalar dimensions and generate a solid 3D model file, to be read by subsequent analysis steps. Since the CAD defines the geometry to be analyzed, there must be a one-to-one correspondence between the dimensions and models, so each given set of dimensions results in exactly one unique 3D model. The following steps describe a generalized CAD process, and specific actions which were taken in order to preserve this correspondence. We use Solidworks to implement this process, but the method is easily adapted to other packages.

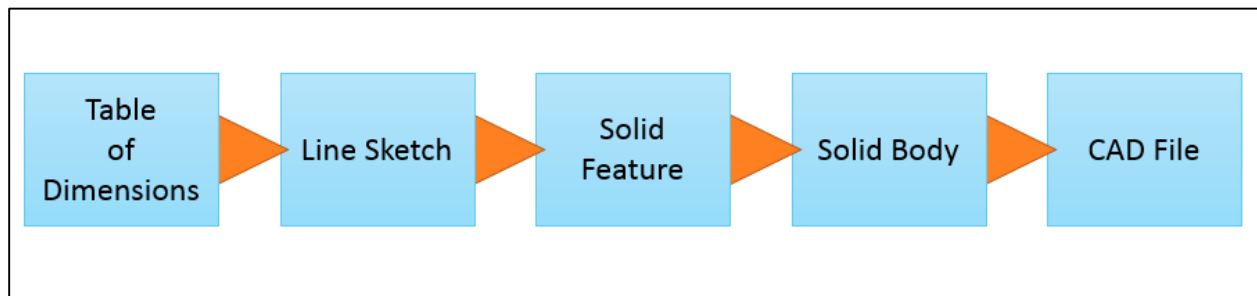


Figure 2.1: CAD Construction Process.

The schematic shown in Figure 2.1 shows the general flow of the construction of the CAD file. The process begins with a table of real-valued scalar dimensions, which define sketches composed of analytical curve segments such as lines, circular arcs, and/or cubic splines. These sketch segments are then used to create or cut solid features using operations including extrusion, revolution, lofting, and sweeping. These solid features combine to make the final geometric bodies which are saved to the CAD files. We refer to the complete system shown in Figure 2.1 as a CAD “deck”; once this deck is constructed, the initial table can be modified to

generate new models automatically. In fact, this method allows the generated models to be arbitrary in both individual geometry as well as total number. Thus, it is often referred to as Parametric CAD, Rubber CAD, or iMod[29].

Building the CAD deck requires some *a priori* knowledge of the desired final product, so some forethought is necessary. Initial considerations must be made of the desired output coordinate system; we use the axis directions defined in [17]. This system defines X pointing longitudinally along the nose of the airplane, Y pointing horizontally in the starboard direction, and Z pointing down. More generally, before building a CAD deck one must consider the amount of flexibility desired in the model. Extreme flexibility is possible; for example, a single CAD deck can create models of conventional, biplane or aft-wing aircraft, with several different engine arrangements, as well as geometric variations of the dimensions of each. However, such configuration options add complexity to the deck and it therefore requires more time to construct and debug.

Building the CAD deck begins with creating line elements which are expanded to form the 3D bodies. The sketching process is critical to the creation of a one-to-one CAD model, since this is the step where the dimensions are applied to the geometry. As new sketches are created, they are defined by dimensions; these then populate the table which is used to control the geometry. In addition to numerical dimensions which are applied, logical relations can be established between elements to define their position. For example, a line can be defined as parallel to the X-axis with one endpoint at the origin. Now, the line can be fully defined by its length and direction (+x or -x).

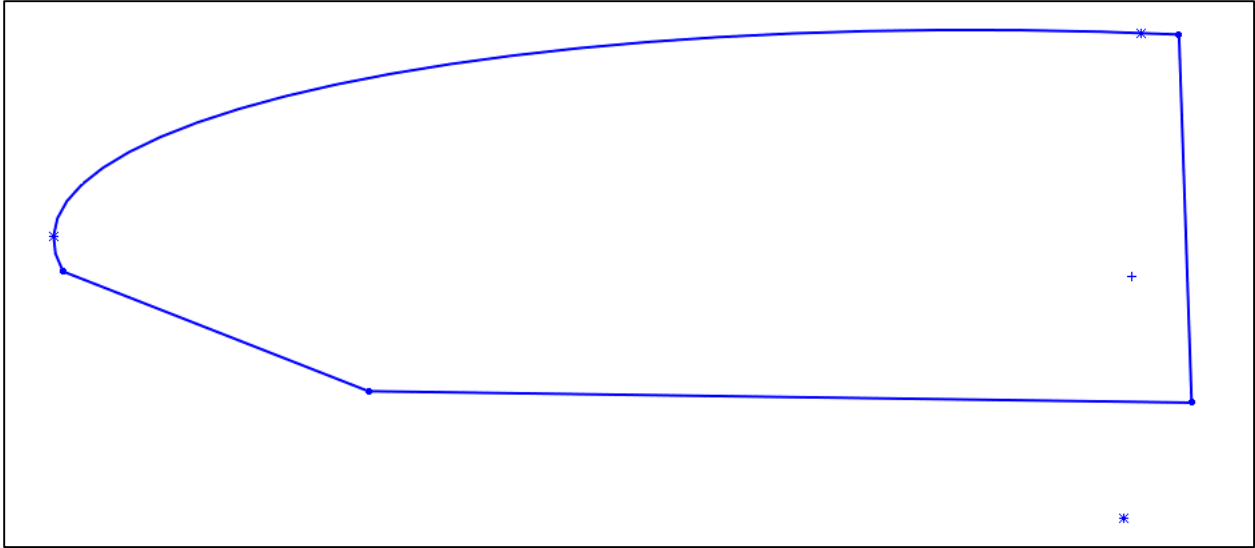


Figure 2.2: Un-dimensioned sketch.

Figure 2.2 shows a sketch which contains neither dimensions nor relations. It simply consists of four basic sketch entities: three straight lines, and a portion of an ellipse. It is a closed sketch and a 3D model could be generated from it, but it is not useful for the problem at hand, since any applied dimensions will produce the same output.

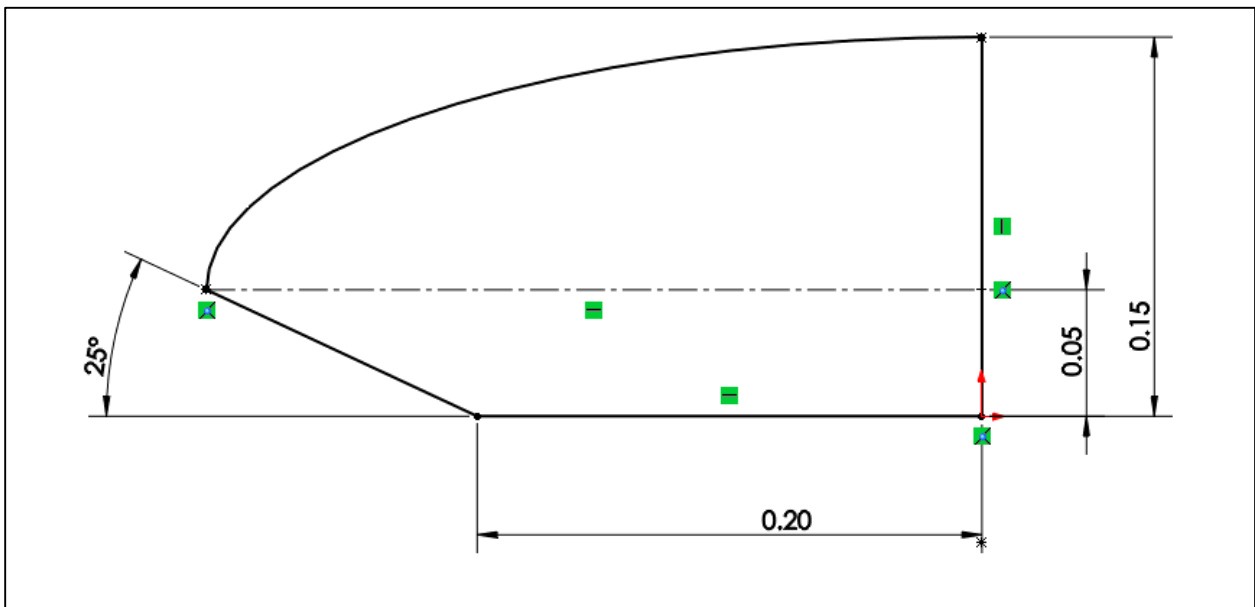


Figure 2.3: Fully Defined sketch.

In Figure 2.3 the sketch has been fully defined using a combination of relations and dimensions. The dimensions are shown using drafting notation, and the relations are identified by green icons. The lower right corner has been defined as coincident with the origin, fixing its location. The two lines adjacent to this corner have been assigned horizontal and vertical relations, as well as distance dimensions. The ellipse has been defined by constraining its center to the vertical line, at a fixed distance from the origin. A horizontal construction line forms the semimajor axis of the ellipse, and defines the endpoint of the angled line. The degenerate cases of a distance pointing the opposite direction are handled behind-the-scenes in Solidworks by signing each created dimension [30]. Applying a negative dimension to a distance or angle will flip the direction and take a value equal to the absolute value applied.

The sketch is now in a state such that any combination of the four applied dimensions will yield exactly one unique result. Solidworks uses the term “Fully Defined” to identify this state. When a sketch is finished, the range of desired movement should be considered, to ensure that the sketch can obtain the range without logical errors. For example, an angle which must be able to take on both positive and negative values should be dimensioned with an added 90 degrees, as shown in Figure 2.4.

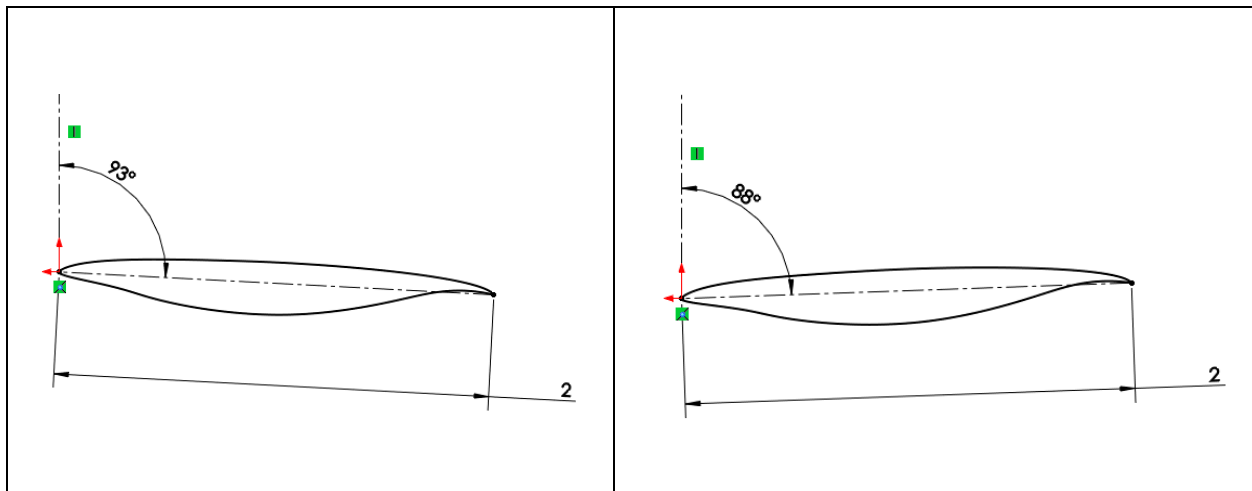


Figure 2.4: Angle of Incidence dimension for airfoil showing +3 and -2 degrees.

The techniques used in sketching and dimensioning the model determine the number of variables required to construct the model; this contributes significantly to the scale of the

analysis as a whole. Since the goal of the analysis is to study the cross-coupling of geometric shape with respect to performance output, the number of dimensions used in sketching adds directly to the dimensionality of this cross-coupling [31]. Figure 2.5 shows an example of this case. Two methods of constructing the desired cross-section are shown; note that the sketch to the left uses arc and ellipse curves and is defined with only three dimensions; the sketch on the right uses cubic spline entities and requires a total of nine dimensions. The shape on the left gives much more flexibility in terms of geometric shape, but this comes at the expense of threefold greater complexity.

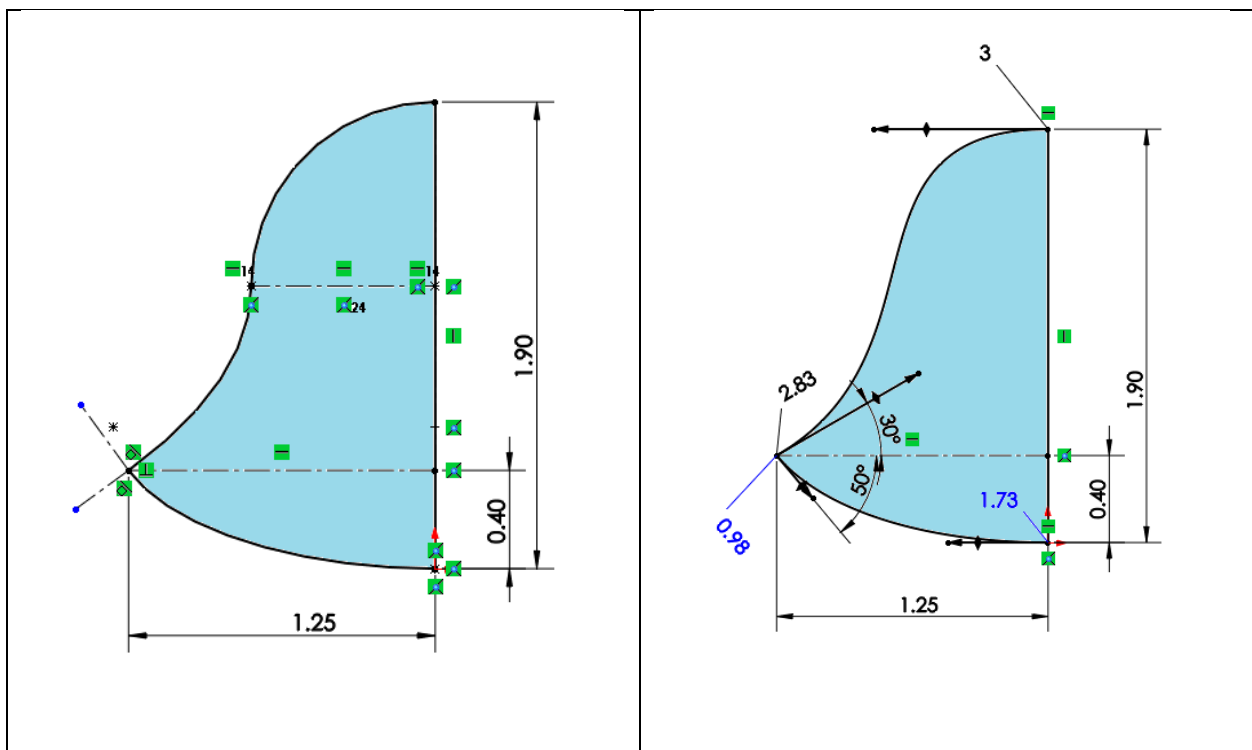


Figure 2.5: Comparison of Sketch Construction; 3 dimensions vs. 9 dimensions.

Once a sketch construction was obtained, the solid model can be built. This is performed using common CAD operations including extrusion, revolution, and lofting. Our design methods do not constrain the type of features used to construct the solid model; however certain features add dimensions to the design. For example, the profile shown in Figure 2.5 can be extruded

normal to the plane to generate a solid volume. The distance of this extrusion then adds a dimension to the analysis. The following images shown in Figure 2.6, show the process of lofting a wing onto our design. The progression begins with the upper left image and proceeds clockwise. First, the sketches which will compose the loft are shown. The wing is composed of three airfoil sections, along with guide curves corresponding to the leading and trailing edges. The loft is performed as two separate features, with the inboard section lofted first followed by the outboard section. The final image shows the completed solid body of the wing.

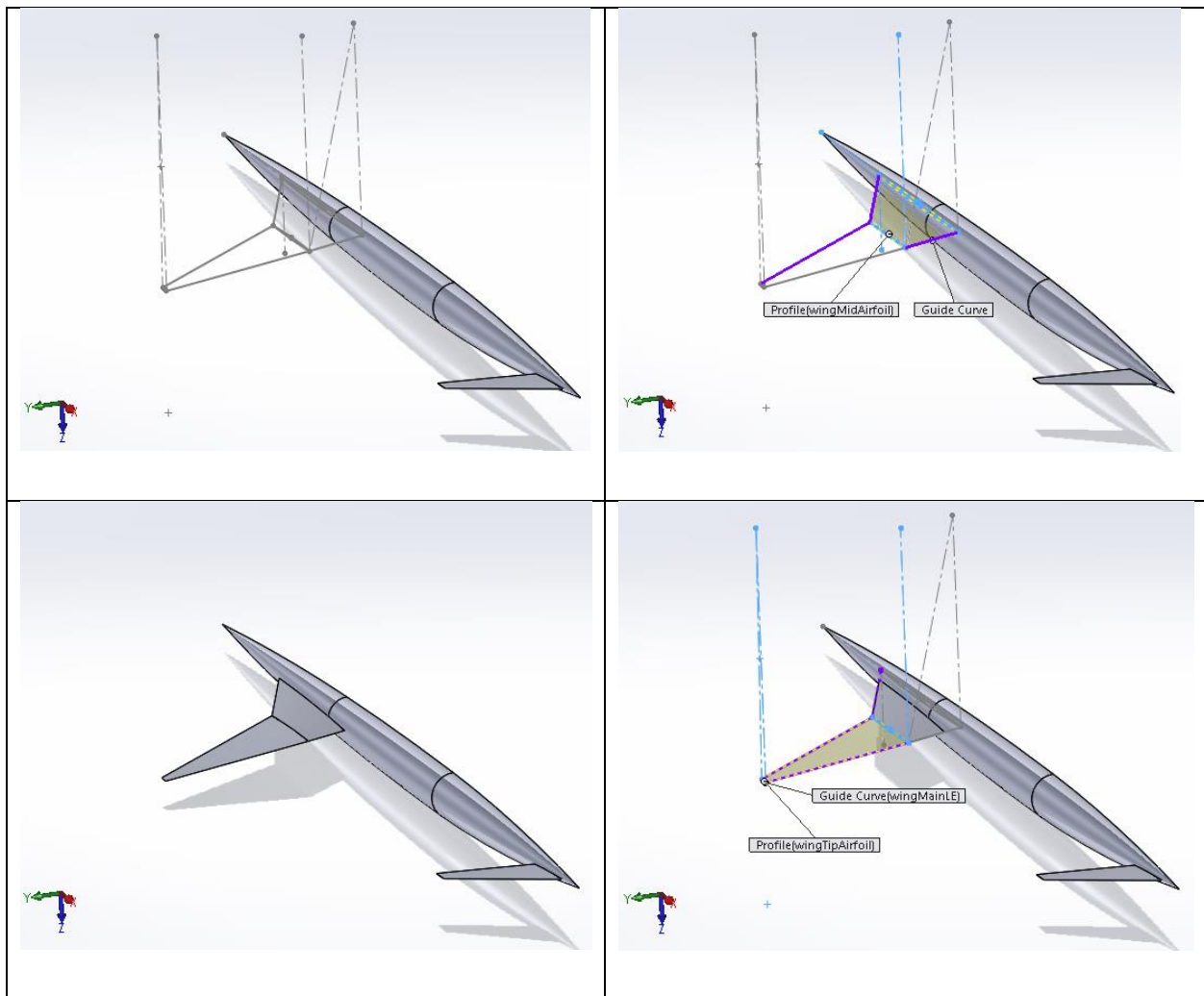


Figure 2.6: CAD Wing Loft Progression.

Note that a portion of the wing volume intersects the fuselage volume already present in the images. One advantage of solid body modeling techniques is that the features can be merged,

which creates a single model from the union of the new feature and the existing solid. This automatically creates a manifold solid with the correct intersection edge.

One section of our design is constructed using imported geometry; namely the engine nacelle, and inlet/nozzle flowpaths. Since the specific engine is specified by the mission profile, we elected to use a fixed geometry for these elements which is imported and placed at the desired location. The inlet geometry was created using the SUPIN code developed by John W. Slater at NASA Glenn Research Center [32][33]; the code takes inputs of Mach number, mass flow rate and fan diameter; it then explicitly generates the geometry of the engine inlet. The generated geometry is shown below in Figure 2.7.

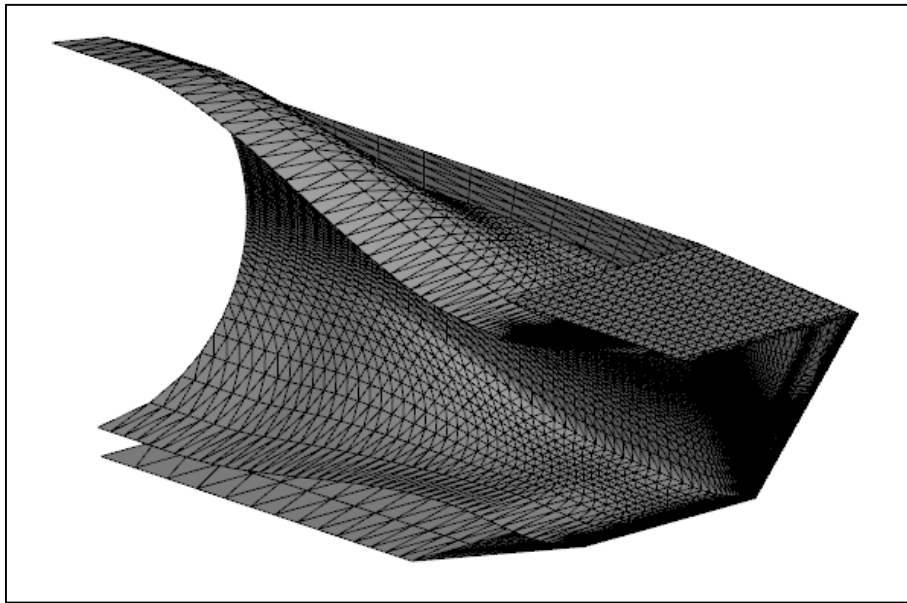


Figure 2.7: Inlet geometry generated by NASA SUPIN.

The geometry output from SUPIN was then wrapped in a best-fit cubic surface within Solidworks to create the final engine CAD. At the other end of the engine, the nozzle geometry was created in Solidworks to match the F-100 inlet under cruise conditions. The usage of the dedicated inlet and nozzle flowpaths is explained in greater detail in Section 8. These three

geometry shapes were saved to a separate Parasolid file, which was imported to the CAD deck and placed on the model.

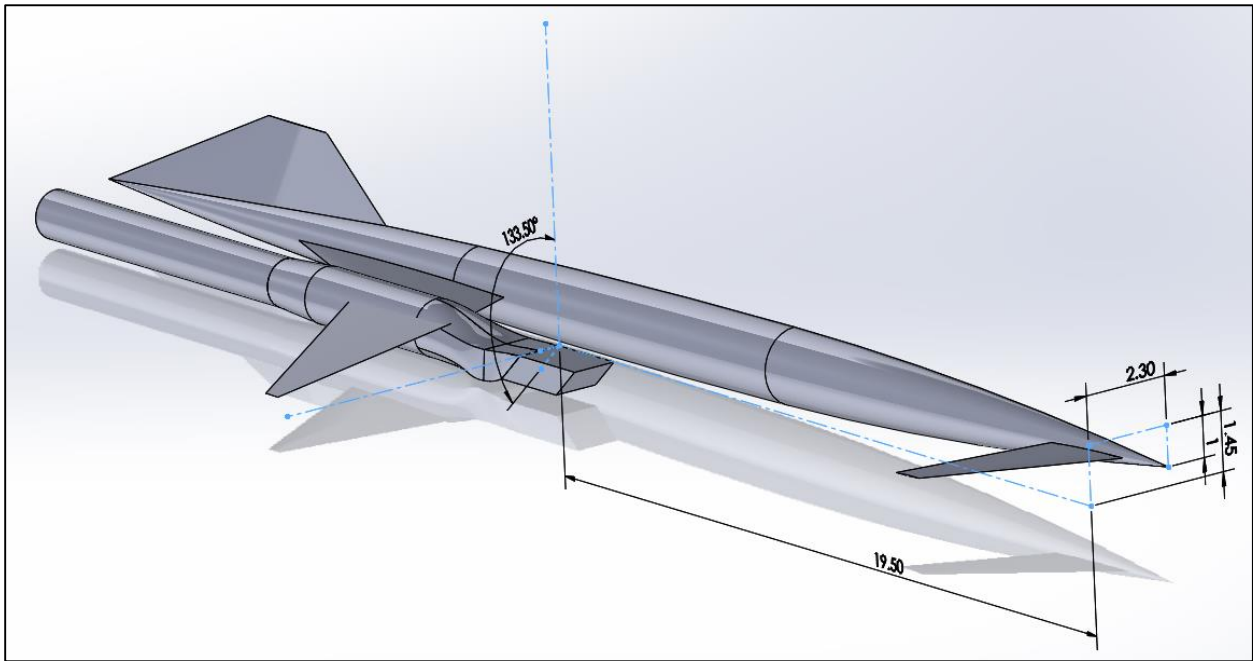


Figure 2.8: CAD Placement of Imported Engine Geometry.

After importing the engine geometry to the CAD model, it must be placed in such a way that dimensions can be applied. This is accomplished by choosing three specific points on the engine geometry, and locating them in x-y-z space within the model. A construction sketch is used to place three points in space with the desired relationship, and the corresponding points on the imported engine geometry are fixed to these sketch points. This is shown in Figure 2.8; the sketch uniquely locates the inboard corners of the inlet mouth as well the top centerline. After placing the engine, additional CAD modeling is used within the deck to provide volume for structural support, such as a pylon or a bulge in the wing surface.

The final CAD operation is to create a negative fluid volume. This converts the aircraft geometry into the geometry of the fluid surrounding the aircraft, which is one of the intended analyses to be performed. This is easily performed using solid modeling techniques by creating a large volume and performing a subtraction operation of the original shapes. This final volume is shown in Figure 2.9; the small collection of hidden lines in the center show the location of the

aircraft. A conical fluid domain has been constructed as the intended analysis regime is supersonic in nature. The fluid domain is also truncated at the centerline plane of the aircraft (i.e. the z-x plane) to apply a symmetry assumption to the simulation.

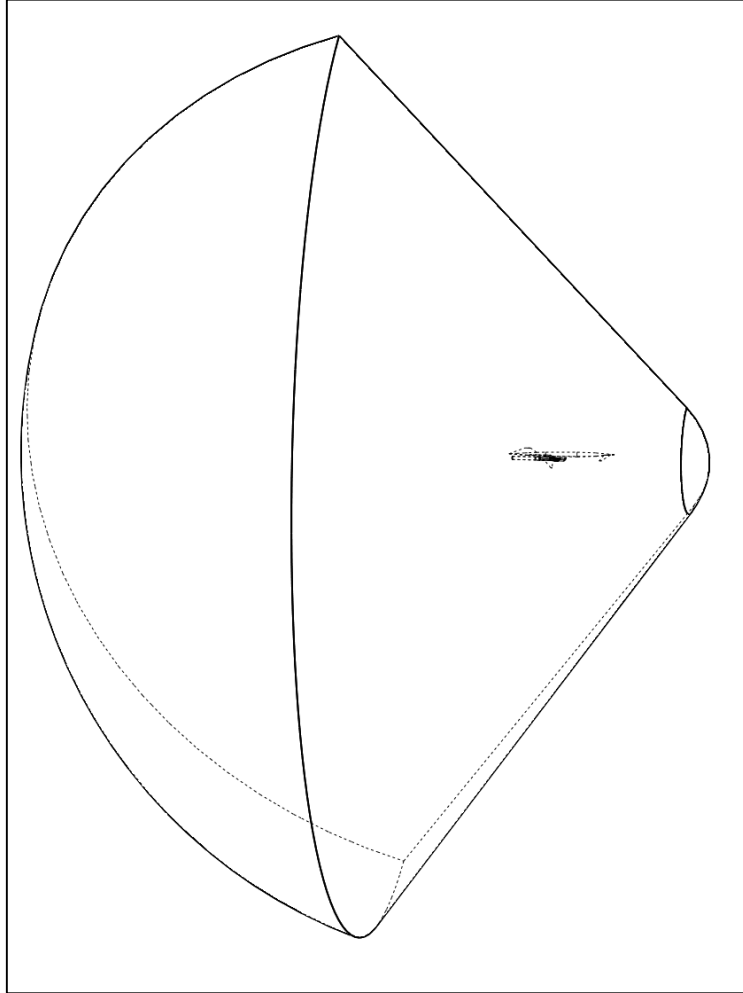


Figure 2.9: Complete Fluid Domain CAD.

The crucial feature of our CAD deck is the table of dimensions which drives the model. In Solidworks, this is called a Design Table and features all the model dimensions as columns, with different model configurations as rows. An excerpt of the upper-left portion of a design table is shown below in Figure 2.10.

Design Table for: Nutmeg3																			
	Fuselage Dimensions (Red)				Fuselage Dimensions (Red)				Fuselage Profile (Red)	Wing Dimensions (Green)			Wing Dimensions (Green)						
Models	Camber@FusDims	NoseL@FusDims	CabinL@FusDims	TailL@FusDims	AftHeight@FusDims	ForeHeight@FusDims	Offset1@FusDims	NoseT@FusDims	Offset2@FusDims	TailT@FusDims	D1@FusProfile	Span@WingLE	Sweep@WingLE	Dihedral@WingLE	AOA@RootAirfoil	Chord@RootAirfoil	TC@RootAirfoil	Chord@MidAirfoil	TC@MidAirfoil
Default	345	15	9	15	1.7	1.7	0.01	0.02	0.01	0.02	0.1	6	85	88	92	5	2	4	1.5
min	300	12	9	12	1.7	1.7	0.01	0.02	0.01	0.02	0.1	4.5	85	88	91	4.5	2	4	1.5
max	450	15	9	15	1.7	1.7	0.01	0.02	0.01	0.02	0.1	7	85	92	94	5.5	2	4	1.5
1	316	14	9	12	1.7	1.7	0.01	0.02	0.01	0.02	0.1	5.2	85	92	92	5.5	2	3.6	1.5
2	400	14	9	12	1.7	1.7	0.01	0.02	0.01	0.02	0.1	6.1	85	92	91	5.1	2.4	3.4	1.5
3	403	14	9	13	1.7	1.7	0.01	0.02	0.01	0.02	0.1	4.9	85	90	93	5.1	2.4	3.2	1.5
4	330	14	9	14	1.7	1.7	0.01	0.02	0.01	0.02	0.1	5	85	91	91	5.3	2.2	3.5	1.5
5	380	13	9	13	1.7	1.7	0.01	0.02	0.01	0.02	0.1	5.8	85	88	93	5.3	2.2	3.4	1.5
6	433	15	9	14	1.7	1.7	0.01	0.02	0.01	0.02	0.1	6.2	85	91	93	4.7	2.4	3.3	1.5
7	322	13	9	13	1.7	1.7	0.01	0.02	0	0.02	0.1	5.6	85	90	94	5.3	2.2	3.5	1.5
	326	14	9	13	1.7	1.7	0.01	0.02	0	0.02	0.1	5.5	85	90	94	5.2	2.2	3	1.5

Figure 2.10: Design Table Excerpt.

We color the columns of the design table by the part of the aircraft they refer to; in this example the red columns refer to fuselage dimensions and the green columns refer to the wing. The rows of the table are populated by the design models which are to be analyzed. We create these rows procedurally by assigning each dimension a generated value; this is done in three ways.

The simplest method to generate model variation direct randomization of the dimensions. We accomplish this by setting fixed minimum and maximum limits; these are seen in the “min” and “max” rows of the table above. The numbered models which make up the remainder of the table are then generated by taking a random value between these limits. Direct randomization has the advantage of a thorough exploration of the design space; since all dimensions vary independently with respect to all other dimensions, it should be possible to detect all cross-coupling effects between the models. Such an exhaustive process is very time-consuming, however. In practice, the dimensions are not entirely independent, so some small engineering

analyses can be performed which correlate several variables. The controls applied to accomplish this reduction are discussed in detail in the final section of this chapter.

Finally, some dimensions are simply held constant. This is performed due to known conditions of the model, derived from the initial mission goals. For example, since our design should seat 20 passengers with 0.9-meter seat pitch, the passenger cabin aisle must be 9m in length. The text color-coding shown in Figure 2.10 shows the method used to generate the model variations; white dimensions are directly randomized, black dimensions are held constant, and grey columns are calculated from other values.

We determine the total number of models to be analyzed based on the number of randomized variables used, and the method used to choose a final design. When using a simple global random search, a single group of models is created and evaluated, and with the highest-performing model selected as the design to use. For this approach, we use 30 models per variable. When using an iterative optimization approach, we use 10 models per variable. Our CAD decks typically used 30-40 randomized variables.

Once the models are generated, they must be exported and debugged. The export process simply saves each model to a Parasolid Binary file, a compact format which contains only the solid geometry with very little metadata. We automate this process using a Solidworks macro written in VBA. The macro simply loops through all the models (termed Configuration in Solidworks) and executes a forced rebuild of the CAD geometry using the table dimensions, and then saves the output file. The source code of this macro is shown in Appendix B.

Ideally, this is the end of the CAD process and the models can now be analyzed. However, there are frequently issues with model generation which must be resolved before the CAD deck is finished. The usual source of these errors are models which failed to rebuild due to some error in the CAD generation, frequently due to a logical flaw in one or more solid features. These become more common using complex constructions and spline curves, due to the increased degrees of freedom. An example of a failed model is shown in Figure 2.11, where a wing loft operation has failed.

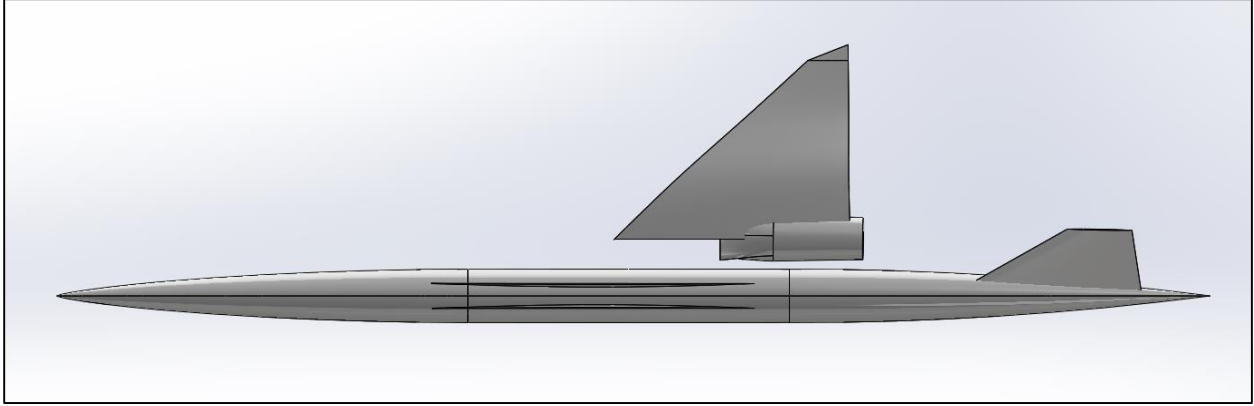


Figure 2.11: CAD Rebuild Failure.

The simplest way to resolve these errors is to limit the range of the dimensions in the design table, but we attempted to avoid this method where possible by more careful application of dimensions and relations within the part. The reason for this preference is that a reduction in the range of dimensions effectively truncates the design space represented in the analysis, therefore compromising the robustness of the final result.

Aerodynamic Analysis Using STAR-CCM+

We performed an analysis in STAR-CCM+ to evaluate the performance of our design candidates during the cruise phase of flight. The overall purpose of our STAR-CCM+ analysis is to open the given set of candidate CAD geometries and perform an identical analysis upon each. The goal is to create a fair benchmark, allowing the design candidates to be ranked based on performance calculations using the results obtained in STAR-CCM+. In order to accomplish the consistency required for such a benchmark, we created a fully automated analysis using macro code written in JAVA. The complete analysis process is automated, and proceeds through the steps shown in Figure 2.12 below.

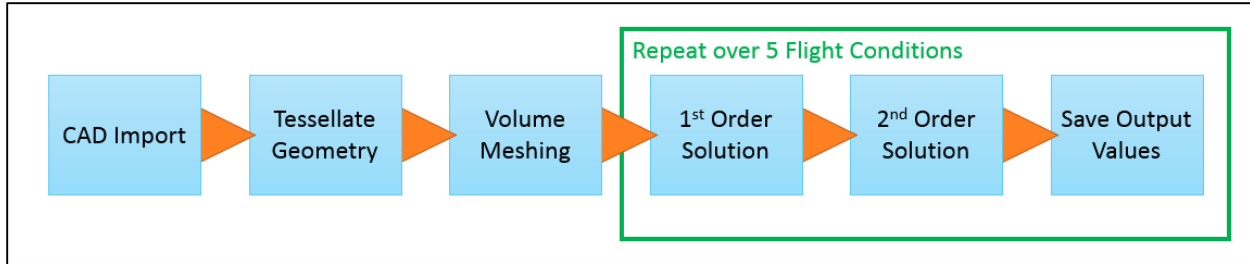


Figure 2.12. STAR-CCM+ Evaluation Workflow Schematic.

The topology of the simulation is shown schematically in Figure 2.13. In the main fluid region, supersonic flow enters the conical boundary from the left (1) and passes over the walls of the vehicle (3). The downstream boundary is a pressure-release boundary (2). The angle of the cone was determined by the Mach angle at the evaluated flow velocity, plus five degrees to account for the pitch angle changes.

The engine installation adds two smaller fluid domains which contact the main region at (6) and (7). Separate regions are used in order to provide additional control over the grid generation process. Within the engine flowpath, the fan face boundary (4) is a pressure-release outlet; the turbine boundary is a mass flow inlet with specified total temperature. Most of the evaluations performed involve pitch angles only, and so bilateral symmetry is used.

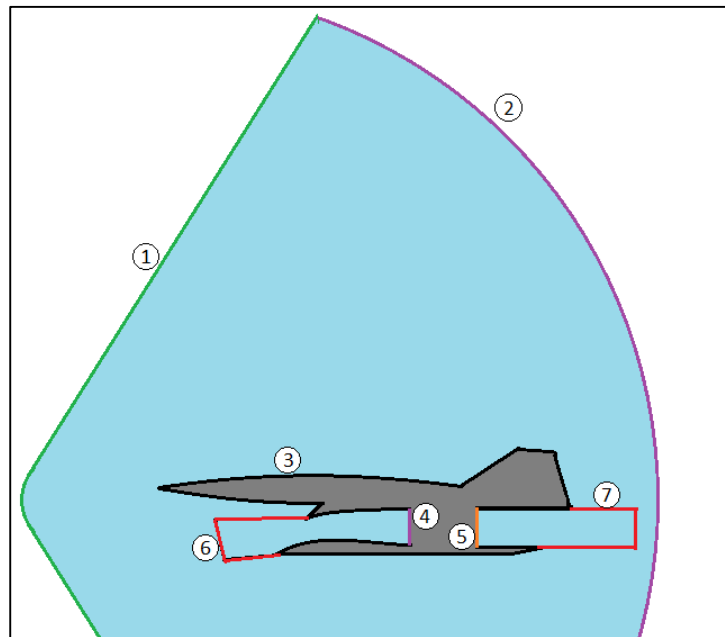


Figure 2.13. STAR-CCM+ Simulation Topology Schematic.

The surfaces of the CAD geometry must be named to apply these boundary conditions within STAR-CCM+. This capability is a feature of the Parasolid file format, and essentially tags the faces of the CAD part so that they can be called in the CFD phase. We chose names that were

human-readable and applied them to the imported part surfaces in Solidworks; when the fluid domain is created the only un-named faces are those belonging to the vehicle itself; these boundaries are automatically collected and the appropriate wall conditions are applied.

The first step in the evaluation process is to import the CAD geometry saved by Solidworks and create a finite volume mesh on which to perform the solution. This is a multi-step process in STAR-CCM+, with each phase generating a Representation. The CAD import results in an “Initial” representation, and the code then proceeds to generate Geometry, Surface Mesh, and Volume Mesh representations; the last of these is finally used for the finite volume solver.

The initial representation is an analytic CAD body. The first step in the mesh process is to tessellate this body into planar surface elements, creating the Part mesh. This discretization is purely curvature-based and incorporates no element sizing; the goal is merely to capture the geometry accurately. This grid is shown in Figure 2.14, highlighting the area of the engine inlet and wing root.

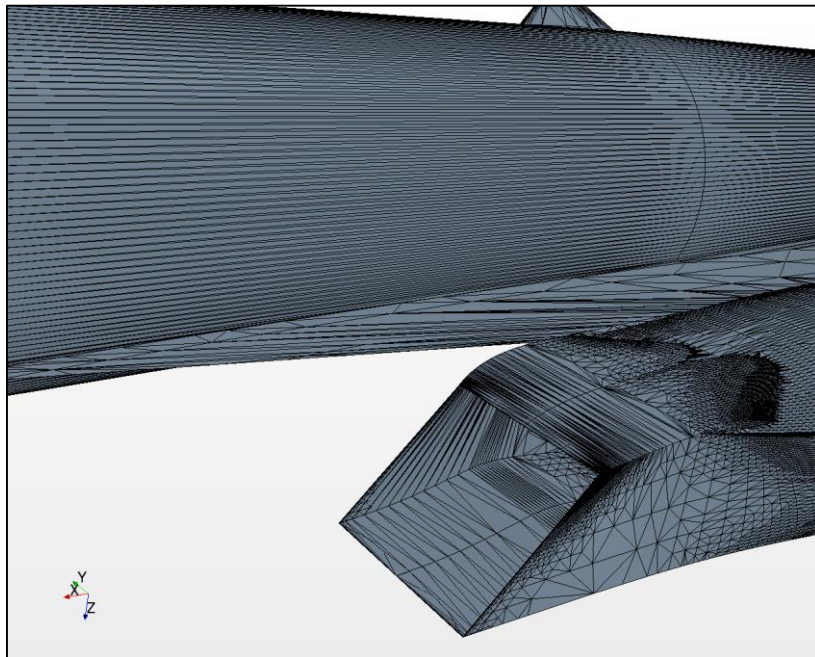


Figure 2.14. STAR-CCM+ Part Tessellation.

The Surface Mesh representation is generated next. This important step imposes the boundary conditions for the simulation, as well as the boundary face sizing of the final volume mesh. The mesh generator is a curvature-based, triangular grid; the sizing is controlled primarily by a target edge length, with curvature refinement down to a prescribed minimum edge length [23]. Within STAR-CCM+, this is performed as a Parts-Based Mesh Operation. We use surface controls applied to individual boundaries to limit the element edge length on the body, with separate controls for internal and external surfaces. The surface size at the far-field boundary is allowed to be large, on the order of the vehicle length [23]. The sizes applied are shown in Table 2.15.

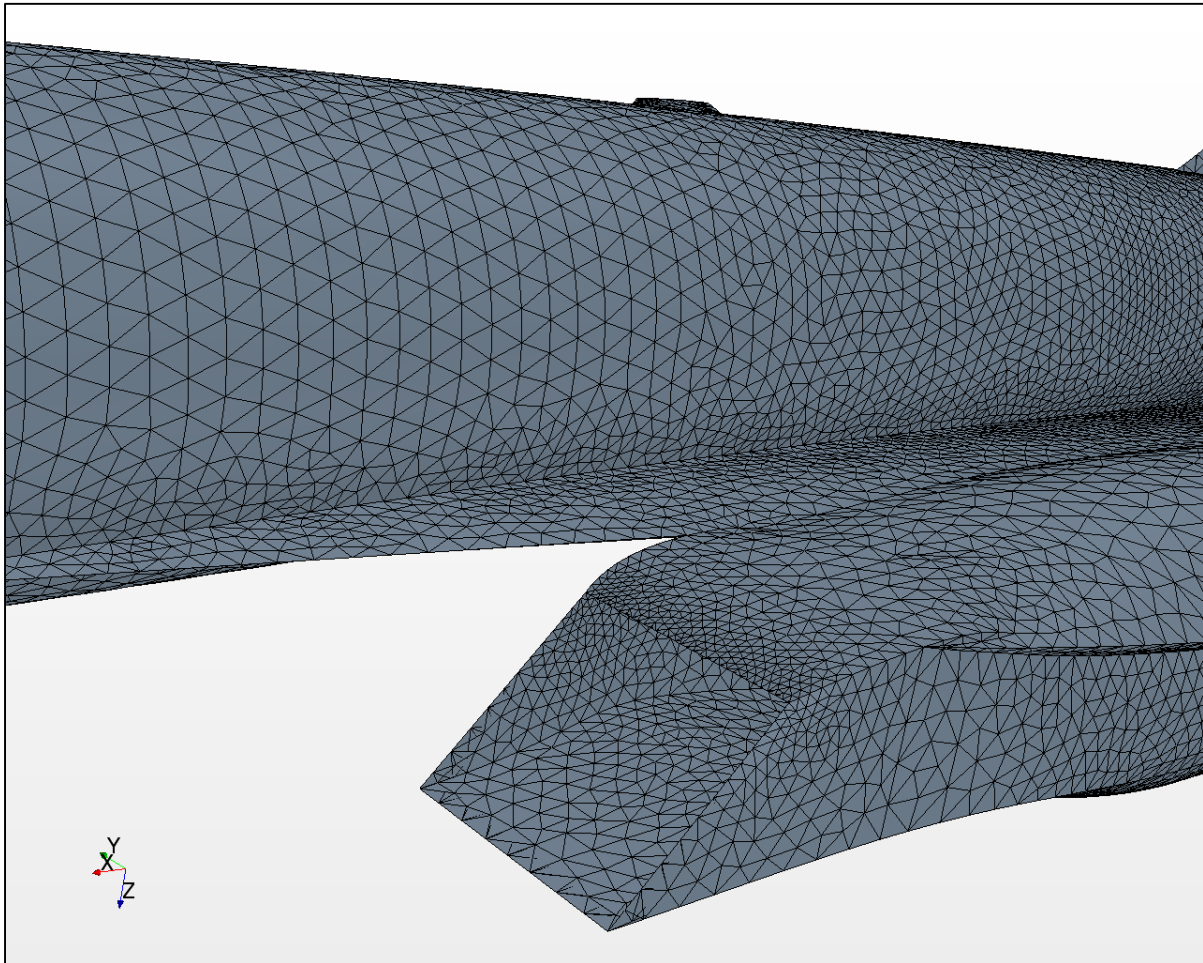


Figure 2.15. STAR-CCM+ Surface Mesh.

Table 2.1 STAR-CCM+ Mesh Parameters.

Parameter	Value	Units
Base Size	30	m
Target Surface Size	100	% Base
Target Surface Size (Internal)	0.5	% Base
Minimum Surface Size	0.3	% Base
Surface Curvature	48	Pts/Circle
Surface Growth Rate	1.16	–
Number of Prism Layers	11	–
Prism Layer Stretching	1.5	–
Prism Layer Total Thickness (External)	0.16	m
Prism Layer Total Thickness (Internal)	0.08	m
Volume Mesh Quality Threshold	0.6	–
Volume Mesh Optimization Cycles	3	–

The final step in the mesh process generates the volume mesh. This single-step process fills the fluid volume with polyhedral cells using sizing based on the surface mesh; it is possible to incorporate additional sizing constraints, but for our purposes the surface size controls were found to be sufficient. This process also generates a layer of prism cells at wall boundaries in order to resolve the boundary layer gradient. STAR-CCM+ does provide the option to use rectangular or tetrahedral volume cells, but these options do not support the fluid region interfaces used for our engine flowpath. Figure 2.16 shows the engine inlet volume mesh, with the actual volume cells shown in green.

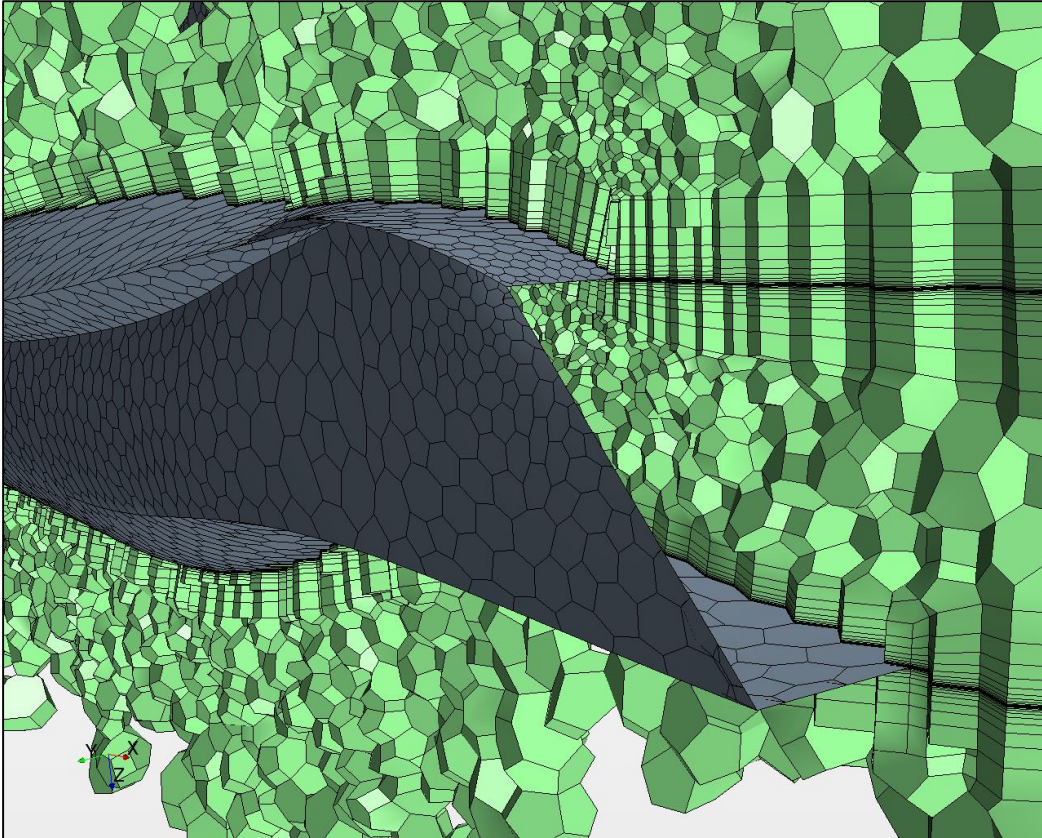


Figure 2.16. STAR-CCM+ Volume Mesh, Intake Detail.

This image also shows the treatment of prism layers at the intake edges; the prisms are continued along the region interface into open space. This feature is used on the nozzle flow region as well, and allows better resolution of several important phenomena related to the engine: the shock-pattern in the inlet and its interaction with the boundary layer, and the fluid shear where the nozzle flow rejoins the free stream.

With the volume mesh in place, the solution process can begin. STAR-CCM+ computes its flow solution in a global Cartesian reference frame and coordinate system, called Laboratory. We first define an additional coordinate system with respect to these global coordinates, which we use as a flow-centered coordinate system. In this way the freestream flow direction as well as directions for lift and drag forces can be computed using the flow coordinate system; when the coordinate system itself is rotated the directions are preserved. This also simplifies the process of applying multiple transformations with the Euler angle convention; the coordinate system is

simply rotated about its own Z-Y-X axes--in that order--to achieve the desired yaw, pitch and roll angles [17][34].

In STAR-CCM+, the governing equations to be solved on the volume mesh are selected by choosing Models within a physics continuum applied to the fluid region. We used a three-dimensional, steady, viscous coupled flow regime (the specific model choices are listed in Table 2.2), derived from the following integral form of the Navier-Stokes equations [23][35]:

$$\frac{\partial}{\partial t} \int_V \mathbf{W} \chi dV + \oint [\mathbf{F} - \mathbf{G}] \cdot d\mathbf{a} = \int_V \mathbf{H} dV \quad (2)$$

where \mathbf{W} includes density and flow velocity, \mathbf{F} includes transport and grid velocity, \mathbf{G} includes viscous stress and heat flux, and finally \mathbf{H} accounts for body forces. χ represents fluid domain porosity, and is unused (unity) for this analysis. The fluid was modeled as an ideal gas, with the properties of standard-atmosphere air. Turbulence and separation are modeled using the RANS SST K-Omega turbulence with source terms taken from dimensionless user boundary conditions Turbulence Intensity and Viscosity Ratio. These were set to 0.01 and 10 respectively for all cases. For near-wall locations, the STAR-CCM+ All-Y+ Wall Treatment was used. This model assumes a linear viscous layer and a logarithmic region, blending the two using an intermediate function. For momentum, this is accomplished using Reichardt's law, here in non-dimensionalized form [23]:

$$u^+ = \frac{1}{\kappa} \ln(1 + \kappa y^+) + C \left[1 - \exp\left(-\frac{y^+}{D}\right) - \frac{y^+}{D} \exp(-by^+) \right] \quad (3)$$

$$D = y_m^+ \quad C = \frac{1}{\kappa} \ln\left(\frac{E^+}{\kappa}\right) \quad b = \frac{1}{2} \left(\frac{D\kappa}{C} + \frac{1}{D} \right) \quad (4)$$

Temperature blending is derived from Kader's law, modified to accommodate a wall roughness function (smooth walls were assumed throughout these cases) [23]:

$$t^+ = \exp(-\Gamma) (t_{\text{lam}}^+ - q_{\text{lam}}^+) + \exp\left(-\frac{1}{\Gamma}\right) (t_{\text{trb}}^+ - q_{\text{trb}}^+) \quad (5)$$

$$\Gamma = \frac{0.01 c (\sigma y^+)^4}{1 + \frac{5}{c} \sigma^3 y^+} \quad (6)$$

Table 2.2. STAR-CCM+ Physics Models used for SCCT Cruise Evaluation.

Regime	Model Used
Space	Three-Dimensional
Time	Steady
Material	Gas
Equation of State	Ideal Gas
Flow Regime	Coupled Flow
Energy Regime	Coupled Energy
Viscous Regime	Turbulent
Turbulence	Reynolds-Averaged Navier-Stokes
Reynolds-Averaged Turbulence	K-Omega Turbulence
K-Omega Turbulence Model	SST (Menter) K-Omega
K-Omega Wall Treatment	All- y^+ Wall Treatment

Coupled flows in STAR-CCM+ allow the use of the expert solver tools, Grid-Sequencing Initialization and the Expert Driver. The former is used to provide a well-posed initial condition by solving the full geometry and boundary conditions for a simplified inviscid flow, then passing that solution to the main solver. The latter is a tool which actively modulates the CFL number of the implicit solver while running to give faster convergence while maintaining stability. As an additional stabilizing feature, the solver is run for the first 250 iterations using a first-order coupled flow discretization, before switching to second-order accuracy. This operation was

necessary to give consistent stability to the engine inlet flow. The parameters used for these solvers are given in Table 2.3; other solver parameters were left at default values.

Table 2.3. STAR-CCM+ Solver Parameters used for SCCT Cruise Evaluation.

Courant Number	80
Grid Sequencing CFL Number	150
Grid Sequencing Convergence Tolerance	0.02
Expert Driver CFL Ramp End Iteration	100
Expert Driver Minimum Explicit Relaxation	0.5

STAR-CCM+ uses tools called Reports to extract values of interest from the simulation. These simply extract the raw flow values on the appropriate cells or faces, and compute the desired output. Many options are available to be reported due to the richness of data available from a CFD solution. The reports used in this analysis are shown below in Table 2.4.

Table 2.4. STAR-CCM+ Reported Values used in SCCT Cruise Evaluation.

Name	Report Type	Entity	Direction
Cell Count	Element Count	All Regions	-
Planform Area, S	Projected Area	All Wall-Type Boundaries	[0, 0, 1]
Wing Semispan, b	Maximum	All Wall-Type Boundaries	[0, 1, 0]
Fuselage Length, l	Minimum	All Wall-Type Boundaries	[1, 0, 0]
Lift	Force	All Wall-Type Boundaries	[0, 0, -1]
Drag	Force	All Wall-Type Boundaries	[-1, 0, 0]
Side Force	Force	All Wall-Type Boundaries	[0, -1, 0]
X-moment	Moment	All Wall-Type Boundaries	[1, 0, 0]
Y-moment	Moment	All Wall-Type Boundaries	[0, 1, 0]
Z-moment	Moment	All Wall-Type Boundaries	[0, 0, 1]
L/D	Expression	Lift, Drag	-
Fan Mass Flow Rate	Mass Flow	Fan Face	-
Fan Total Pressure	Mass-Flow Averaged Total Pressure	Fan Face	-
Fan Total Temperature	Mass-Flow Averaged Total Temperature	Fan Face	-

These reports are saved to the final output of the simulation, but are also used to control the solution during the implicit iteration process. The flow conditions at the fan face are used by the macro code for the on-line engine deck to compute the nozzle flow conditions and the specific fuel consumption. Also, the report values are used to stop the solver when sufficient convergence is attained; this is done when the L/D ratio and engine mass flow rate stabilize asymptotically to five significant digits.

The STAR-CCM+ macro code evaluates five different flow solutions in order to provide sufficient data to the subsequent performance analysis. The first solutions to be run are three pitch angles at 1, 3, and 5 degrees angle of attack. Forces and moments are recorded at each angle. Within the macro code, a least-squares regression is then used to fit a line and parabola to the Drag and Lift force curves, respectively; a numerical solver then computes the alpha value which maximizes L/D. This angle of attack is then evaluated.

To obtain a sideslip solution, the bilateral symmetry assumption is abandoned. We create a copy of the mesh and mirror those cells about the symmetry plane, then combine the regions. In this process, the centerline symmetry boundary is removed. A single sideslip case is evaluated using a 1-degree sideslip angle, and pitched at the previously computed maximum L/D angle of attack.

Once these solutions have been completed, the recorded values are written to an output file in comma-separated value (.csv) format. Forces and moments are recorded for all simulated conditions, as well as cell count, mass flow rate, body length, span, as well as the system time required to solve all cases. Also, the computed thrust-specific fuel consumption and maximum L/D angle of attack are written. All other simulation data is discarded, and the STAR-CCM+ process repeats over each CAD model provided to the macro.

Post-Analysis using MATLAB

MATLAB is used to collect and analyze the data results from STAR-CCM+. We used a written script to accomplish this task, as opposed to the interactive MATLAB prompt; this allows the post-analysis to be repeated precisely for different data sets without significant effort. First, the data results from STAR-CCM+ are read from the comma-separated text files, then collate each recorded value into a vector of that quantity across all tested CAD models. The code is written to search a given directory and read all .csv files present, concatenating the data into a single set. This was done so that a STAR-CCM+ analysis could be split into multiple operations for the range of models, allowing that operation to be parallelized.

After this concatenation, the MATLAB workspace contains a series of result vectors, each having dimension equal to the population of models tested. At this point, additional derived result values are computed from the direct results. These derived values are used to evaluate the optimization objective function and assess the performance of each model, as well as provide additional information about the population. The values computed are based on an assumed cruise-climb flight profile, terminating at the required maximum altitude and operating at maximum lift-to-drag ratio throughout the cruise.

First, the weights are calculated. The lift produced at maximum L/D is corrected to the maximum altitude by multiplying by the density ratio between the evaluation flow and the known maximum altitude. This maximum-altitude lift is equated to the weight of the vehicle at cruise termination, assumed to consist of the aircraft empty weight, 5% fuel, and the full payload. Using an empirical relation from Nicolai, the gross takeoff weight is computed as a function of the empty weight, and the cruise start weight is computed as a function of the gross takeoff weight. This computation requires the use of the secant method. Fuel weight is obtained simply by subtracting the payload and empty weight from the gross vehicle weight.

Once the weights are computed, performance results can be estimated. The range is calculated using the Breguet range equation and the cruise phase weight ratio [4]. The stability derivatives with respect to pitch and sideslip are obtained from the central differences of the STAR-CCM+ results. The aerodynamic center and neutral point are then computed using simple

ratios, and used to obtain the static margin [4]. The use of aerodynamic center assumes that the vehicle center of gravity is located at the aerodynamic center, which is known as the trimmed flight condition. Finally, we define a Dutch roll coefficient to quantify the roll moment direction during sideslip flight; this coefficient is simply the ratio of the roll moment to the yaw restoring moment. All the relations described above are listed below in Equations X-Y.

$$W_4 = \frac{\rho_{min}}{\rho_{eval}} (L)_{L/Dmax} \quad (7)$$

$$W_e = 0.911W_0^{0.947} \quad (8)$$

$$W_3 = (0.9)(0.95)W_0 \quad (9)$$

$$W_f = W_0 - W_e - W_p \quad (10)$$

$$R = \frac{V L}{C D} \ln \left(\frac{W_3}{W_4} \right) \quad (11)$$

$$X_{ac} = \left(\frac{M_y}{L} \right)_{L/Dmax} \quad (12)$$

$$X_{np} = \frac{\frac{\partial M_y}{\partial \alpha}}{\frac{\partial L}{\partial \alpha}} \quad (13)$$

$$SM = (X_{np} - X_{ac}) \frac{b}{S} \quad (14)$$

$$C_{DR} \stackrel{\text{def}}{=} \left(\frac{M_x}{M_z} \right)_{sideslip} \quad (15)$$

We then find the subset of tested design candidates which pass feasibility tests based on these computed values, and sort those models according to their performance. These feasibility tests are based on the defined mission parameters, as well as stability criteria for steady flight. The criteria used are listed and described in Table 2.5.

Table 2.5. SCCT Candidate Feasibility Criteria.

Criterion	Description
$W_0 < 121000 \text{ lb}_f$	Gross Takeoff weight must satisfy mission requirement.
$R > 5000 \text{ nmi}$	Cruise Range must satisfy mission requirement.
$\left (X_{ac} - \bar{X}) \left(\frac{b}{S} \right) \right < 0.01$	Trimmed CG location must be within 1% MAC of the aircraft volume centroid.
$D < T$	Cruise drag must be less than the available engine thrust.
$0.01 < SM < 0.05$	Static Margin must be between 1% and 5%. [Phillips][Raymer]
$C_{DR} > 0$	Dutch Roll coefficient must be positive. [Phillips]

Once the set of feasible design candidates are found, they are sorted according to performance using an objective function. The choice of the objective function is a crucial aspect of a design optimization [24] and as such we devoted careful thought to the decision. In the end, two options were seriously considered, and both make an effective objective function for optimization of the SCCT design. Both of these options consider the most challenging performance criteria of the mission profile; the range target of 5,000 nautical miles, and the weight target of 121,000 pounds. The first option is to attempt to find the lowest-weight candidate that satisfies the range requirement; the second option is to attempt to find the longest-range candidate that satisfies the weight constraint.

In the case of the first option, the objective function simply becomes the calculated gross takeoff weight. The chief advantage of this approach is that the vehicle weight is an excellent predictor of the vehicle cost [4]; thus, minimizing the weight can be considered to minimize the unit cost as well. However, a major motivation for the mission weight target is the noise regulations governing the aircraft; improving the gross takeoff weight below this limit does not change the noise category further.

For the second option, the objective function is the negative of the range. Minimizing this function gives a longer cruise range; Figure 1.1 shows that a longer range allows a higher proportion of trips to be made by the aircraft, and consequently broadens the available market for the vehicle. Also, this benefit is not coarsely discretized; even marginal improvements to the

cruise range provide additional feasible trips. For this reason, we chose to use the calculated range as the objective function for this project.

Finally, a number of data measures are plotted to understand the data. Scatter plots of the range with respect to weight, wing area and fuel volume are created; histogram plots are created for several output values. Parameter estimations for the mean and variance of the computed range and takeoff weight are made, using a normal distribution assumption.

Meta-Analysis using DSM

Before approaching the details of our analysis, we first performed an overall study of our analysis process. This meta-analysis helped to set the general approach to the process, indicating the scope of analysis which should be undertaken as well as the order in which to perform each simulation step. We used a Design Structure Matrix (DSM) to accomplish this meta-analysis. Each task in a process such as ours requires certain data as input, and produces different data as output; DSM uses analytical methods to identify these information relationships between a series of tasks. The tasks within the process are referred to as Subsystems, and are placed on the main diagonal of a square matrix [26]. The input dependency is shown in all off-diagonal positions: Any required inputs from preceding analyses are shown in the upper right of the matrix while inputs required from subsequent analysis are shown in the lower left. The DSM for our conceptual design process is shown below in Figure 2.17. The figure shows many individual subsystems, grouped into the partitions of the overall analysis. They have been organized to both minimize the internal iteration needed within each partition, as well as eliminate any overall iteration. Practically, this means that the solution is closed for a given set of input candidates.

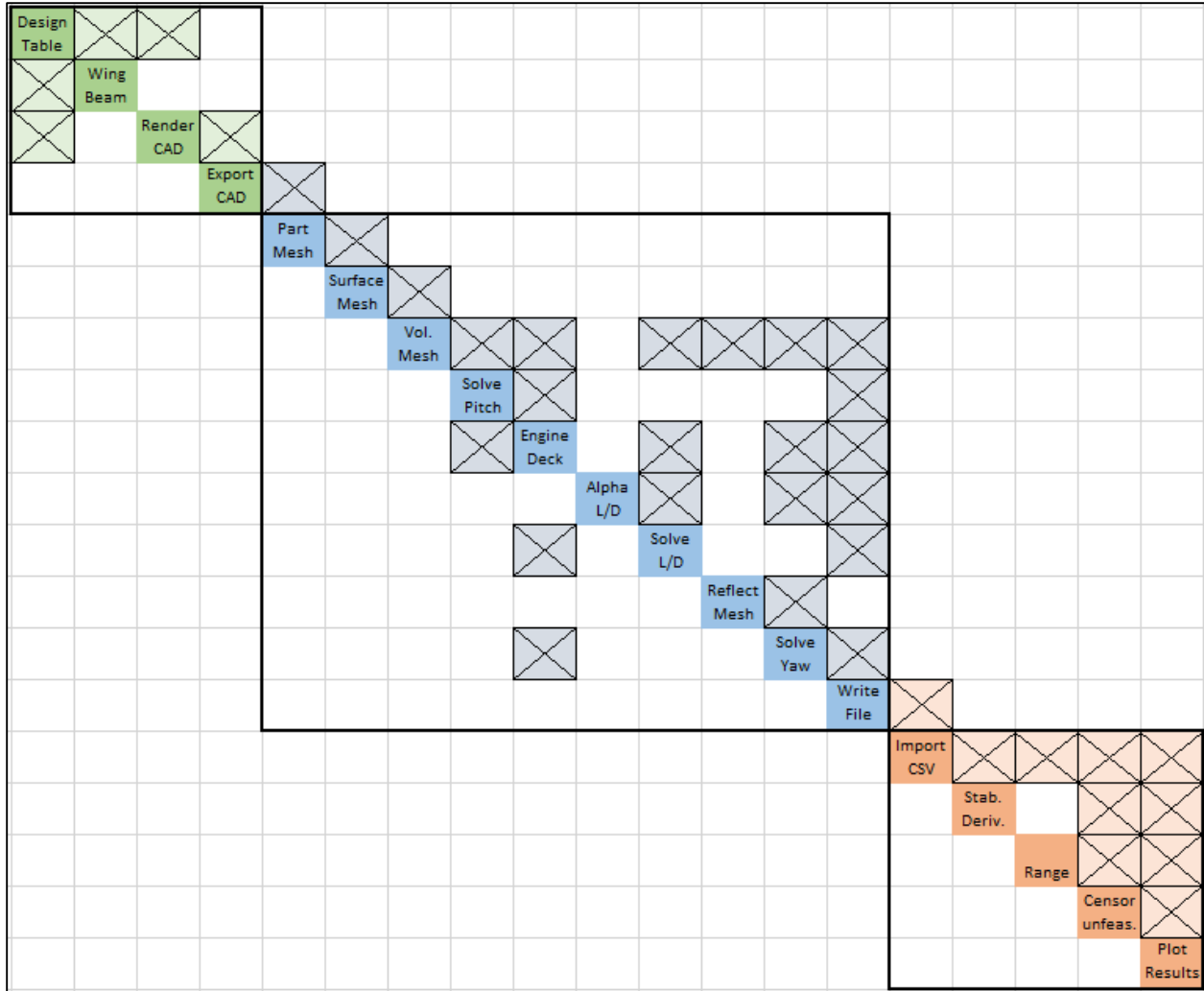


Figure 2.17. Supercruise Analysis DSM.

The subsystems in our DSM were chosen to provide a thorough analysis of the cruise phase of flight for our design. Since the design is a long-range transport, cruising flight has the largest impact on mission performance. Also, the major subsystem groups (i.e. fluid flow, structural analysis) used to analyze cruising flight are also applied to other phases of flight. Therefore, the methods developed in our approach can easily be expanded to include other these phases. The first group is the CAD geometry, which forms the physical geometry of the aircraft design candidates. Since the design table is determined by the construction methods used in the CAD deck, but also drives the construction of the vehicle, these are a coupled system.

Coupling in the STAR-CCM+ partition is present between the engine deck and solver subsystems. Frequent iteration was implemented between the engine deck and the solver while running a solution, in order to update the engine boundary condition; however the computation of the engine deck is extremely rapid and does not impede the solver subsystem.

CHAPTER III

RESULTS

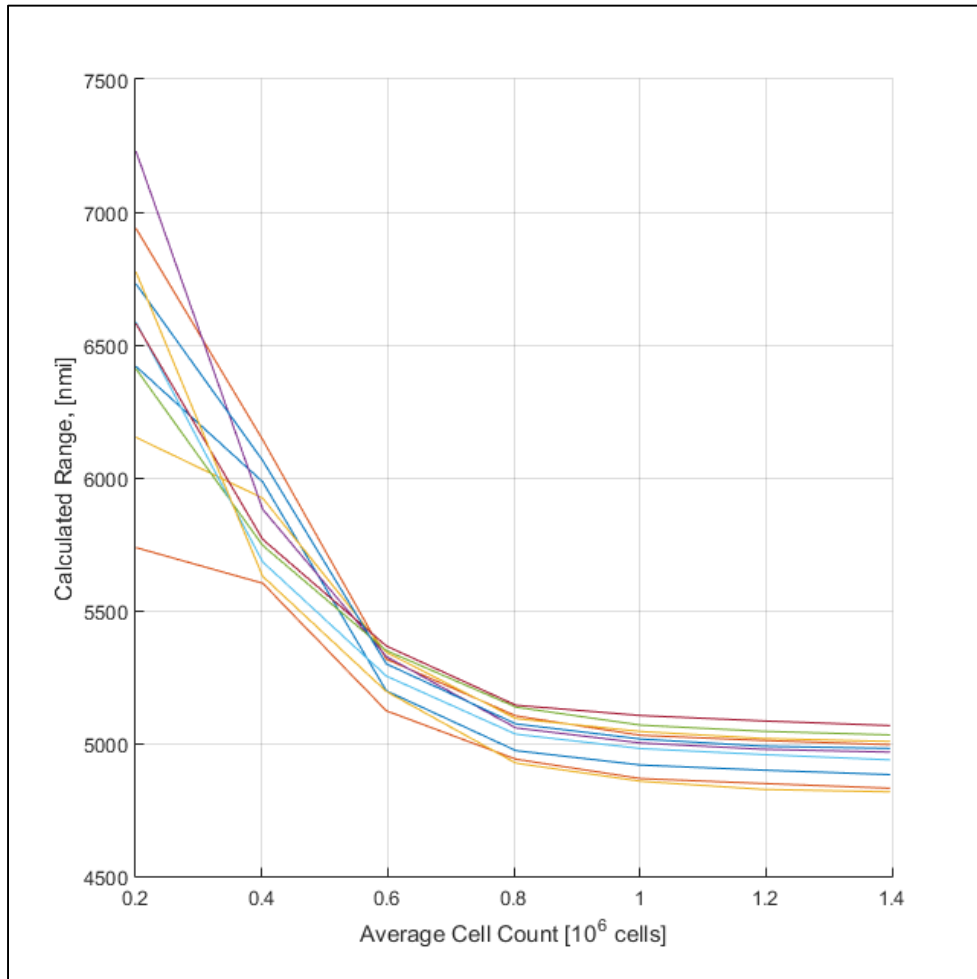


Figure 3.1. Sensitivity of Range to STAR-CCM+ mesh resolution.

Figure 3.1 shows the computed range of ten randomly selected SCCT candidate models; each line represents the results of a given model across a variety of different element counts. The element counts were adjusted by changing the minimum size and prism cell count from the values listed in Table 2.1; the cell count listed is for the sideslip case using the reflected, non-symmetric mesh. This study was used to judge the correct number of elements required to obtain consistent and accurate results from the workflow.

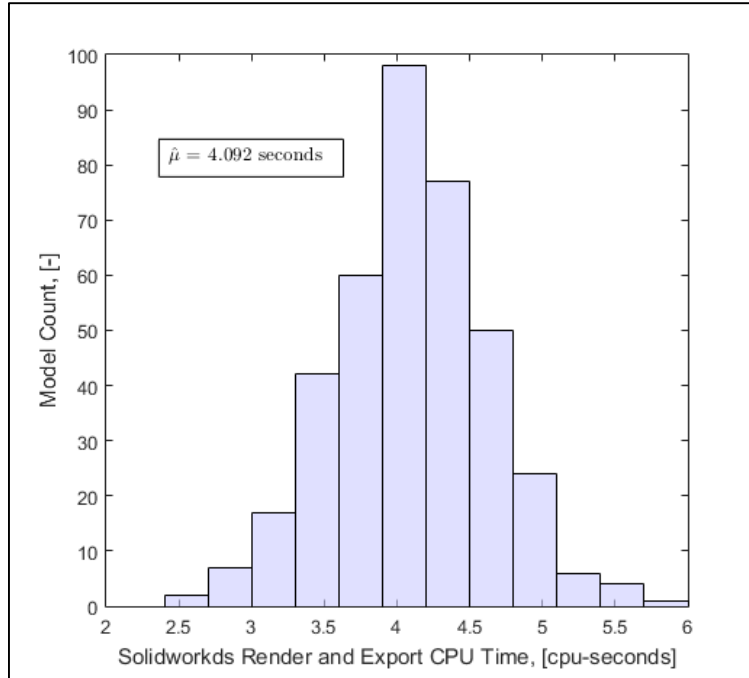


Figure 3.2. Evaluation Time comparison for Solidworks analysis.

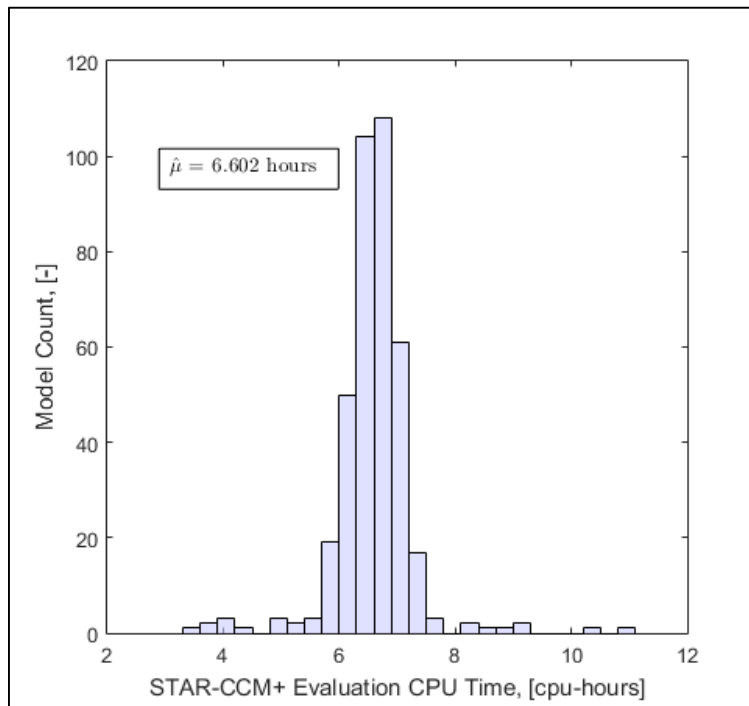


Figure 3.3. Evaluation Time comparison for STAR-CCM+ analysis.

Figures 3.2 and 3.3 show the evaluation time for the Solidworks and STAR-CCM+ portions of analysis. The data was collected from a single group of design candidates generated from a CAD deck; a total of 388 models were used. The evaluation times are given in CPU time, which is the overall time of the analysis multiplied by the number of CPU cores used; alternatively, this can be thought of as the expected time for the analysis to run on a single CPU core. All computations were performed on Intel Xeon CPU dies having 8 cores total and clock speeds between 3.1 and 3.5 GHz. Note that the time units in Figure 3.2 are seconds, while those of Figure 3.3 are hours. The sample mean CPU time is shown in the inset for each figure.

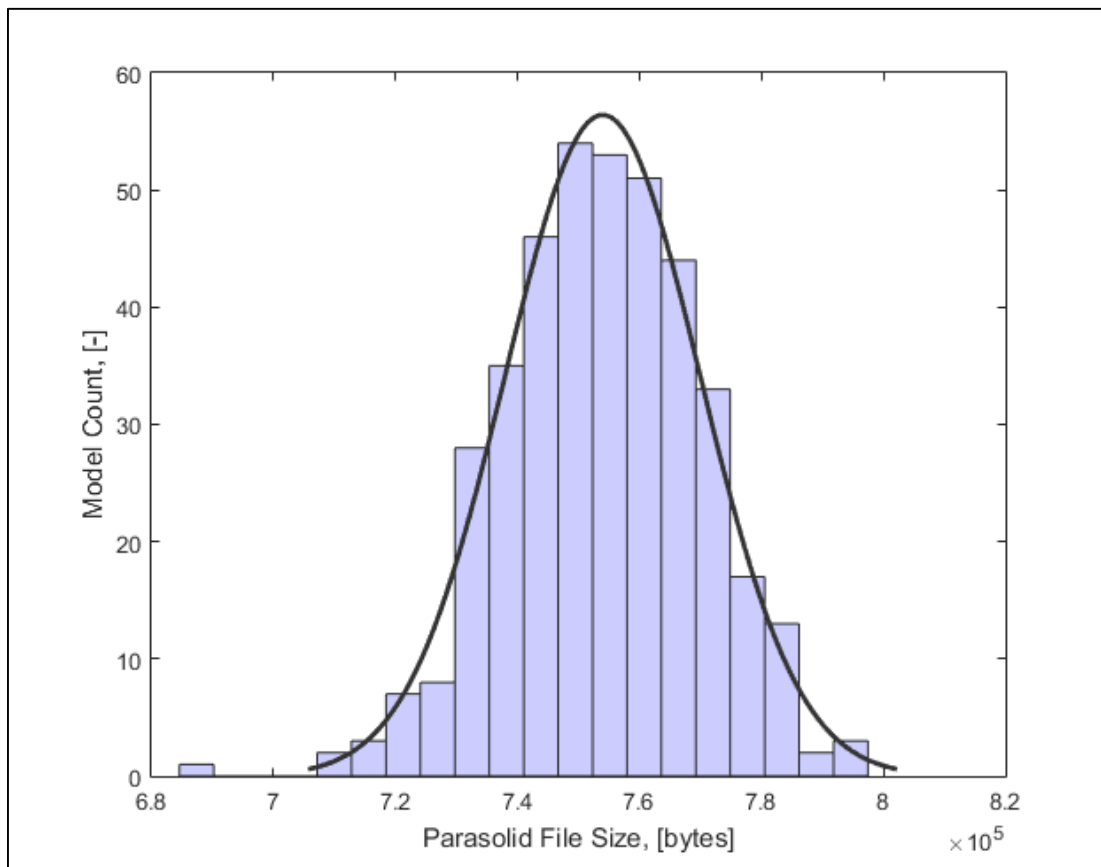


Figure 3.4. Histogram of Candidate Population CAD export file sizes.

Figure 3.4 illustrates the variation in file size for a typical randomized population of CAD models. The size shown is the on-disk size of the file. The line represents a least-squares normal

distribution computed for the sample data. This population in particular contains the failed model shown in Figure 2.11; this model is indeed the outlier visible at the left of the figure, and its file size is 4.34 standard deviations below the mean.

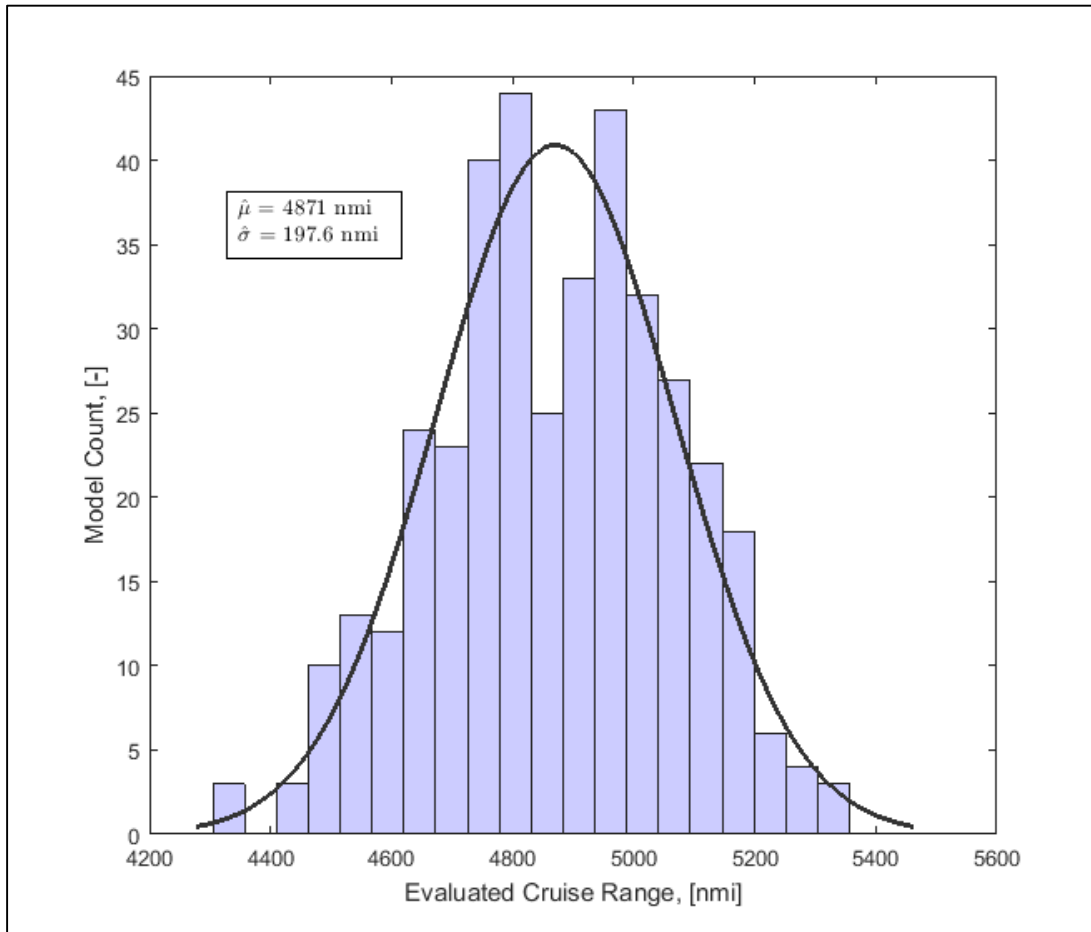


Figure 3.5. Histogram of Candidate Population Evaluated Cruise Range.

Figure 3.5 shows the evaluated range of a randomized sample population from a CAD deck, and the estimated normal distribution of the output. Parameter estimations for the distribution mean and standard deviation are shown in the inset. The data shown represents the complete set of models tested, prior to being filtered by the constraints given in Table 2.5.

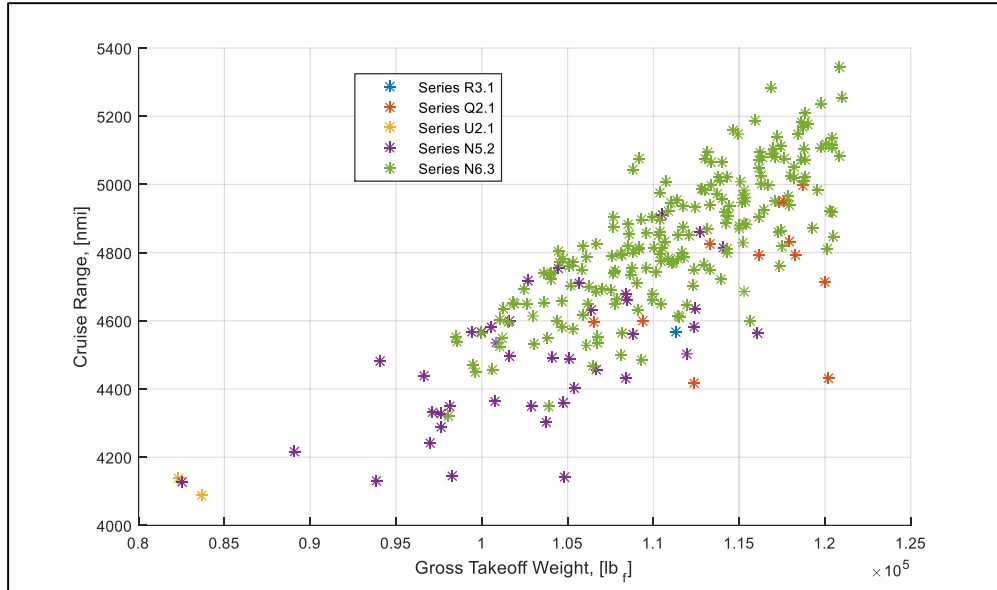


Figure 3.6. Comparison of feasible candidates from multiple configuration options.

Figure 3.6 shows a comparison of models across five different CAD decks, comparing the evaluated gross takeoff weight and the cruise range. The five series listed in the legend correspond to the different decks used; these were constructed in order to implement aircraft configurations whose differences were too large to implement in a single CAD deck. In the order listed, the series implemented a delta wing with underslung engines, a delta wing with engines above the wing, a three-engine design, an aft-wing model with embedded engines, and an aft-wing model with wing-mounted engines. Infeasible models (those which fail any of the criteria listed in Table 2.5) are omitted from this plot.

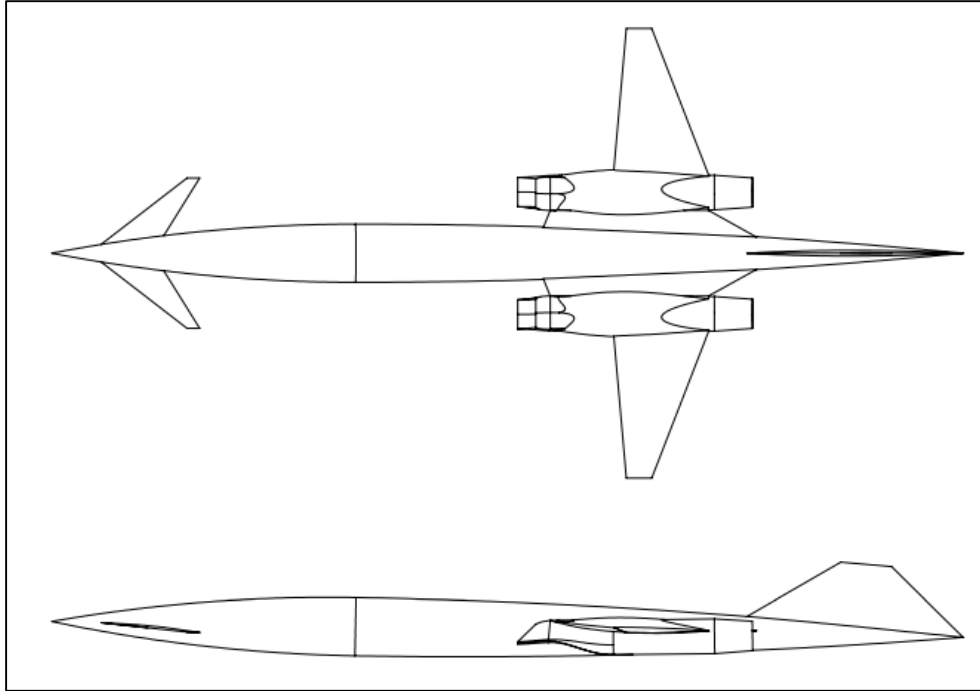


Figure 3.7. Design Study Best Performing Candidate.

Figure 3.7 shows a diagram of the best performing model tested, according to the range-based objective function discussed in Chapter 2 and subject to the constraints of Table 2.5. This model obtains a range of 5350 nautical miles, at a gross takeoff weight of 120,650 pounds; the cruise lift-to-drag ratio is 8.13, and its thrust-specific fuel consumption is 0.879 $\text{lb}_m/\text{lb}_f/\text{hr}$, with the engine at 79% of maximum available thrust during cruise. Its wingspan is 58.5 feet, with an overall length of 118 feet. The total fuel capacity is estimated to be 9800 gallons. This model is a member of the population shown in Figure 3.5 and represents a result 2.4 standard deviations above the sample mean.

CHAPTER IV

DISCUSSION

Meta-Analysis

We are primarily interested in the effectiveness of this analysis method with regard to the problem of conceptual design. We therefore wish to know whether it produces a more robust result than established methods, and whether the time cost of the analysis falls within the one-hour target we established. The first step in judging the efficacy of the design methodology we have developed is to examine the realism of the results. The grid sensitivity study shown in Figure 3.1 allows us to estimate the mesh resolution required for our analysis. In particular, the STAR-CCM+ analysis must accomplish two overall objectives. It should accurately predict the absolute range of the design candidates, but more importantly it must provide the correct relative ranking of the candidates when evaluated according to the objective function. This can obviously be accomplished by maximizing the number of cell elements used, but the analysis comes at a very high time cost; STAR-CCM+ is by far the most time-consuming portion of the analysis. Therefore, the minimum number of elements required to provide acceptable results should be used. With fewer than 400,000 elements, the simulation produces no useful result. The ranking of models is arbitrary and inconsistent, and the cruise range is significantly overestimated. The inaccuracy is due to an underestimation of the shear drag, due to insufficient boundary layer resolution. High cell counts continue to obtain more accurate results for this drag, but only 750,000 cells are required to predict the correct ranking of model performance. The mesh sizing used for subsequent simulations was then set to provide at least this number of elements.

The time required to perform the analysis is of critical importance. In order for the analysis to be feasible at the conceptual design stage, the time required should be less than one hour per design candidate. This time should also be achievable using a reasonable level of computation resource. Figures 3.2 and 3.3 show the time distribution among the analysis partitions, and show clearly the time difference between STAR-CCM+ and Solidworks. The MATLAB computation time is negligible by comparison, occupying less than one second for an

entire population of candidates. In total, the analysis has an expected total CPU time per model of 23,771 seconds, or 6.6 hours. In order for this analysis to be completed within one hour, at least seven computation cores must be used. Modern workstation machines are normally equipped with eight or sixteen CPU cores, so this analysis is feasible for conceptual design on common hardware. The use of a larger computation cluster would allow significantly more analyses to be performed.

Since random variables are used in our CAD deck, we can consider the output to be a random variable as well; we can then analyze its distribution. From statistics[36], the Central Limit Theorem holds that the mean of a large number of independent random variables will be approximately normally distributed, regardless of the individual distributions of the starting variables. The function used in this analysis is not a simple mean and so cannot be expected to always produce a normally-distributed result, but we may intuitively test the assumption with a fit and provide some insight to the data results. Some examples are shown in Figures 3.4 and 3.5.

In the first of these, we see how the file size of the data is distributed. This assumption can be used to assess the quality of the CAD models produced without examining each model individually. Using this assumed distribution to check for outliers proves to be a reliable method to detect failed CAD models, and can be performed automatically. This can be used to test the quality of a CAD deck, by assessing its ability to reproduce models throughout a given range of design variables. Figure 3.5 shows the distribution of objective function results produced by the population. This can also be used to assess the potential for further examination of a design; we can see that high-performing results are potentially available in the upper tail of this distribution. Accessing these results can be done by simply generating larger random populations, or (more feasibly) by using MDO tools like Particle Swarm or Genetic Algorithm [27]. Conversely, a result distribution which falls far below the mission target would have a vanishingly small probability of producing a feasible result; this would indicate that changes need to be made in the construction of the CAD deck to allow different or broader configuration changes.

Design Candidate Variations

In Figure 3.6 we see the individual results pertinent to the design case. This result shows that certain configuration options are clearly superior to others, with all of the best-performing options having an aft-mounted wing and wing-mounted engines. For this case, each configuration option used a different CAD deck; this was done only so that the results could be separated and assess the relative merits of the configurations. The methods we use to construct the CAD deck could produce all the listed configurations in a single population. The figure shows an obvious correlation between the evaluated weight and range. This is a result of the coupling between gross takeoff weight and wing area; higher-weight models have correspondingly higher wing area, directly proportional to the increased lift generation. If this increase in lift comes with a minimal increase in drag, then this increase also propagates to the lift-to-drag ratio and the cruise range.

Figure 3.7 gives the best performing candidate found in this case study, which meets all the mission criteria established. The detail of this result shows the benefit of applying these analysis methods; the amount of information available about this candidate is much greater than that produced by simpler analysis methods [4][11][19][31], allowing for a much more sophisticated preliminary design phase. Since we also have significant information about the design space occupied by this result, we have confidence in its robustness. Notable characteristics of this model are a wing which passes through the fuselage centerline behind the pressure vessel, allowing for a more slender fuselage. Additionally, the lack of leading edge extensions or chines eliminates associated vortex drag and lends more efficient cruise flight.

Included Preliminary Design factors

One of the goals of this design methodology was to include preliminary design factors in the conceptual stage, without increasing the cost; we will now enumerate and discuss those factors. The use of a complete CAD geometry allows the examination of several phenomena which are beyond the scope of simpler geometric representations. We include complex wing designs incorporating variable forward and backward sweep, leading edge extensions and

fuselage chines, fuselage/wing blending, complex twist and thickness profiles, and tip treatments. The flexibility of the approach allows unconventional topologies to be modeled, such as diamond or ring-shaped wings [20].

The use of full-body CFD captures interactions between the various components of the aircraft; wing-fuselage, wing-tail, fuselage-inlet, etc. are couplings which are naturally detected using our methods; these interactions are very difficult to characterize when analyzing the components separately [4]. Additionally, the use of coupled, viscous flow resolves the shear drag of the vehicle without including additional relations; this is very important for supersonic flow where the skin friction accounts for a significant portion of the overall drag [37][38].

Several propulsion related factors are calculated automatically in our approach. Most importantly, the STAR-CCM+ evaluation includes an engine deck, based on a non-ideal low-bypass turbofan, as found in Chapter 7 of [39]. The conditions at the fan-face (pressure out let boundary) are passed to a Java method which computes the nozzle boundary conditions. This is updated every 10 iterations in STAR-CCM+. The engine deck also saves the cruise TSFC and maximum thrust, which are included in the output calculation. Further, the cell-weighted standard deviation of mass flow is computed for the fan face boundary at each angle of attack, allowing us to characterize the flow uniformity feeding the engine. In addition to this performance data, geometric information is captured by the CAD implementation. The engine nacelle orientation has a profound effect on aircraft performance; analysis involving the engine nacelle angle simply adds an extra variable to our analysis. Since the engine location/orientation is known from the CAD, we can compute the moment due to an engine out or unstart condition. Once again, we assume that the CG is located at the computed X_{ac} location; the engine-out moment is then the gross thrust of the engine multiplied by the perpendicular distance to the thrust line of action.

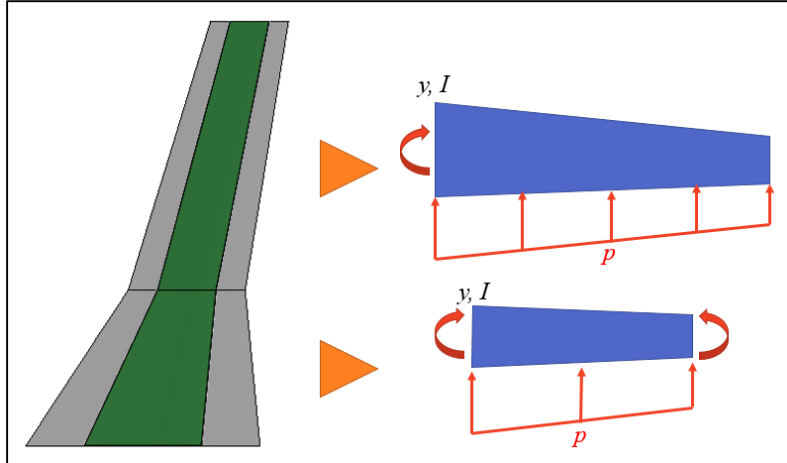


Figure 4.1. Wing Beam Bending Schematic.

To address structural considerations, we use a cantilevered beam approximation to limit the minimum thickness of the wing [18], as diagrammed in Figure 4.1. We assume the limiting case where the center 40% of the wing section is a solid beam, shown in green; the loading is assumed to be a constant pressure corresponding to the average wing loading observed at a +3-g flight condition. Starting from the tip, the next-section minimum moment of area required to support the wing is computed based on the loading, as well as the moment exerted by any outboard sections, and based on that section's chord length we obtain the minimum allowable airfoil section thickness. This computation is performed for the canard and main wing, and occurs in the CAD design table.

CHAPTER V

SUMMARY AND CONCLUSIONS

Summary

We created a methodology to perform conceptual design analysis on aircraft. Our aim was to use off-the-shelf, high-fidelity software tools to explore the project design space, including important preliminary design factors and thereby producing a more robust result which is less subject to compromise at later design stages. We claim that this analysis can be performed in one hour with commonly available computation resources, and therefore is applicable to conceptual design.

We used the case study of a supersonic transport jet to develop these methods. For this application, we used Solidworks to create a parameterized three-dimensional CAD solid to define the exterior geometry of the aircraft, and create populations of design candidates. We used STAR-CCM+ to perform an automated fluid flow analysis of these candidates, using three-dimensional, viscous, turbulent finite volume analysis and incorporating internal engine performance characteristics. We then used MATLAB to collect the data produced by these analyses, compute additional results of interest, and quantify the design space represented by a population of candidates. We heavily automated the steps of this process, to allow large studies or optimization frameworks to be implemented.

Our results show that the method produces a data set which is much richer than conventional conceptual design techniques. The method captures many interactions between aircraft systems which are normally not quantified until later phases of design: aerodynamic interactions between external lifting surfaces and between the external body and internal engine performance, and how structural constraints affect wing performance. We also produce detailed information about the aircraft static stability. Further, the method is able to produce these results with commonly available computer hardware within the one-hour timeframe we allow for a conceptual design analysis.

Conclusions

Aircraft design—and indeed any design process—must be supported by some analysis of potential candidates. We have claimed that the level of detail involved in aircraft conceptual design should be tailored to give an analysis which can be performed in a single hour using reasonable computation resources. In order for this analysis to provide the most robust result, the analysis should be as detailed as possible within these constraints. We explored this problem by constructing a set of automated analyses using off-the-shelf commercial software, applied to the design of a supersonic transport aircraft. Our analysis used parametric 3D CAD geometry feeding aerodynamic analysis using Solidworks, and STAR-CCM+.

Our analysis successfully produced results under the constraints listed above, and produced viable design candidates which satisfy both the mission requirements and the feasibility limits of flight. The method produces a rich set of result data which captures many preliminary design factors which are normally left to later phases of the design process. In the case study at hand, we obtained detailed information about the installed engine performance and the coupling between all external lifting surfaces; the outer geometry was also constrained to allow for structural feasibility. The method also produces a statistical analysis of the design space, allowing for an understanding of the distribution of results and prediction of further design steps. The analysis scales easily with additional computation resources, which would allow still more detail to be included within the limiting timeframe of conceptual design. In this particular case, it would be beneficial to include aerodynamic analysis of takeoff, landing, climb and descent phases of flight [39], as well as internal packaging of components and structure [18].

REFERENCES

- [1] Spitz, W., et al., "Development Cycle Time Simulation for Civil Aircraft," NASA CR-2001-210658, January 2001.
- [2] Wikimedia Foundation. <http://www.wikimedia.org>.
- [3] Parnell, G. Q. (ed.), *Decision Making in Systems Engineering and Management, Second Edition*, Wiley, Hoboken, NJ, 2011, Chaps. 2, 4.
- [4] Nicolai, L. and Carichner, G., *Fundamentals of Aircraft and Airship Design, Vol. I*, AIAA Education Series, AIAA, Reston, VA, 2010.
- [5] Bottcher, J., "Aircraft Noise Certification," *ICAO Noise Certification Workshop*, ICAO, Montreal, 2004.
- [6] Dickson, N., "ICAO Noise Standards," *ICAO Symposium on Aviation and Climate Change, Destination Green*, ICAO, Montreal, 2013.
- [7] Higgins, M., "New York to Dubai for \$19,000," *New York Times*, New York, February 10, 2012.
- [8] Vincenty, T., "Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations," *Survey Review*, Vol. 22, No. 176, Apr. 1975, pp. 88-93.
- [9] Civil Aviation Subcommittee, Aviation Safety Committee, Aerospace Medical Association, "Cabin Cruising Altitudes for Regular Transport Aircraft," *Aviation, Space, and Environmental Medicine*, Vol. 79, No. 4, 2008, pp. 433-439.
- [10] Bowman, Martin W. *Lockheed F-104 Starfighter*. Crowood Press Ltd., Ramsbury, Marlborough, Wiltshire, UK 2000.
- [11] Cumpsty, N., *Jet Propulsion: A Simple Guide to the Aerodynamic and Thermodynamic Design and Performance of Jet Engines, 2nd Edition*, Cambridge University Press, Cambridge, UK, 2003.
- [12] Beniada, F., *Concorde*, Zenith Press, Minneapolis, MN, 2006.
- [16] SAI Corporation. <http://www.sai-qsstx.com>.
- [17] Etkin, B., and Reid, L.D., *Dynamics of Flight: Stability and Control, Third Edition*, Wiley, Hoboken, NJ, 2010, Chaps. 2, 4.
- [18] Megson, T. H. G., *Aircraft Structures for Engineering Students, Fifth Edition*, Elsevier, Oxford, UK, 2013, Chaps. 1, 2.

- [13] Sweetman, B., *F-22 Raptor*, Motorbooks Press, Minneapolis, MN, 1998.
- [14] Aerion Corporation, <http://www.aerionsupersonic.com>.
- [15] Spike Aerospace. <http://www.spikeaerospace.com>.
- [19] Tornado Project, "TORNADO 1.0 User's Guide," Release 2.3, January 31, 2001. Retrieved from: <http://tornado.redhammer.se/images/manual.pdf>
- [20] Planchard, M., "Engineering Design with Solidworks 2011," SDC Publications, Mission, KS, 2011.
- [21] Iualiano, E., et al., "Design of a Natural Laminar Flow Wing-Body," *Journal of Aircraft*, Vol. 48, No. 4, pp. 1147-1162.
- [22] Durbin, P. A., and Petterson Reif, B. A., *Statistical Theory and Modeling for Turbulent Flows*, Wiley, Hoboken, NJ, 2003.
- [23] CD-adapco, STAR-CCM+ User Guide, Version 9.06.009, CD-adapco, Melville, NY, 2014.
- [24] Alonso, J. J. and Colonno, M. R., "Multidisciplinary Design Optimization with Applications to Sonic-Boom Minimization," *Ann. Rev. Fluid Mech*, Vol. 44, 2012, pp. 505-526.
- [25] Pektas, S. T., Pultar, M., "Modelling Detailed Information Flows in Building Design with the Parameter-Based Design Structure Matrix," *Design Studies*, Vol. 27, No. 1, January 2006, pp. 99-122.
- [26] Tang, D., et al., "Product Design Knowledge Management based on Design Structure Matrix," *Advanced Engineering Informatics*, Vol. 24, No. 2, April 2010, pp. 159-166.
- [27] Li, W., et al., "Hierarchy and Adaptive Size Particle Swarm Optimization Algorithm for Solving Geometric Constraint Problems," *Journal of Software*, Vol. 7, No. 11, November 2012, pp. 2567-2574.
- [28] Venter, G. and Sobieszczanski-Sobieski, J., "Particle Swarm Optimization," *AIAA Journal*, Vol. 41, No. 8, August 2003, pp. 1583-1589.
- [29] Lee, A. D., "Parametric modeling for simulation based hypersonic vehicle design," M.S. Thesis, Aerospace Engineering Dept., Iowa State University, 2015.
- [30] Dassault Systemes, SOLIDWORKS 2014 API Help, Dassault Systemes, Vélizy-Villacoublay, France, 2014.
- [31] Berguin, S. H., and Mavris, D. N., "Dimensionality Reduction Using Principal Component Analysis in Aerodynamic Design," *AIAA Paper No. 2014-0112*, 2014.

- [32] Slater, J. W., “Design and Analysis Tool for External-Compression Supersonic Inlets,” AIAA Paper 2012-0016, Jan. 2012.
- [33] Slater, J. W., “Methodology for the Design of Streamline-Traced External-Compression Supersonic Inlets,” Propulsion and Energy Forum, AIAA/ASME/SAE/ASEE Joint Propulsion Conference, 2014.
- [34] Phillips, W. F., *Mechanics of Flight, Second Edition*, Wiley, Hoboken, NJ, 2010, Chaps. 1, 3, 4, 5.
- [35] Pletcher, R. H., Tannehill, J. C., and Anderson, D. A., *Computational Fluid Mechanics and Heat Transfer*, CRC Press, Boca Raton, FL, 2013.
- [36] Miller, M., *Mathematical Statistics with Applications*, Pearson, Upper Saddle River, NJ, 2014, Chap. 8.
- [37] Ishikawa, H., et al., “Development of Supersonic Natural Laminar Flow Wing Design System Using CFD-Based Inverse Method,” 28th *International Congress of the Aeronautical Sciences*, Brisbane, Australia, September 2012.
- [38] Jones, R. T., “Theory of Wing-Body Drag at Supersonic Speeds,” NACA Report 1284, 1956.
- [39] Mattingly, J. D., *Elements of Propulsion: Gas Turbines and Rockets*, AIAA Education Series, AIAA, Reston, VA, 2006, Chaps. 6, 7.
- [40] Bryson, A. E., et al., “The Energy-State Approximation in Performance Optimization of Supersonic Aircraft,” AIAA Paper No. 68-877, August 1968.
- [41] ANSYS, Mechanical APDL Help 15.0, ANSYS, Inc., Canonsburg, PA, 2013.
- [42] Baldwin, A. N., et al., “Planning Building Design by Simulating Information Flow,” *Automation in Construction*, Vol. 8, No. 2, 1998, pp. 149-163.
- [43] CD-adapco, STAR-CCM+ Java API, Version 9.06.009, CD-adapco, Melville, NY, 2014.
- [44] Denham, W. F., and Bryson, A. E., “A Steepest-Ascent Method for Solving Optimum Programming Problems,” *Journal of Applied Mechanics*, Vol. 29, No. 2, 1962, pp. 247-257.
- [45] Koziel, S., and Leifsson, L., “Multi-Objective Airfoil Design Using Variable-Fidelity CFD Simulations and Response Surface Surrogates,” AIAA Paper 2014-0289, 2014.

APPENDIX A

JAVA SOURCE CODE FOR STAR-CCM+

The following source code was used to evaluate collections of Parasolid models for the SCCT design case. It performs the complete simulation and is constructed for batch operation over large sets of models. It also contains numerous error-capture scenarios to make the code more robust against poor geometry and/or evaluation errors.

```

/*=====
    SCCT Evaluation

This macro imports aircraft bodies from .x_b files and
evaluates four flow conditions: three pitch angles using
a centerline symmetry assumption as well as a single
full-body yaw condition. From this it saves a .csv file
containing the forces and moments necessary to evaluate
total range and static stability derivatives.

The create() method creates the simulation and sets mesh
and physics constants.

The import_poly() method contains a loop to import and
evaluate many bodies in a separately imported free-stream
region. Shock-refined meshing at a lower mach is used. The
import and mesh portion of the code includes error-capturing
in order to reject poor CAD models, continuing to the next
loop cycle.

Lines of interest:
    124..137      mesh settings
    140..145      physics settings
    151..163      solver settings
    171..172      loop numbers (model count)
    212, 222, 224 import paths
    266..267      mesh freestream velocity
    290..292      mesh refine settings
    364..365      run freestream velocity
    206, 436      output path

=====*/
package macro;

import java.util.*;
import java.lang.*;
import java.io.*;

import star.base.report.*;
import star.base.neo.*;
import star.common.*;
import star.coupledflow.*;
import star.energy.*;
import star.flow.*;
import star.kwturb.*;
import star.material.*;
import star.meshing.*;
import star.metrics.*;
import star.prismmesher.*;
import star.resurfacer.*;

```

```

import star.trimmer.*;
import star.turbulence.*;
import star.vis.*;

public class cruise_climb2_3 extends StarMacro {

    public void execute() {
        create();
        import_poly();
    }

    private void create() {
        Simulation simulation_0 = getActiveSimulation();

        // create continua and turn on models
        MeshContinuum meshContinuum_0 =
simulation_0.getContinuumManager().createContinuum(MeshContinuum.class);
        PhysicsContinuum physicsContinuum_0 =
simulation_0.getContinuumManager().createContinuum(PhysicsContinuum.class);
        meshContinuum_0.enable(ResurfacrMeshingModel.class);
        meshContinuum_0.enable(TrimmerMeshingModel.class);
        meshContinuum_0.enable(PrismMesherModel.class);
        physicsContinuum_0.enable(ThreeDimensionalModel.class);
        physicsContinuum_0.enable(SteadyModel.class);
        physicsContinuum_0.enable(SingleComponentGasModel.class);
        physicsContinuum_0.enable(CoupledFlowModel.class);
        physicsContinuum_0.enable(IdealGasModel.class);
        physicsContinuum_0.enable(CoupledEnergyModel.class);
        physicsContinuum_0.enable(TurbulentModel.class);
        physicsContinuum_0.enable(RansTurbulenceModel.class);
        physicsContinuum_0.enable(KOmegaTurbulence.class);
        physicsContinuum_0.enable(SstKwTurbModel.class);
        physicsContinuum_0.enable(KwAllYplusWallTreatment.class);

        // get mesher objects
        MaximumCellSize maximumCellSize_0 =
meshContinuum_0.getReferenceValues().get(MaximumCellSize.class);
        GenericRelativeSize genericRelativeSize_0 = ((GenericRelativeSize)
maximumCellSize_0.getRelativeSize());
        NumPrismLayers numPrismLayers_0 = meshContinuum_0.getReferenceValues().get(NumPrismLayers.class);
        PrismThickness prismThickness_0 = meshContinuum_0.getReferenceValues().get(PrismThickness.class);
        prismThickness_0.getRelativeOrAbsoluteOption().setSelected(RelativeOrAbsoluteOption.ABSOLUTE);
        GenericAbsoluteSize genericAbsoluteSize_0 = ((GenericAbsoluteSize)
prismThickness_0.getAbsoluteSize());
        SurfaceGrowthRate surfaceGrowthRate_0 =
meshContinuum_0.getReferenceValues().get(SurfaceGrowthRate.class);
        SurfaceSize surfaceSize_0 = meshContinuum_0.getReferenceValues().get(SurfaceSize.class);
        RelativeMinimumSize relativeMinimumSize_0 = surfaceSize_0.getRelativeMinimumSize();
        RelativeTargetSize relativeTargetSize_0 = surfaceSize_0.getRelativeTargetSize();

        // make region
        Region region_0 = simulation_0.getRegionManager().createEmptyRegion();
        region_0.setPresentationName("FluidDomain");
        Boundary boundary_veh = region_0.getBoundaryManager().getBoundary("Default");
        boundary_veh.setPresentationName("Faces");
        Boundary boundary_0 = region_0.getBoundaryManager().createEmptyBoundary();
        Boundary boundary_1 = region_0.getBoundaryManager().createEmptyBoundary();
        Boundary boundary_2 = region_0.getBoundaryManager().createEmptyBoundary();
        boundary_0.setPresentationName("symm");
        boundary_1.setPresentationName("inlet");
        boundary_2.setPresentationName("outlet");
        boundary_0.setBoundaryType(SymmetryBoundary.class);
        boundary_1.setBoundaryType(FreeStreamBoundary.class);
        boundary_2.setBoundaryType(PressureBoundary.class);
        FeatureCurve featureCurve_0 = region_0.getFeatureCurveManager().createEmptyFeatureCurve();
        featureCurve_0.setPresentationName("CraftEdges");
        FeatureCurve featureCurve_1 = region_0.getFeatureCurveManager().getFeatureCurve("Default");

```



```

featureCurve_1.setPresentationName("Edges");

// make flow coordinates
Units units_0 = simulation_0.getUnitsManager().getPreferredUnits(new IntVector(new int[] {0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}));
Units rads = simulation_0.getUnitsManager().getPreferredUnits(new IntVector(new int[] {0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}));
LabCoordinateSystem labCoordinateSystem_0 =
simulation_0.getCoordinateSystemManager().getLabCoordinateSystem();
CartesianCoordinateSystem flow_coord =
labCoordinateSystem_0.getLocalCoordinateSystemManager().createLocalCoordinateSystem(CartesianCoordinateSystem.class, "Flow");
Coordinate coordinate_0 = flow_coord.getOrigin();
coordinate_0.setCoordinate(units_0, units_0, units_0, new DoubleVector(new double[] {0.0, 0.0, 0.0}));
coordinate_0.setValue(new DoubleVector(new double[] {0.0, 0.0, 0.0}));

// mesh constants
meshContinuum_0.getReferenceValues().get(BaseSize.class).setValue(20.0);
genericRelativeSize_0.setPercentage(100.0);
numPrismLayers_0.setNumLayers(8);
genericAbsoluteSize_0.getValue().setValue(0.15);
surfaceGrowthRate_0.setGrowthRate(1.08);
relativeMinimumSize_0.setPercentage(0.4);
relativeTargetSize_0.setPercentage(100.0);
SurfaceSizeOption surfaceSizeOption_0 =
featureCurve_0.get(MeshConditionManager.class).get(SurfaceSizeOption.class);
surfaceSizeOption_0.setSurfaceSizeOption(true);
SurfaceSize surfaceSize_1 = featureCurve_0.get(MeshValueManager.class).get(SurfaceSize.class);
RelativeMinimumSize relativeMinimumSize_1 = surfaceSize_1.getRelativeMinimumSize();
relativeMinimumSize_1.setPercentage(0.5);
RelativeTargetSize relativeTargetSize_1 = surfaceSize_1.getRelativeTargetSize();
relativeTargetSize_1.setPercentage(0.5);

// physics constants
StaticTemperatureProfile staticTemperatureProfile_0 =
physicsContinuum_0.getInitialConditions().get(StaticTemperatureProfile.class);
physicsContinuum_0.getReferenceValues().get(ReferencePressure.class).setValue(18753.9);

staticTemperatureProfile_0.getMethod(ConstantScalarProfileMethod.class).getQuantity().setValue(216.65);
FlowDirectionProfile flowDirectionProfile_0 = boundary_1.getValues().get(FlowDirectionProfile.class);
flowDirectionProfile_0.getMethod(ConstantVectorProfileMethod.class).getQuantity().setComponents(-1.0,
0.0, 0.0);
flowDirectionProfile_0.setCoordinateSystem(flow_coord);

// solver params
CoupledImplicitSolver coupledImplicitSolver_0 = ((CoupledImplicitSolver)
simulation_0.getSolverManager().getSolver(CoupledImplicitSolver.class));

coupledImplicitSolver_0.getExpertInitManager().getExpertInitOption().setSelected(ExpertInitOption.GRID_SEQ
_METHOD);
GridSequencingInit gridSequencingInit_0 = ((GridSequencingInit)
coupledImplicitSolver_0.getExpertInitManager().getInit());
coupledImplicitSolver_0.setCFL(100.0);
gridSequencingInit_0.setMaxGSLevels(5);
gridSequencingInit_0.setConvGSTol(0.02);
gridSequencingInit_0.setGSCfl(150.0);

coupledImplicitSolver_0.getSolutionDriverManager().getExpertDriverOption().setSelected(ExpertDriverOption.
EXPERT_DRIVER);
ExpertDriverCoupledSolver expertDriverCoupledSolver_0 = ((ExpertDriverCoupledSolver)
coupledImplicitSolver_0.getSolutionDriverManager().getDriver());
expertDriverCoupledSolver_0.setEndIteration(100);
KwTurbSolver kwTurbSolver_0 = ((KwTurbSolver)
simulation_0.getSolverManager().getSolver(KwTurbSolver.class));

kwTurbSolver_0.getRampCalculatorManager().getRampCalculatorOption().setSelected(RampCalculatorOption.LINEA
R_RAMP);

```



```

// loop over models
for(int model = start; model<=end; model++){

    epoch = System.currentTimeMillis()/1000;
    //=====
    // Import Parts and make Fluid Region
    //=====
    File psolid = new File(resolvePath("../cad//"+model+".x_b"));
    if(!psolid.exists()){continue;}
    partImportManager_0.importCadPart(resolvePath("../cad//"+model+".x_b"), "SharpEdges",
30.0, 3, 1.0E-5, true, false);
    CadPart cadPart_0 = null;
    try{cadPart_0 = ((CadPart)
simulation_0.get(SimulationPartManager.class).getPart(Integer.toString(model)));}
    catch(java.lang.Exception modelerr0){
        simulation_0.println(modelerr0);
        CompositePart compositePart_0 = ((CompositePart)
simulation_0.get(SimulationPartManager.class).getPart("Assembly 1"));
        simulation_0.get(SimulationPartManager.class).removeParts(new NeoObjectVector(new
Object[] {compositePart_0}));
        continue;
    }
    PartCurve partCurve_0 = ((PartCurve) cadPart_0.getPartCurveManager().getObject("Edges"));
    partCurve_0.setPresentationName("CraftEdges");
    CadPart cadPart_2 = null;
    try{cadPart_2 = (CadPart) meshActionManager_0.subtractParts(new NeoObjectVector(new
Object[] {cadPart_0, cadPart_1}), cadPart_1, "CAD");}
    catch(java.lang.Exception modelerr1){
        simulation_0.close(ServerConnection.CloseOption.ForceClose);
        simulation_0 = null;
        simulation_0 = new Simulation(resolvePath("../CCworking"+start+".sim"));
        reflect = simulation_0.getUnitsManager().getPreferredUnits(new IntVector(new
int[] {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}));
        rads = simulation_0.getUnitsManager().getPreferredUnits(new IntVector(new int[]
{0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}));
        labCoordinateSystem_0 =
simulation_0.getCoordinateSystemManager().getLabCoordinateSystem();
        flow_coord = ((CartesianCoordinateSystem)
labCoordinateSystem_0.getLocalCoordinateSystemManager().getObject("Flow 1"));
        physicsContinuum_0 = ((PhysicsContinuum)
simulation_0.getContinuumManager().getContinuum("Physics 1"));
        meshContinuum_0 = ((MeshContinuum)
simulation_0.getContinuumManager().getContinuum("Mesh 1"));
        meshManager_0 = simulation_0.getMeshManager();
        meshPipelineController_0 = simulation_0.get(MeshPipelineController.class);
        partImportManager_0 = simulation_0.get(PartImportManager.class);
        meshActionManager_0 = simulation_0.get(MeshActionManager.class);
        region_0 = simulation_0.getRegionManager().getRegion("FluidDomain");
        boundary_1 = region_0.getBoundaryManager().getBoundary("inlet");
        velocityProfile_0 =
physicsContinuum_0.getInitialConditions().get(VelocityProfile.class);
        machNumberProfile_0 = boundary_1.getValues().get(MachNumberProfile.class);
        partImportManager_0.importCadPart(resolvePath("../cad//halfDomain35.x_b"),
"SharpEdges", 30.0, 3, 1.0E-5, true, false);
        cadPart_1 = ((CadPart)
simulation_0.get(SimulationPartManager.class).getPart("Body 2"));
        simulation_0.println(modelerr1);
        continue;
    }
    cadPart_2.setPresentationName("FluidDomain");
    region_0.getPartGroup().setObjects(cadPart_2);

    //=====
    // Create shock-resolved mesh
    //=====

```

```

velocityProfile_0.getMethod(ConstantVectorProfileMethod.class).getQuantity().setComponents(-
442.31, 0.0, 0.0);

machNumberProfile_0.getMethod(ConstantScalarProfileMethod.class).getQuantity().setValue(1.5);

    try{meshPipelineController_0.generateVolumeMesh();}
    catch(java.lang.Exception mesherr0){
        simulation_0.println(mesherr0);
        simulation_0.get(SimulationPartManager.class).removeParts(new NeoObjectVector(new
Object[] {cadPart_0, cadPart_2}));
        continue;
    }
    flow_coord.setBasis0(new DoubleVector(new double[] {1.0, 0.0, 0.0}));
    flow_coord.setBasis1(new DoubleVector(new double[] {0.0, 1.0, 0.0}));
    flow_coord.getLocalCoordinateSystemManager().rotateLocalCoordinateSystems(new
NeoObjectVector(new Object[] {flow_coord}), new DoubleVector(new double[] {0.0, -1.0, 0.0}), new
NeoObjectVector(new Object[] {rads,rads,rads}), alpha[1]*Math.PI/180.0, flow_coord);
    try{simulation_0.getSimulationIterator().step(80);}
    catch(java.lang.Exception runerr0){
        simulation_0.println(runerr0);
        solution_0 = simulation_0.getSolution();
        solution_0.clearSolution();
        simulation_0.get(SimulationPartManager.class).removeParts(new NeoObjectVector(new
Object[] {cadPart_0, cadPart_2}));
        continue;
    }
    UserFieldFunction userFieldFunction_0 =
simulation_0.getFieldFunctionManager().createFieldFunction();
    userFieldFunction_0.setDefinition("mag(grad($MachNumber))");
    UserFieldFunction userFieldFunction_1 =
simulation_0.getFieldFunctionManager().createFieldFunction();
    userFieldFunction_1.setDefinition("0.888*pow($Volume,0.3333)");
    FvRepresentation fvRepresentation_0 = ((FvRepresentation)
simulation_0.getRepresentationManager().getObject("Volume Mesh"));
    CellSet cellSet_0 = fvRepresentation_0.getCellSetManager().createEmptyCellSet();
    cellSet_0.addThreshold(userFieldFunction_0, 1, 0.0015, 0.0015);
    // Mach gradient threshold
    PrimitiveFieldFunction primitiveFieldFunction_0 = ((PrimitiveFieldFunction)
simulation_0.getFieldFunctionManager().getFunction("Volume"));
    cellSet_0.subsetThreshold(primitiveFieldFunction_0, 1, 0.0035, 0.0035); //
Volume gradient threshold
    XyzInternalTable xyzInternalTable_0 =
simulation_0.getTableManager().createTable(XyzInternalTable.class);
    xyzInternalTable_0.getParts().setObjects(cellSet_0);
    xyzInternalTable_0.setFieldFunctions(new NeoObjectVector(new Object[]
{userFieldFunction_1}));
    xyzInternalTable_0.extract();
    TrimmerMeshingModel trimmerMeshingModel_0 =
meshContinuum_0.getModelManager().getModel(TrimmerMeshingModel.class);
    trimmerMeshingModel_0.setMeshSizeTable(xyzInternalTable_0);
    meshPipelineController_0.generateVolumeMesh();
    trimmerMeshingModel_0.setMeshSizeTable(null);
    simulation_0.getTableManager().remove(xyzInternalTable_0);
    fvRepresentation_0.getCellSetManager().removeObjects(cellSet_0);
    simulation_0.getFieldFunctionManager().removeObjects(userFieldFunction_0,
userFieldFunction_1);

    //=====
    // Monitors & Stopping Criteria
    //=====
    // force & moment reports
    ForceReport drag = simulation_0.getReportManager().createReport(ForceReport.class);
    drag.setPresentationName("Drag");
    ForceReport lift = simulation_0.getReportManager().createReport(ForceReport.class);
    lift.setPresentationName("Lift");
    ForceReport side = simulation_0.getReportManager().createReport(ForceReport.class);

```

```

MomentReport Mx = simulation_0.getReportManager().createReport(MomentReport.class);
MomentReport My = simulation_0.getReportManager().createReport(MomentReport.class);
MomentReport Mz = simulation_0.getReportManager().createReport(MomentReport.class);
drag.setCoordinateSystem(flow_coord);
lift.setCoordinateSystem(flow_coord);
side.setCoordinateSystem(flow_coord);
Boundary boundary_5 = region_0.getBoundaryManager().getBoundary("Faces");
Mx.getParts().setObjects(boundary_5);
My.getParts().setObjects(boundary_5);
Mz.getParts().setObjects(boundary_5);
drag.getParts().setObjects(boundary_5);
lift.getParts().setObjects(boundary_5);
side.getParts().setObjects(boundary_5);
drag.getDirection().setComponents(-1.0, 0.0, 0.0);
lift.getDirection().setComponents(0.0, 0.0, -1.0);
side.getDirection().setComponents(0.0, 1.0, 0.0);
Mx.getDirection().setComponents(1.0, 0.0, 0.0);
My.getDirection().setComponents(0.0, 1.0, 0.0);
Mz.getDirection().setComponents(0.0, 0.0, 1.0);
ExpressionReport LvsD =
simulation_0.getReportManager().createReport(ExpressionReport.class);
LvsD.setDefinition("$LiftReport/$DragReport");
ReportMonitor reportMonitor_0 = LvsD.createMonitor();
MonitorIterationStoppingCriterion monitorIterationStoppingCriterion_0 =
reportMonitor_0.createIterationStoppingCriterion();
StepStoppingCriterion stepStoppingCriterion_0 = ((StepStoppingCriterion)
simulation_0.getSolverStoppingCriterionManager().getSolverStoppingCriterion("Maximum Steps"));
((MonitorIterationStoppingCriterionOption)
monitorIterationStoppingCriterion_0.getCriterionOption()).setSelected(MonitorIterationStoppingCriterionOpt
ion.ASYMPTOTIC);
MonitorIterationStoppingCriterionAsymptoticType
monitorIterationStoppingCriterionAsymptoticType_0 = ((MonitorIterationStoppingCriterionAsymptoticType)
monitorIterationStoppingCriterion_0.getCriterionType());
monitorIterationStoppingCriterionAsymptoticType_0.getMaxWidth().setValue(0.0001);
stepStoppingCriterion_0.setMaximumNumberSteps(350);

// cell count report
ElementCountReport elementCountReport_0 =
simulation_0.getReportManager().createReport(ElementCountReport.class);
elementCountReport_0.getParts().setObjects(region_0);

// planform area report
FrontalAreaReport frontalAreaReport_0 =
simulation_0.getReportManager().createReport(FrontalAreaReport.class);
Coordinate coordinate_0 = frontalAreaReport_0.getViewUpCoordinate();
Units units_0 = ((Units) simulation_0.getUnitsManager().getObject("m"));
coordinate_0.setCoordinate(units_0, units_0, units_0, new DoubleVector(new double[] {1.0,
0.0, 0.0}));
frontalAreaReport_0.getParts().setObjects(boundary_5);

// span & length reports
MaxReport maxReport_0 = simulation_0.getReportManager().createReport(MaxReport.class);
MinReport minReport_0 = simulation_0.getReportManager().createReport(MinReport.class);
PrimitiveFieldFunction primitiveFieldFunction_1 = ((PrimitiveFieldFunction)
simulation_0.getFieldFunctionManager().getFunction("Position"));
VectorComponentFieldFunction vectorComponentFieldFunction_0 =
((VectorComponentFieldFunction) primitiveFieldFunction_1.getComponentFunction(0));
VectorComponentFieldFunction vectorComponentFieldFunction_1 =
((VectorComponentFieldFunction) primitiveFieldFunction_1.getComponentFunction(1));
maxReport_0.setScalar(vectorComponentFieldFunction_1);
minReport_0.setScalar(vectorComponentFieldFunction_0);
minReport_0.getParts().setObjects(boundary_5);
maxReport_0.getParts().setObjects(boundary_5);

//=====
// Run
//=====

```

```

velocityProfile_0.getMethod(ConstantVectorProfileMethod.class).getQuantity().setComponents(-
472.11, 0.0, 0.0);

machNumberProfile_0.getMethod(ConstantScalarProfileMethod.class).getQuantity().setValue(1.6);

// get 3 initial points
for(int i=0; i<=2; i++){

    solution_0 = simulation_0.getSolution();
    solution_0.clearSolution();
    flow_coord.setBasis0(new DoubleVector(new double[] {1.0, 0.0, 0.0}));
    flow_coord.setBasis1(new DoubleVector(new double[] {0.0, 1.0, 0.0}));

    flow_coord.getLocalCoordinateSystemManager().rotateLocalCoordinateSystems(new NeoObjectVector(new
Object[] {flow_coord}), new DoubleVector(new double[] {0.0, -1.0, 0.0}), new NeoObjectVector(new Object[]
{rads,rads,rads}), alpha[i]*Math.PI/180.0, flow_coord);
    try{simulation_0.getSimulationIterator().run(step(10));}
    catch(java.lang.Exception runerr1){
        simulation_0.println(runerr1);

simulation_0.getSolverStoppingCriterionManager().removeObjects(monitorIterationStoppingCriterion_0
);

        simulation_0.getMonitorManager().removeObjects(reportMonitor_0);
        simulation_0.getReportManager().removeObjects(drag, elementCountReport_0,
LvsD, frontalAreaReport_0, lift, maxReport_0, My);
        simulation_0.get(SimulationPartManager.class).removeParts(new
NeoObjectVector(new Object[] {cadPart_0, cadPart_2}));
        solution_0.clearSolution();
        continue;
    }
    D[i] = drag.monitoredValue();
    L[i] = lift.monitoredValue();
    m_x[i] = Mx.monitoredValue();
    m_y[i] = My.monitoredValue();
    m_z[i] = Mz.monitoredValue();
}

// fit function to L/D and optimize alpha for L/Dmax
Dc[2] = 0.125*D[0] - 0.25*D[1] + 0.125*D[2];
Dc[1] = -1.0*D[0] + 1.5*D[1] - 0.5*D[2];
Dc[0] = 1.875*D[0] - 1.25*D[1] + 0.375*D[2];
Lc[2] = 0.0;
Lc[1] = -0.25*L[0] + 0.25*L[2];
Lc[0] = 1.25*L[0] - .25*L[2];
double e = 1.0;
double diff = - 0.01;
double amax = 5.5;
double eval = 0.0;
double old = (Lc[1]*amax + Lc[0])/(Dc[2]*Math.pow(amax,2) + Dc[1]*amax + Dc[0]);
while(e>Math.pow(10.0,-14.0)){
    amax = amax + diff;
    eval = (Lc[1]*amax + Lc[0])/(Dc[2]*Math.pow(amax,2) + Dc[1]*amax + Dc[0]);
    if(eval<old) {
        amax = amax - 2*diff;
        diff = diff/10;
        eval = (Lc[1]*amax + Lc[0])/(Dc[2]*Math.pow(amax,2) + Dc[1]*amax + D[0]);
    }
    e = Math.abs((eval-old)/eval);
    old = eval;
}

// evaluate L/Dmax condition
solution_0.clearSolution();
flow_coord.setBasis0(new DoubleVector(new double[] {1.0, 0.0, 0.0}));
flow_coord.setBasis1(new DoubleVector(new double[] {0.0, 1.0, 0.0}));

```



```

        physicsContinuum_0 = ((PhysicsContinuum)
simulation_0.getContinuumManager().getContinuum("Physics 1"));
        meshContinuum_0 = ((MeshContinuum) simulation_0.getContinuumManager().getContinuum("Mesh
1"));

        meshManager_0 = simulation_0.getMeshManager();
        meshPipelineController_0 = simulation_0.get(MeshPipelineController.class);
        partImportManager_0 = simulation_0.get(PartImportManager.class);
        meshActionManager_0 = simulation_0.get(MeshActionManager.class);
        region_0 = simulation_0.getRegionManager().getRegion("FluidDomain");
        boundary_1 = region_0.getBoundaryManager().getBoundary("inlet");
        velocityProfile_0 = physicsContinuum_0.getInitialConditions().get(VelocityProfile.class);
        machNumberProfile_0 = boundary_1.getValues().get(MachNumberProfile.class);
        partImportManager_0.importCadPart(resolvePath("../cad/halfDomain35.x_b"), "SharpEdges",
30.0, 3, 1.0E-5, true, false);
        cadPart_1 = ((CadPart) simulation_0.get(SimulationPartManager.class).getPart("Body 2"));
    }
    simulation_0.close(ServerConnection.CloseOption.ForceClose);
}

public static double[] scctEngine(double mdot, double M0, double T0, double M2, double Tt2, double Pt2){

    double alpha = 0.25; //bypass ratio
    double c[] = {1.4, 1.27, 1005, 42800000.0, 287.0}; //gamma, gamma combusted, Cpc, hf, R
    double eta[] = {0.0, 0.92, 0.89, 0.998, 0.89}; //component efficiency
    double pi[] = {0.0, 4.25, 8.25, 0.0, 0.0}; //pressure ratio
    double tau[] = new double[6]; //temperature ratio
    double Tt[] = new double[20];
    double Pt[] = new double[20];
    double T[] = new double[20];
    double P[] = new double[20];

    // 0 diffuser r
    Tt[2] = Tt2;
    Tt[0] = 1.0 + (c[0]-1.0)/2.0*pow(M0,2);
    Pt[2] = Pt2;
    T[0] = T0;
    a0 = Math.sqrt(c[0]*c[4]*T[0]);
    tau[0] = Tt[2]/Tt[0];

    // 1 fan c'
    tau[1] = Math.pow(pi[1],(constants[0]-1)/(eta[1]*constants[0]));
    Pt[13] = Pt[2] * pi[1];
    Tt[13] = Tt[2] * tau[1];

    // 2 compressor c
    tau[2] = Math.pow(pi[2],(constants[0]-1)/(eta[2]*constants[0]));
    Pt[3] = Pt[2] * pi[2];
    Tt[3] = Tt[2] * tau[2];

    // 3 burner b
    Pt[4] = Pt[3] * eta[3];
    Tt[4] = 1500.0;
    tau[3] = Tt[4] / Tt[3];
    tau[5] = Tt[4] / To[0];
    double f = (tau[5] - tau[0] * tau[2])/(eta[3]*c[3]/c[2]/T[0]-tau[5]);

    // 4 turbine t
    //tau[4] = 1.0 - (tau[0]/tau[5])*((tau[2]-1.0) + alpha*(tau[1]-1.0)); // oates 5.111
    tau[4] = 1.0 - (tau[0]/tau[5])*((tau[2]-1.0) + alpha*(tau[1]-1.0))/eta[4]/(1.0+f); // Mattingly

    Tt[5] = Tt[4]*tau[4];
    pi[4] = Math.pow(tau[4],(constants[1]/(constants[1]-1.0)/eta[4]));
    Pt[5] = Po[3]*pi[4];

    double ST = 1.0 - 1.0/(tau[0]*tau[1]*pow(Pt[9]/Pt[13],(c[0]-1)/c[0]));
    ST = Math.sqrt(2.0/(c[0]-1)/(1.0+alpha) * ST * (tau[5]*tau[4]+alpha*tau[0]*tau[1]));
}

```

7.52p


```
ST = alpha * (ST - M0)
double S = f/(1.0+alpha)/ST;

double answers = {f, ST, S};
return answers;
}
}
```

APPENDIX B

VBA SOURCE CODE FOR SOLIDWORKS

The following source code was used to save collections of CAD geometry from Solidworks. It contains a loop over integer-named models and implements the ForceRebuild3 and SaveAs3 methods [30].

```
Dim swApp As Object

Dim Part As Object
Dim boolstatus As Boolean
Dim longstatus As Long, longwarnings As Long
```

```
Sub main()

Set swApp = Application.SldWorks
Set Part = swApp.ActiveDoc
Dim cname As String
Dim fname As String
Dim i As Integer

i = 1
cname = i
fname = "C:\Temp\models\" & i & ".x_b"
boolstatus = Part.ShowConfiguration2(cname)
i = 2

Do While boolstatus

boolstatus2 = Part.ForceRebuild3(True)
longstatus = Part.SaveAs3(fname, 0, 0)
cname = i
fname = "C:\Temp\models\" & i & ".x_b"
boolstatus = Part.ShowConfiguration2(cname)

i = i + 1

Loop

End Sub
```