Airplane Numerical Simulation for the Rapid Prototyping Process

By

Paul F. Roysdon
B.S. (University of California, Davis) 2004

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

MECHANICAL & AEROSPACE ENGINEERING

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

J.J. Chattot (Chair)

C.P. van Dam

R.A. Hess
Committee in Charge
2010

(This page intentionally blank.)

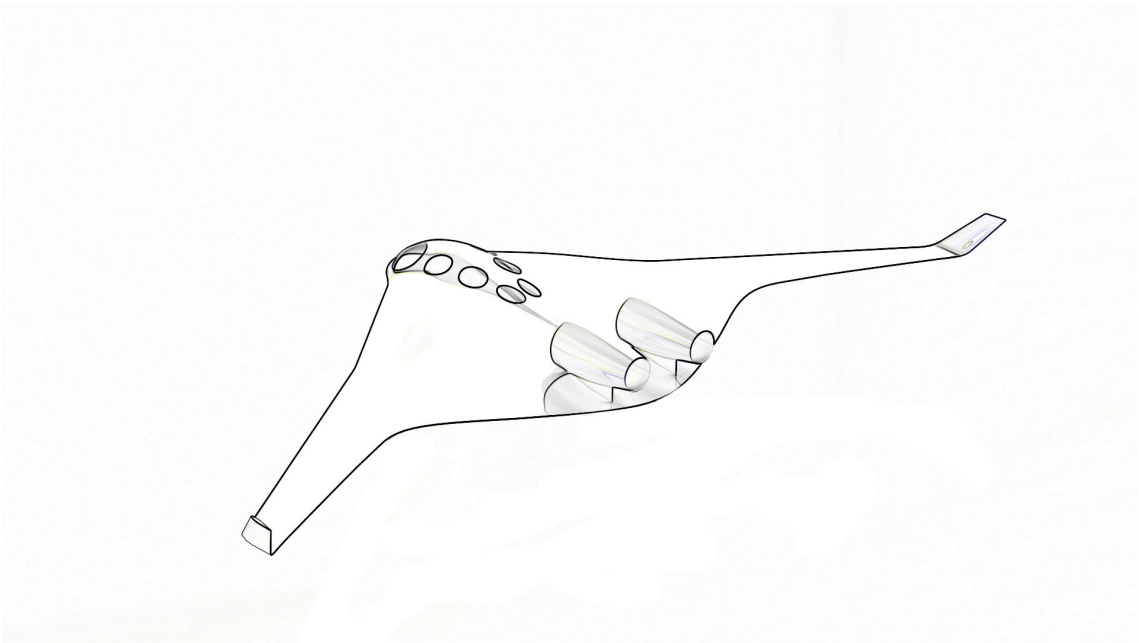
Declaration

By submitting this thesis, I declare that the entirety of this work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: June 1, 2010

Dedication

I would like to dedicate this work to:
David and Mina Roysdon, my parents,
and
'Pa-Pa' and Nancy Cobern, my grandparents.
Thank you for your love, support, and prayers.



Contents

List of Figures

Nomenclature

Physical:

b	Wing Span
c	Reference Chord
$\bar{c}$	Mean Aerodynamic Chord
m	Mass
S	Surface Area
A	Aspect Ratio
e	Efficiency
cg_p	Aircraft Center of Mass Position
I	Moment of Inertia Matrix
I_{xx}	Moment of Inertia around roll axis
I_{yy}	Moment of Inertia around pitch axis
I_{zz}	Moment of Inertia around yaw axis

Natural Constants:

ρ	Air Pressure
g	Gravitational Acceleration

Aerodynamic:

$\bar{q}$	Dynamic Pressure
C_L	Aerodynamic Lift Coefficient
C_D	Aerodynamic Drag Coefficient
C_l	Aerodynamic Roll Coefficient
C_m	Aerodynamic Pitch Coefficient
C_n	Aerodynamic Yaw Coefficient
C_x	Aerodynamic Axial Force Coefficient
C_y	Aerodynamic Lateral Force Coefficient
C_z	Aerodynamic Normal Force Coefficient

Linear Quadratic Regulator:

J	Cost Function
Q_1	State weighting matrix
Q_2	Actuator weighting matrix

Position and Orientation:

P	Position Vector
N	North Position
E	East Position
D	Down Position
h	Height
α	Angle of Attack
β	Angle of Side Slip
γ	Flight path angle
ϕ, θ, ψ	Euler Angles (roll, pitch and yaw)
i, j, k	Basis Vectors
DCM	Direct Cosine Matrix

Velocity and Rotation:

V	Velocity vector
$\bar{V}$	Velocity Vector Magnitude
u	Axial Velocity
$omega$	Angular velocity Vector
p	Roll Rate
q	Pitch Rate
r	Yaw Rate

Forces, Moments and Accelerations:

M	Moment Vector
L	Roll Moment
M	Pitch Moment
N	Yaw Moment
F	Force Vector
X	Axial Force
Y	Lateral Force
Z	Normal Force
a_a	Axial Specific Acceleration
a_n	Normal Specific Acceleration

Actuation:

T_c	Thrust Command
T	Thrust State
τ_T	Thrust Time Constant
δ_E	Elevator Deflection
δ_A	Aileron Deflection
δ_R	Rudder Deflection
δ_T	Thrust Deflection

System:

A, F	Continuous System Matrix
B, G	Continuous Input Matrix
C, H	Output Matrix
Φ	Discrete System Matrix
Γ	Discrete Input Matrix
T_s	Sampling Time
K	Feedback Gain Matrix

Subscripts:

B	Coordinated in Body Axes
I	Coordinated in Inertial Axes
W	Coordinated in Wind Axes
S	Coordinated in Stability Axes
G	Gravitational force or acceleration
T	Thrust force of acceleration
0	Static of Initial Value

Superscripts:

BI	Body relative to Inertial
BW	Body relative to Wind
WI	Wind relative to Inertial

Acronyms:

ADC	Analog to Digital Converter
BWB	Blended Wing Body
CAD	Computer Aided Design
$COTS$	Common Off The Shelf
FAA	Federal Aviation Administration
ESL	Electronics Systems Laboratory
GUI	Graphical User Interface
LQR	Linear Quadratic Regulator
$MIMO$	Multi-Input-Multi-Output
MOI	Moment Of Inertia
NSA	Normal Specific Acceleration
SPM	Short Period Mode
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle

Misc:

α	Angle of attack
β	Angle of side slip
γ	Flight path angle
δ_a	Aileron position
δ_e	Elevator position
δ_r	Rudder position
δ_{th}	Throttle position
θ	Pitch angle
ϕ	Roll angle
ψ	Yaw angle
ω_{eng}	Engine speed (power)
τ	Engine time constant
$\dot{\phi}$	Roll rate
$\dot{\theta}$	Pitch rate
$\dot{\psi}$	Yaw rate
$\dot{\omega_{eng}}$	Engine speed (power) rate

g	Gravity
h	Altitude
A	Jacobian for $\frac{dF}{d\vec{X}}$
A_n	Normal acceleration
A_y	Lateral acceleration
B	Jacobian for $\frac{dF}{d\vec{U}}$
C	Jacobian for $\frac{dG}{d\vec{X}}$
D	Jacobian for $\frac{dG}{d\vec{U}}$
I_{xx}	Roll moment of inertia
I_{yy}	Pitch moment of inertia
I_{zz}	Yaw moment of inertia
I_{xz}	Roll/yaw product of inertia
P	Angular velocity about the body x-axis
P_N	North position
P_E	East position
P_D	Down position (-altitude)
Q	Angular velocity about the body y-axis
R	Angular velocity about the body z-axis
U	Body x-axis velocity
V	Body y-axis velocity
V_T	True velocity
W	Body z-axis velocity
X_{cg}	X position of the CG in aircraft frame
Y_{cg}	Y position of the CG in aircraft frame
Z_{cg}	Z position of the CG in aircraft frame
$\dot{h}$	Altitude rate
$\vec{u}$	Control vector (used when noting deviation of a nominal condition)
$\vec{v}$	State vector (used when noting deviation of a nominal condition)
$\vec{w}$	Output vector (used when noting deviation of a nominal condition)
$\dot{U}$	Body x-axis velocity rate
$\dot{V}$	Body y-axis velocity rate
$\dot{W}$	Body z-axis velocity rate
$\vec{X}$	State vector
$\vec{U}$	Control vector
$\vec{Y}$	Control vector

Paul F. Roysdon
June 2010
Mechanical & Aerospace Engineering

Airplane Numerical Simulation for the Rapid Prototyping Process

Abstract

Airplane Numerical Simulation for the Rapid Prototyping Process is a comprehensive research investigation into the most up-to-date methods for airplane development and design. Uses of modern engineering software tools, like MatLab and Excel, are presented with examples of batch and optimization algorithms which combine the computing power of MatLab with robust aerodynamic tools like XFOIL and AVL. The resulting data is demonstrated in the development and use of a full nonlinear six-degrees-of-freedom simulator. The applications for this numerical tool-box vary from un-manned aerial vehicles to first-order analysis of manned aircraft. A Blended-Wing-Body airplane is used for the analysis to demonstrate the flexibility of the code from classic wing-and-tail configurations to less common configurations like the blended-wing-body. This configuration has been shown to have superior aerodynamic performance - in contrast to their classic wing-and-tube fuselage counterparts - and have reduced sensitivity to aerodynamic flutter as well as potential for increased engine noise abatement. Of course without a classic tail elevator to damp the nose up pitching moment, and the vertical tail rudder to damp the yaw and possible rolling aerodynamics, the challenges in lateral roll and yaw stability, as well as pitching moment are not insignificant. This thesis work applies the tools necessary to perform the airplane development and optimization on a rapid basis, demonstrating the strength of this tool though examples and comparison of the results to similar airplane performance characteristics published in literature.

Acknowledgments

I would like to thank the following people who have lead to the my growth as an engineer, both in Academia and Industry:

- To my faculty advisor, Dr. J.J. Chattot. Thank you for providing me the opportunity to attend UC Davis for my Masters degree, and providing advise to aeronautical problems during my several years at UC Davis. Without your help, I would not have had the many opportunities to grow and mature as both a student and an engineer.

- To Dr. Ron Hess. Thank you for your keen ability in conveying your thoughts clearly and concisely to me in the area of aircraft stability and control. You have encouraged me to push my limits and learn new and exciting material within the aerospace discipline.

- To Dr. "Case" van Dam. Thank you for the encouragement I needed during difficult times. You have special way of helping students both academically and personally.

- To Dr. Dan Raymer. Thank you for your mentoring, and your support of my pursuits in both Academia and Industry.

- To Mr. Douglas Meyer. I owe you so many thanks for my development professionally. You have mentored me since the start of my career, and you have provided a wealth of knowledge in the aircraft design and aerodynamics disciplines. You have also been a friend and guide, with a kind heart, willing and able to spend a little extra time to explain the solutions to engineering problems, even when the solution seemed quite trivial.

Thank you all.

Chapter 1

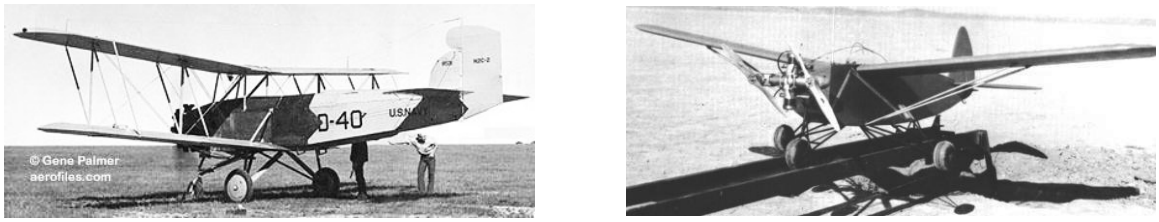
Introduction

1.1 Background & Historical Motivation

1.1.1 Emerging Markets

During the last decade, the Unmanned Aerial Vehicle (UAV) and/or Unmanned Aerial System (UAS) was the fastest growing sector of the aerospace industry. The earliest UAV's date back to the late 1930's, when the US Army converted a few surviving Curtis N2C-2 *Fledgling* (Figure 1.1.1) training biplanes to radio-controlled target drones [9] for gunnery practice. By the time the United States was involved in World War II, and the use of aerial targets had become a mainstay in industry, most notably with the RadioPlane RP-4 [10]. Radio Plane founder, Reginald Denny, started building his remotely-controlled planes in his garage, starting with the RP-1. In 1939, Mr. Denny demonstrated his RP-4 (Figure 1.1.1) variant for the US Army, who almost without hesitation, purchased 50 of his RP-4 drones. Soon after the *Radioplane Company* (now *Northrop-Grumman*) was founded, and by the end of World War II, several thousand RP-4's and RP-4 variants has been sold to the

US Army. This officially established the aerial target, and the UAV in general, as a profitable business endeavor.



(a) (b)

Figure 1.1.1: Curtis N2C-2 *Fledgling* (a), Radioplane Company RP-4 (b).

At the present, the advent of the personal computer and inexpensive electronics has evermore enabled start-up companies to design and build UAV's from a home work-shop, similar to the start of the Radioplane Company. The present budgetary constraints encountered by the military and other organizations, are another reason for the strong interest in UAV's. During the last 20 years, the economic system has encouraged the design and development of relatively inexpensive UAV's to service roles in a variety of missions from weather tracking, surveillance and strike capability, as evident in the following examples: the *Aerosonde* UAV (weather tracking) Figure 1.1.2 Ref. [2], the RQ-4 *Global Hawk* (military surveillance) Figure 1.1.3 Ref. [12], and the MQ-1 *Predator* and the larger variant MQ-9 *Reaper* (military surveillance and defense) Figure 1.1.4 Ref. [11].

Figure 1.1.2: Aerosonde weather tracking UAV.

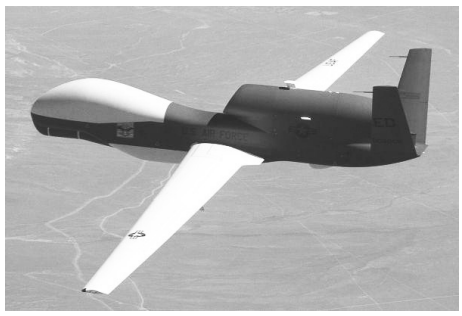


Figure 1.1.3: RQ-4 *Global Hawk* long range surveillance UAV.

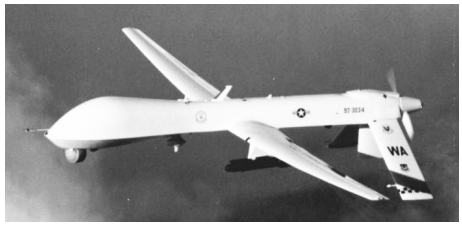


Figure 1.1.4: MQ-1 *Predator* tactical multi-mission UAV .

A primary reason for which the UAV is replacing piloted airplanes, is the UAV's capability to fly longer multi-role missions without the limitation of pilot fatigue. Presently there exists several UAV's capable of multi-day missions, and it is foreseeable that a UAV will soon be capable of single flight missions of several weeks or months. In the afore mentioned cases, the pilot remains at a home-base and can be relieved on a regular basis by taking 4-8 hour shifts with other pilots: if a pi-

lot is indeed required at all. The inherent advantage of this is two fold. First, the UAV eliminates harm to the pilot in the event of a crash or otherwise compromising situation. And second, there exists an ability to simplify the airplane systems significantly; however in some cases the pilot is replaced with larger quantities of advanced systems. Assuming the UAV systems are simplified, they provide a cost effective and low-risk solution to these and other mission requirements.

1.1.2 A Job for an Engineer

For the engineer, the computing power presently available is staggering. And the ability to design an airplane within a highly coupled environment is evermore a reality. The design cycle of an airplane during the last several decades has been on the order of months and years. Investigation of the aerodynamics alone, for example, could take several months for a single configuration. With present computers it is possible, within a few hours to a few days, to perform a multi-disciplinary optimization of a design which meets various requirements. Thus, time to completion of a design, from concept to flight-test, is on the order of weeks and months. Furthermore, for the aeronautical engineer, the numerical methods presently available allow the complete development of an airplane configuration prior to expensive and rigorous testing, such as wind tunnel testing. This provides both, calendar-time and financial benefits, allowing the engineer to test the configuration that most closely, if not exactly, represents the future flying article. The numerical techniques therefore reduce the time and cost of rigorous testing of various configurations to determine the best design for production.

1.2 Task Description & Project Outcomes

1.2.1 Software Tools

The goal of this work was to find a low-cost approach to the accelerated airplane design process by using as many free-ware or share-ware codes as possible. To validate this approach, the results of each code was compared to higher-fidelity codes or experimental data. Figure 1.2.1 depicts the process flow for a typical airplane design study.

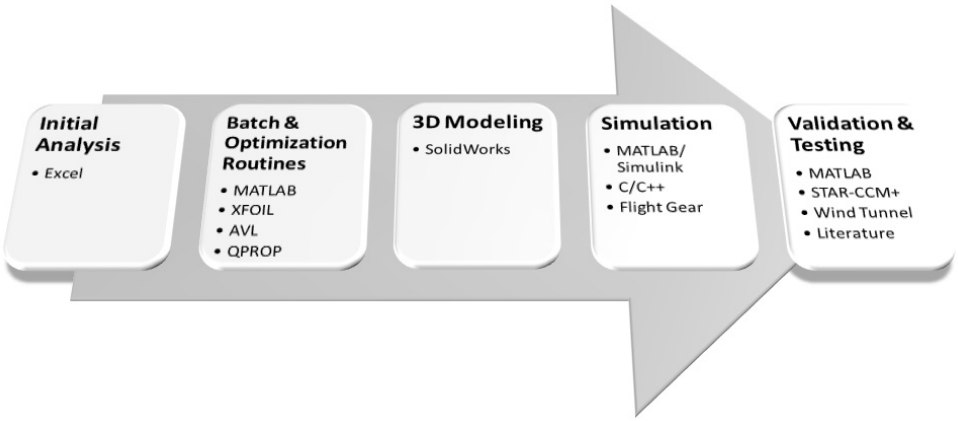


Figure 1.2.1: Airplane design calculation flow.

As seen in the flow chart, Figure 1.2.1, the development of the airplane can be performed with as few as nine programs, all of which can be used, even simultaneously, on a personal laptop computer.

- XFOIL, AVL and QPROP are open-source codes written by Mark Drela [13] [14] [16], and will be discussed in a later section.
- In this example, Microsoft Office Excel 2003 was used for the spreadsheet design, however it is entirely possible to use OpenOffice for the same analysis.

- *MatLab* & *Simulink*[33] was used for much of the analysis, batch, and optimization routines. This is an essential part of this work and there is really no substitute for this software. [1] [2]

- Any CAD package may be used, either an open-source CAD or a more robust CAD software like *Dassault SolidWorks*; the latter which was chosen for this project due to the authors prior experience in this CAD software package.

- *Flight Gear* was chosen as the visual interface for the HWIL simulator; Flight Gear is available as an open-source flight simulator from FlightGear.org.

- Any C/C++ compiler will work, several of which are available as freeware, however *Microsoft Visual Studio .Net 2003* was chosen for this investigation.

- Star-CCM+ is a CFD package, available from CD-ADAPCO, and was used to provide higher confidence in the aero model. However, any CFD package can be used, provided that the user is well acquainted with the code being used to know its limitations.

[1] MatLab is a high performance interactive software for technical computing, analysis, and program development. The MatLab environment uses .m-files, are *script-files* or *function-files* in which code is written, stored, and referenced. And though there is a very extensive library of code supplied with MatLab, the user is able to write his or her own files for further development, typically using the MatLab library for nearly all calculations. The primary reason for using MatLab in the airplane design process is the ability to perform large quantities of calculations and use the graphics engine in MatLab to produce results that may be viewed and interrogated for further analysis.

[2] Simulink, is an extension to the MatLab computing engine which allows the simulation modeling of real-world situations, e.g. the 2D trajectory and velocity versus time for a rubber ball when dropped from a kitchen table, or more complex 3D simulation of a re-entry vehicle from space. In both examples, Simulink uses a block-diagram environment to model systems and sub-systems through simple mathematical or logic blocks. In the case of airplane design and development, Simulink provides the numerical tools necessary for aerospace system design, integration, and simulation. It is capable of nearly everything from environmental models to equations of motion, and from gain scheduling to 2D and 3D animation.

A General Note:

Wind tunnel testing should be carried out to validate the numerical models. However, as the focus of this work is the development of the numerical tools necessary prior to testing, wind tunnel testing is listed here as a strong recommendation.

1.2.2 The Art of Airplane Design

Airplane design is a compromise of requirements, design, trade studies, and analysis (see Figure 1.2.2). It is a process of multi-disciplinary optimization and development against a set of rules or guidelines either imposed by a national agency, like the Federal Aviation Administration (FAA), or as a list of requirements from a customer. The entire process is really three-fold: conceptual design, preliminary design, and detail design. *Conceptual design* is the initial stage of development where the requirements are set against design metrics such as: sizes, geometries, manufacturing capabilities and/or available materials, (see Figure 1.2.3). *Preliminary design* is the intermediate stage where the design is nearly frozen, and more comprehensive —and typically more expensive— design studies can be made, e.g. CFD analysis, wind tunnel testing, structural analysis. *Final design* is where the design-for-manufacturing begins, and all part details are defined, and manufacturing planning and processes are implemented.

As a guideline, the methods introduced by Raymer [42] and Roskam [45] were used to make initial conceptual design decisions. Both authors discuss in great detail the recommended methods by which the engineer should design an airplane in

the *conceptual*, to *initial*, and *final* design stages. This work will follow these guidelines, presenting detailed examples of implementing these methods into the *Airplane Numerical Simulation for the Rapid Prototyping Process*.

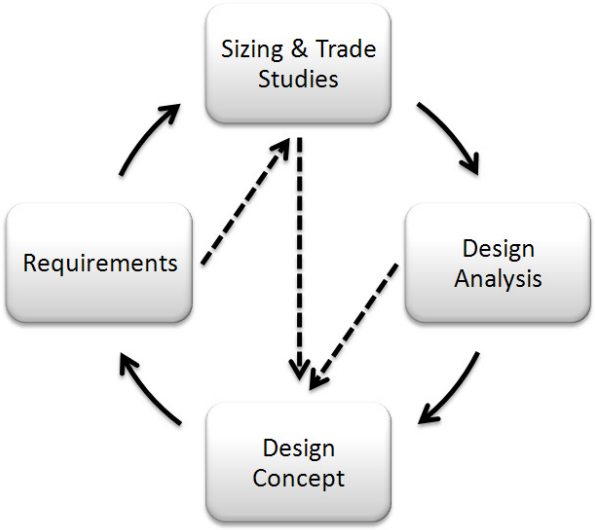


Figure 1.2.2: Airplane design wheel [42].

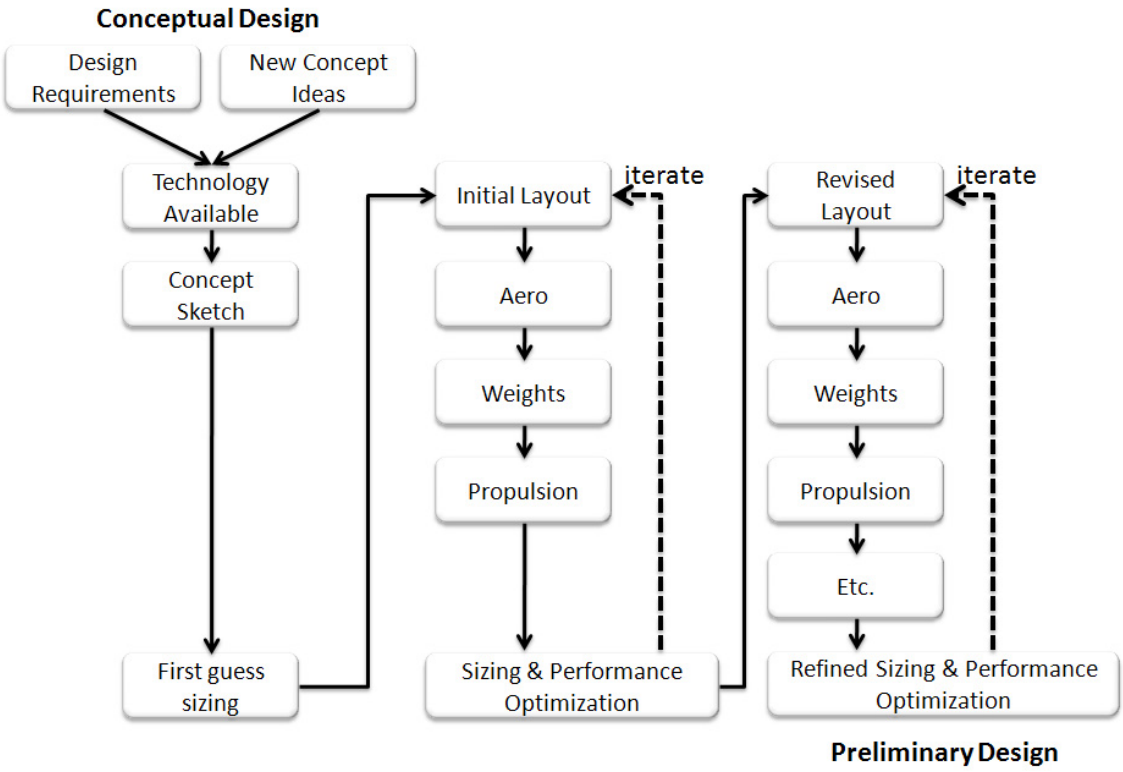


Figure 1.2.3: Airplane design flow [42].

1.2.3 The Expected Results

This work addresses the ability to develop an airplane on an accelerated basis, presenting the methods necessary to meet the objectives within the limits of an aggressive schedule. The goal of this work is to employ the use of presently available computer codes and methods for airplane design and analysis, by developing a numerical structure which communicates between individual codes in a variety of modes: batch, optimization or error analysis modes. Methods for numerical analysis of a general airplane configuration are addressed, cross examined, and compared to supporting data in aerospace literature. Common engineering software, such as MatLab and Microsoft Excel ^{TM} will be presented, as well as numerical codes like AVL by Drela [14],

XFOIL by Drela [13], and Digital DATCOM by the USAF [22], among others.

The result, or capstone, of the investigation is a complete six-degrees-of-freedom (6DOF) non-linear simulator. A high-fidelity 6DOF simulator provides the engineer the numerical power to test a vehicle in a simulated environment prior to flight test. With this numerical tool, the engineer can explore the flight envelope to predict flight capabilities or abnormalities. Furthermore, the 6DOF simulator provides a tool on which to perform statistical testing and analysis, for example: gust loads of random sequence, magnitude and direction, can be modeled and tested during take-off or landing conditions. This enables the engineer to determine the failure modes of the airplane in a variety of cases. The possibilities for testing are too numerous to mention, but the author hopes that this brief description will relate to the reader, the value and necessity of such numerical tools in the rapid prototype process for airplane.

The example airplane for this numerical approach to airplane design, is the authors original design of a Blended-Wing-Body scale demonstrator [3]: a comprehensive research investigation performed during the author's research sabbatical at the Von Karman Institute for Fluid Dynamics (VKI), in Belgium. This design will compare the numerical approach of this work to the experimental data provided in a NASA report on a comparable Business Jet configuration [48].

[3] a $\frac{1}{16}^{th}$ scale, semi-autonomous demonstrator for flight performance analysis

Airplane Configuration

Blended Wing Body configurations are based on the concept that the area rule provides a better aerodynamic performance from shapes when fuselage, wings, empennage, engine pods, etc., get integrated into a single shape resulting in a smooth delta shape wing body configuration. This configuration has been shown to have superior aerodynamic performance - in contrast to their wing-and-tube fuselage counterparts - and have reduced sensitivity to aerodynamic flutter as well as potential for increased engine noise abatement. The resulting inner space is demonstrated to cover a larger volume than the classic fuselage supported on cantilevered wings and a crossed T tail. Additionally, the interior structure of the vehicle can be simplified with a monocoque structure, taking advantage of modern composite materials. Of course without a classic tail elevator to damp the nose up pitching moment, and the vertical tail rudder to damp the yaw and possible rolling aerodynamics, the challenges in lateral roll and yaw stability, as well as pitching moment are not insignificant. The work at VKI consisted of designing attempts using lower order methods and vortex lattice based methodology to get a comprehensive understanding of the aerodynamic performance. Followed by more accurate CFD methods to fine tune the performance. A 1/16th scale model was also designed and will be tested (at a later date) in the wind tunnel to validate the CFD based modeling strategies. The BWB design is shown in Figures 1.2.4 and 1.2.5.

This thesis work will seek to obtain and optimize a stable BWB airplane design, which exceeds the performance of the Lear 23 based on the following criteria:

- Payload: 9,000 Lbs. (including fuel), Maximum takeoff weight: 12,500 Lbs.
- Cruise speed: Mach 0.73, Maximum speed: Mach 0.82
- Range: 1,376 nm.
- Service ceiling: 45,000 ft.
- Stability characteristics comparable to Reference [48]

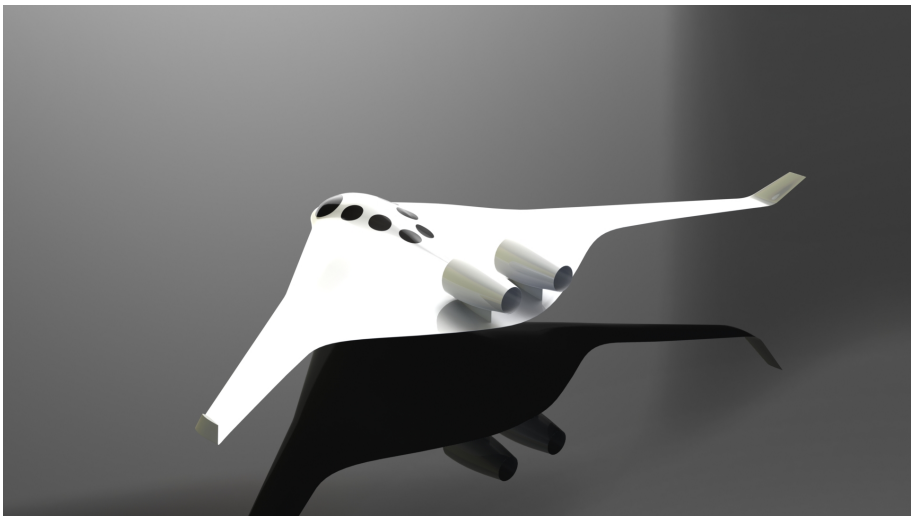


Figure 1.2.4: The Roysdon BWB concept for Business Jet-class operation.

1.3 Thesis Outline

This document is presented in the following series of chapters; see Figure 1.3.1. Chapter 1 introduces the subject matter, provides a background on airplane design, and introduces the software which was used in this thesis work. Chapters 2-7 discuss the

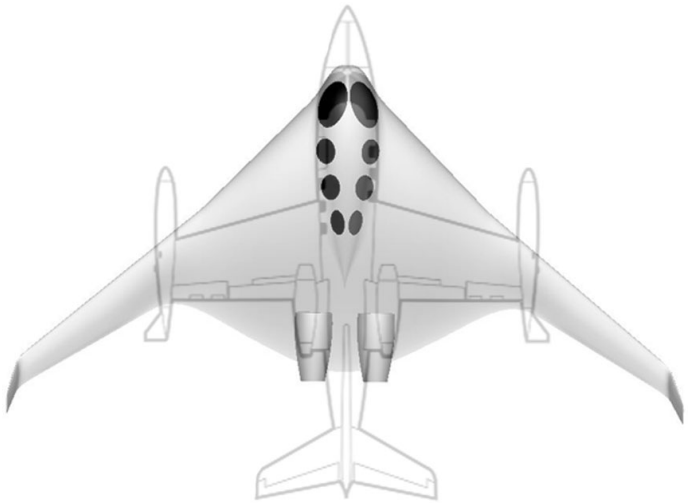


Figure 1.2.5: The Roysdon BWB concept overlay with a Lear 23.

various steps necessary to develop a 6DOF non-linear simulator, as well as the numerical tools used and/or created for accelerating the airplane development process. Chapter 8 presents the final goal of this work, the 6DOF non-linear simulator. In Chapter 8, the various components related to simulation will be discussed in detail. Chapter 9 concludes the work, and provides recommendations for future work.

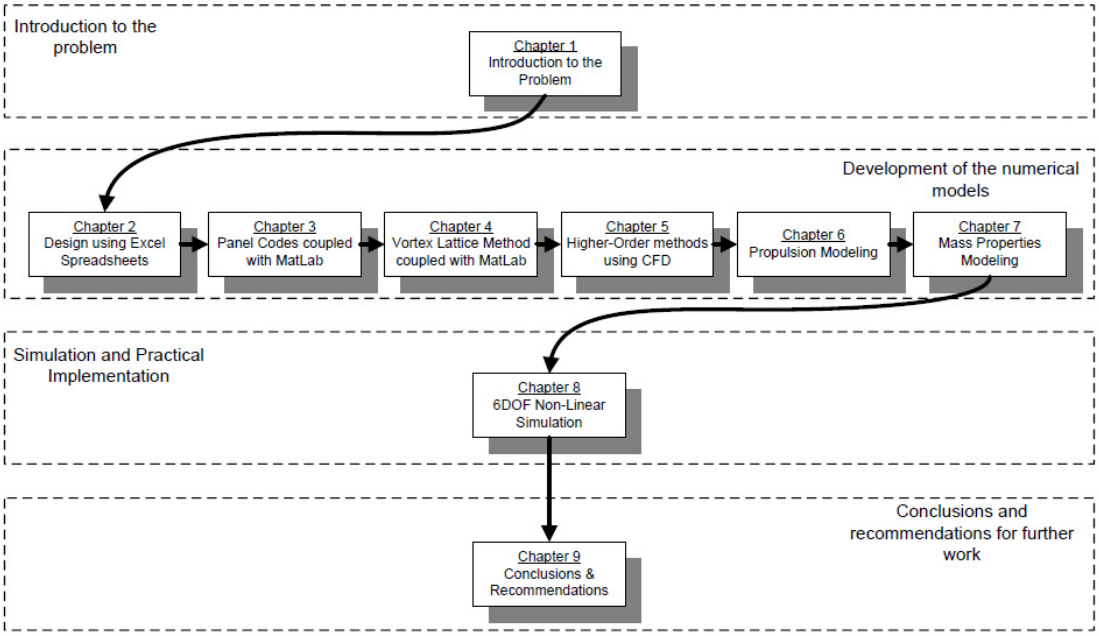


Figure 1.3.1: Thesis Outline.

Chapter 2

Spreadsheets for the Aero Engineer

2.1 Background

Based on the work of Raymer [43], and Hays [20], *Microsoft Excel*^{TM} was chosen as a tool to facilitate the design study, providing maximum design flexibility. In application, the author assembled a rather extensive Excel spreadsheet which is capable of calculating and simultaneously updating the following quantities:

- Aerodynamics
- Static & Dynamic stability
- Structural loading & limits
- Mass properties
- Rotational moments of inertia
- Propulsion requirements or capabilities
- Flight performance characteristics
  - Turn Rate and Radius vs. Velocity
  - Climb Rate and Distance vs. Velocity
  - Glide Rate and Distance vs. Velocity

- Descent Rate and Distance vs. Velocity
- Maximum power envelopes
- Speed vs. throttle settings
- Range and Endurance estimations.

Both authors [43] [20], provide examples of the ease with which the aeronautical engineer can develop new airplane designs and perform trade studies with a simple program such as Excel. Furthermore, with its extensive capabilities which combine spreadsheet programming with Visual Basic macros, it has the potential for very complex computing.

2.2 Spreadsheet Application

The author's spreadsheet uses the equations listed in Raymer's book [42] almost exclusively, and the spreadsheet is separated into three primary modules; Static, Dynamic and graphical results (see Figure 2.2.1 below).

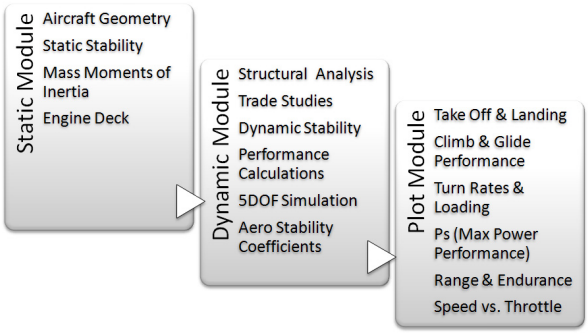


Figure 2.2.1: Block Diagram of the spreadsheet calculation flow.

The *Static Module* requires inputs for airplane geometries; wing geometries like span and chord (fig. 2.2.2), tail geometry and location relative to the wing, and fuselage geometry, as well as mass properties and propulsion characteristics.

The *Dynamic Module* requires inputs of altitude, temperature and velocity ranges.

The *Plotting Module* provides the results of both the static and dynamic information and provides trade studies of variations of the entered airplane properties. The plotted results are common to *Airplane Users Manuals*, such as the Cessna 152 & Cessna 172 Skylane. These plots provide information for flight planning, and/or comparative information to other airplanes. For example, one could use the plots to compare the Specific Range and Specific Endurance from one airplane to another, and for at various altitudes and payload capabilities. See Figures 2.2.3 thru 2.2.6.

Wing

Airfoil =	SD7037	
C_{lmax} =	1.40	2D airfoil
t/c =	0.10	
S_{ref} =	19.861	sq ft
Root Chord =	14.000	in.
Tip Chord =	8.000	in.
(span) b =	260	in. >> ft. 21.67
AR (Aspect Ratio) =	23.64	(Table 4.1 Raymer)
λ (Taper Ratio) =	0.714	(Fig. 4.22 Raymer, 0.5 is near elliptic optimum)
Twist =	-2.0	deg.
(incidence) i_w =	1.0	deg.
Y_{bar} (y-location of MAC) =	61.39	in.
Λ_{LE} (LE sweep) =	0.00	deg.
cos $\Lambda_{0.25c}$ =	-0.66	deg.
Dihedral =	2.5	deg. (Table 4.2 Raymer)
C_{Lmax} =	1.26	3D wing (Eq. 12.16 Raymer)
C_{bar} =	12.111	in.
ΔC_{Lmax} =	0.0	High AR (Figure 12.9 Raymer)
W/S (wing loading) =	1.51	**lbs**/sq.ft.
W/S (wing loading) =	24.17	**oz**/sq.ft.
Aileron chord =	30	% c from TE
span =	50	% half-span
Flap chord =	30	% c fromTE
span =	50	% half-span

Figure 2.2.2: Wing Tail Sizing.

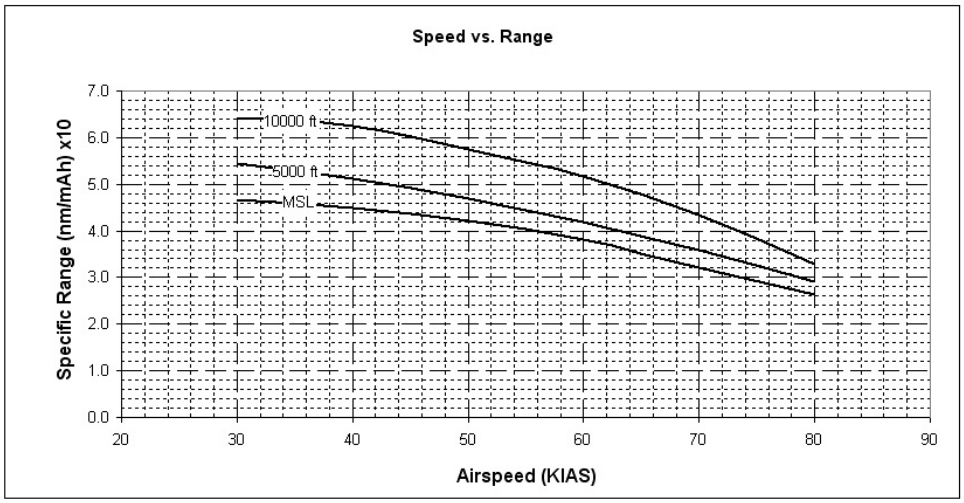


Figure 2.2.3: Specific Range results.

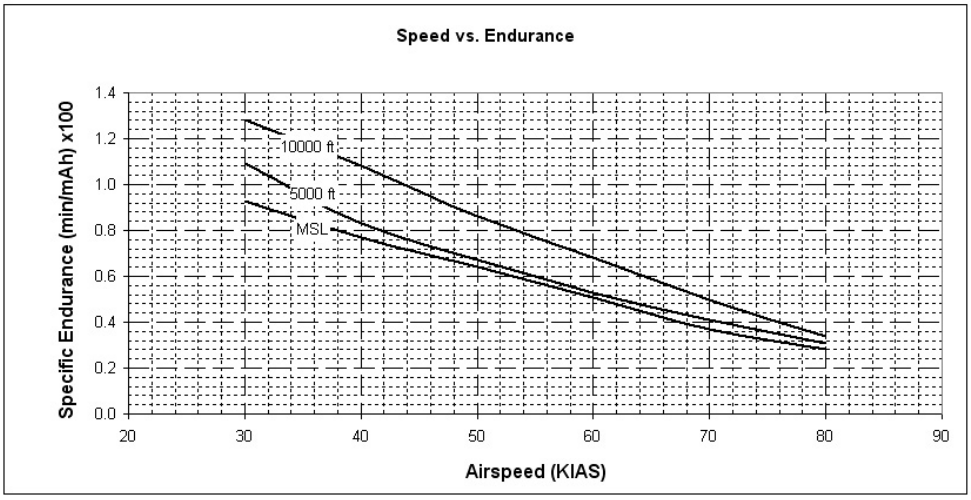


Figure 2.2.4: Specific Endurance results.

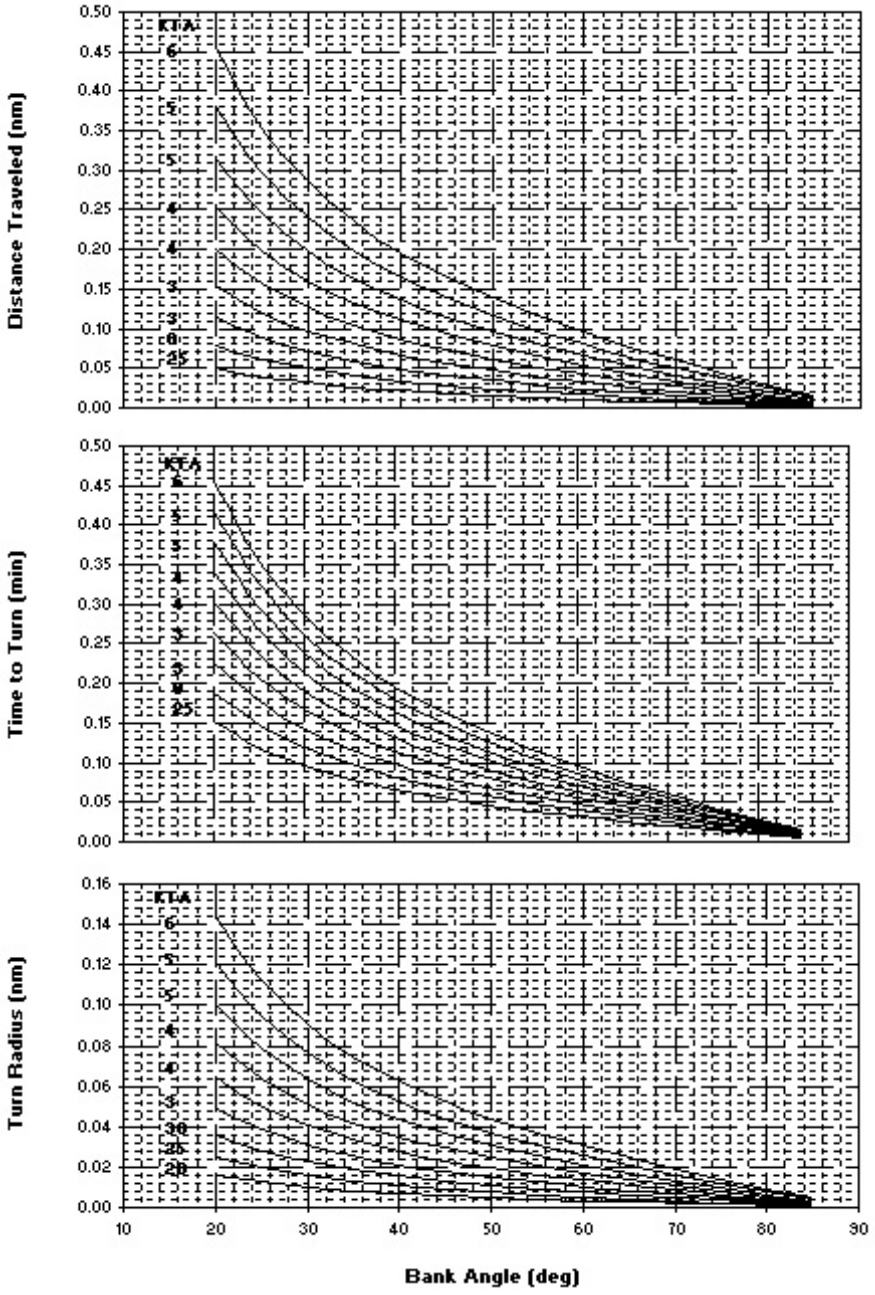


Figure 2.2.5: Time to turn.

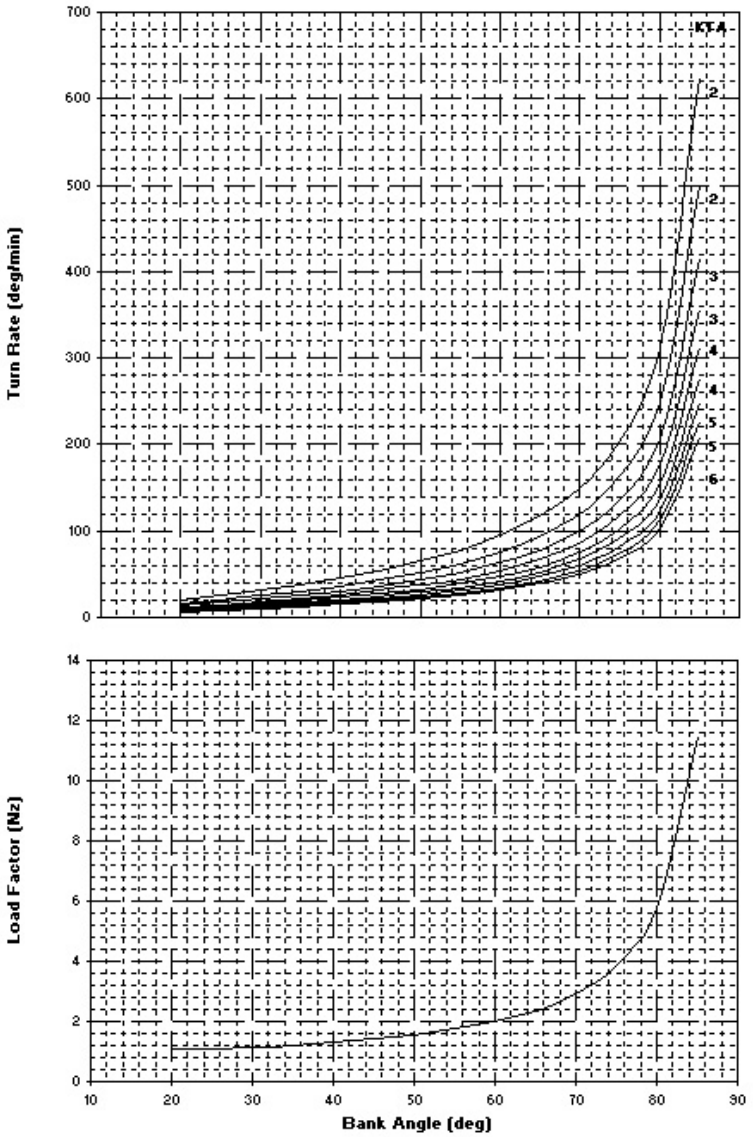


Figure 2.2.6: Turn vs. Load factor.

In the initial design stage, this spreadsheet provides nearly immediate information with very little input. However, the accuracy increases as the information provided to it by higher-order methods is applied. For example; a result of the dynamic module is a 5DOF simulation of trim conditions. This trim simulation is increasingly more accurate as higher fidelity data is supplied to it. This high-fidelity data is supplied

via a table entry on a separate spreadsheet, wherein data can be entered from VLM, CFD or Wind Tunnel results. Refer to Appendix A, for a comprehensive look at the arrangement of the spreadsheet tool.

Chapter 3

Airfoil Selection via 2D Panel Code coupled with MatLab

3.1 History of panel codes

Panel methods date back to the early days of aerodynamics, when engineers were trying to solve the lift over a wing surface by means of numerical solutions, see the following section 4 for literature references. Generally this method is applied to airfoils where there is a known thickness, length, camber, etc. In the 1930's it was found that by applying a system of equations to the flow field surrounding the airfoil, the lift generated could be determined by descretizing the airfoil into a series of panels. Furthermore, by representing the flow field as a vortex sheet, Figure 3.1.1, or more specifically as a series of source-doublet combinations, Figure 3.1.2, numerically the flow could be represented quite well.

This work uses the XFOIL code developed by Mark Drela [13], which, unlike most

inviscid panel methods developed during the 1970's and 1980's, XFOIL is a fully-coupled viscous/inviscid solver. XFOIL is a 2D panel code which uses 2D boundary layer integral equations with corrections which can accurately predict boundary layer separation and re-attachment for low Mach number flow at small angles of attack; Mach number less than 1, and angles of attack where separation is limited, up to and including Cl max for most airfoils. XFOIL also employs inverse modes and geometric manipulation modes, allowing the user to develop, modify and test a large variety of airfoil geometries, with a high confidence level. In many cases, the solutions which result from the XFOIL flow solver, are more accurate than available literature, like the NACA sections provided in Ref [1]. This is because the numerical flow is generally pure, and without the disturbances and model inconsistencies which existed in the re-circulating wind tunnel used for the NACA airfoil tests.

As this is just an introduction to the history and theory, the discussion will close here. The reader is encouraged to refer to the many books and on-line literature which cover this topic in great detail.

3.2 Coupling XFOIL with MatLab

When selecting an appropriate airfoil(s) for a new airplane design, XFOIL provides a fast and accurate result within a high degree of confidence. As the user familiarizes himself or herself with the various modules within XFOIL, it is increasingly apparent that airfoil selection, in a timely manner, is quite difficult without prior knowledge of a particular variety or family of airfoils from which to start the investigation. To

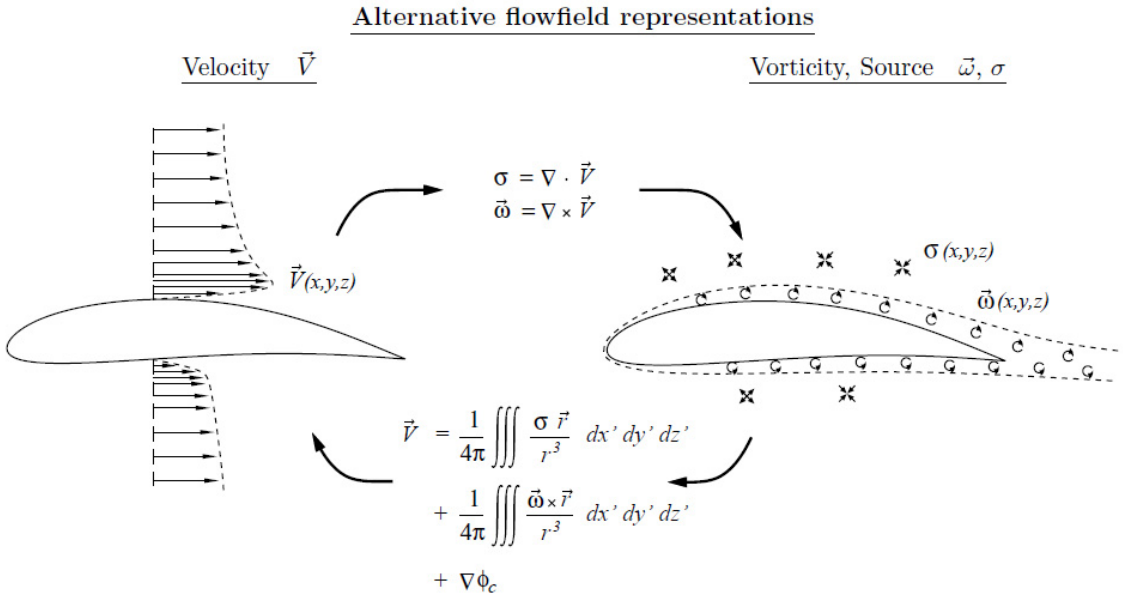


Figure 3.1.1: Flowfield representation through vortex sheet [15].

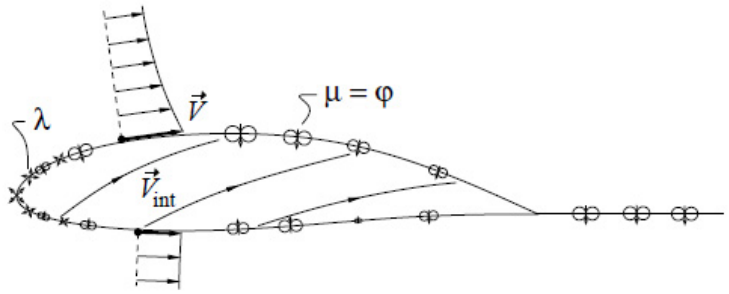


Figure 3.1.2: 2D Source-Doublet application over an airfoil [15].

alleviate this workload, the author assembled a database of nearly all known airfoils from the simple Clark Y, to the advanced NASA Langley Natural Laminar Flow (NLF) family of airfoils. This database was obtained through the UIUC website [52], and saved to a directory on a local drive of the author's personal laptop computer. This facilitated the fast access to a large variety of airfoils without the need for an Internet connection and web interface for downloading airfoil coordinate files.

Following this, a set of ASCII (text) files were created to run the XFOIL executable in "silent-mode", e.g. without the Command Window visible, which greatly improves the calculation time from one airfoil to the next. First a definition file, *xfoil.def*, was written to set the global conditions for XFOIL, refer to Appendix B. Next a batch program file, *run_xfoil.bat*, was written for MatLab to be capable of setting — via a DOS command — the airfoil files to be executed by XFOIL. A file, which is also necessary for each run, is a data file, which sets the local commands for XFOIL. This file, labeled *xfoil_run_def.dat*, is created by MatLab and updated after each XFOIL run. Finally, a MatLab script was created to run XFOIL — either iteratively or sequentially — saving the results of each run case in tabulated form to a .mat file (or Excel file), and plotted results to a Microsoft Word document. An optional flag was used in MATLAB to optionally save the calculated results from a "final selection", to the *Roysdon_Aero_Analysis.xls* spreadsheet for the 5DOF reference for airfoil characteristics of C_l, C_d and C_m.

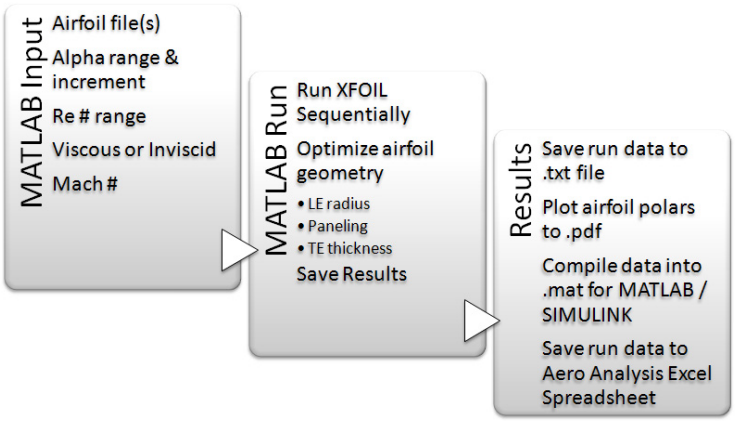


Figure 3.2.1: Block Diagram of the XFOIL to MatLab interface.

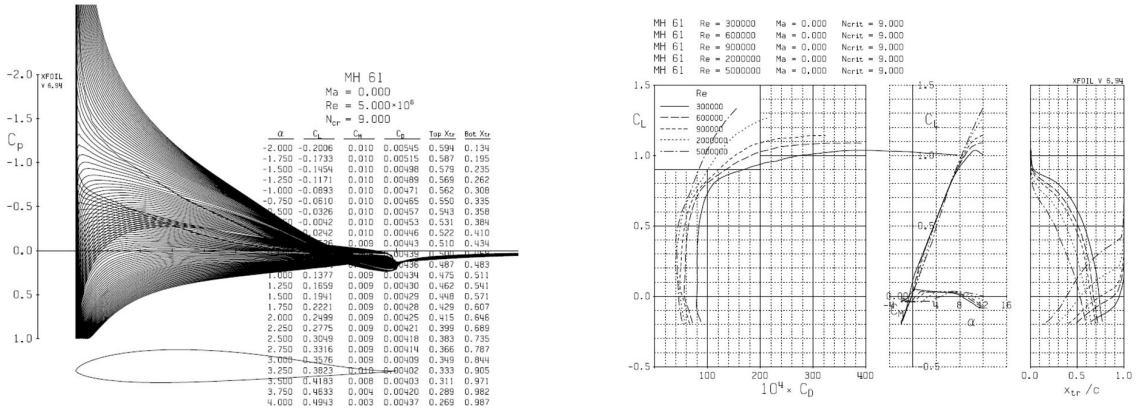
The resulting MatLab script can run XFOIL in either a batch or optimization mode. The batch-mode allows the selection of several airfoils, and is capable of running a sweep of alpha values and Reynolds Numbers to obtain a complete picture of the selected airfoils capabilities and limitations. Conversely the optimization mode is set such that the C_l, C_d and C_m characteristics are the drivers for the optimization and the airfoil can be selected from a directory of airfoils. This mode then runs in a batch-mode as described before, to determine which airfoil best meets the requirements. Additionally, the MATLAB script, for both routines, will optimize the individual airfoil coordinate files by assessing the panel distribution and updating the paneling on the leading edge and trailing edges if necessary.

3.3 Results of the Coupled Routine

To validate the advantage of this MatLab routing, the following example is provided: A typical XFOIL session, run from the command prompt, and evaluated for a single airfoil over a series of alphas and Reynolds numbers, can take in excess of 20 minutes for a proficient user. To then add additional airfoils and cross-compare the plotted results either individually or in a spreadsheet, the time required increases considerably. In contrast, the XFOIL Batch can be run for 35 airfoils, for a series of five Reynolds numbers, and a sweep of alpha values of -6 to 16 degrees at 0.5 degree increments, in less than five minutes on a personal laptop computer. The same calculation performed manually at the command prompt would have taken nearly 12 hours if performed sequentially, without hesitation or breaks. Clearly the XFOIL batch-mode has proven its value in the design process. Furthermore the batch mode can simultaneously optimize the airfoil coordinate spacing and tweak the airfoil ge-

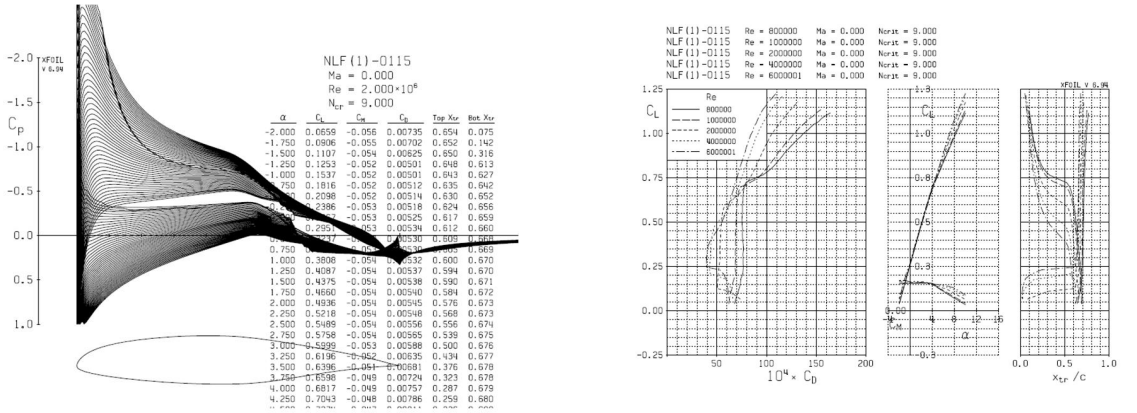
ometry as desired, and still run in just less than 5 minutes.

The following figures 3.3.1 and 3.3.2, were produced by the coupled MatLab-XFOIL code in less than one minute on a personal laptop computer.



(a) (b)

Figure 3.3.1: MH-61 alpha sweep (a), MH-61 coefficient polar (b).



(a) (b)

Figure 3.3.2: NLF-0115 alpha sweep (a), NLF-0115 coefficient polar (b).

Chapter 4

Selection of airplane planform geometry via Vortex Lattice Method coupled with MatLab

4.1 History of vortex lattice methods

The first vortex lattice methods (VLM) were developed in the 1930's, but were not applied until the early 1960's when computers made the calculations more practical. During the 1970's [38] the VLM process saw great development and growth with several publications from NASA Langley engineers, Gloss [28], Herbert [29][21], Margason [32] and Lamar [28] [29] [21] [32].

The vortex lattice method, similar to the panel method, provide the engineer a numerical tool for calculating the aerodynamics of 3D geometry in a 2.5D sense, meaning, using 2D planar surfaces to represent 3D geometry. Vortex lattice methods

are based on solutions to Laplaces Equation, and are subject to the same basic theoretical restrictions that apply to panel methods. The simple comparison follows:

Panel Method vs. VLM Similarities

- Singularities are placed on a surface.
- At the control points, the non-penetration condition is imposed.
- The singularities are solved as a system of linear algebraic equations.
- No limitations on surface thickness

Panel Method vs. VLM Differences

- 3D Surfaces are represented by 2D planar surfaces, and thickness is generally ignored (thin airfoil theory)
- Assumes that the surfaces used, represent lifting surfaces.
- Boundary conditions (BC's) are applied on a mean surface.
- Singularities are not distributed over the entire surface

Drela [15], defines the similarities of the source-doublet sheet to the vortex ring sheet in Figure 4.1.1. In the application to a 3D wing; the wing can be represented as a matrix, or *lattice*, of vortex ring elements as in Figure 4.1.2, and the velocity vectors can then solved by solving the Laplace Equation.

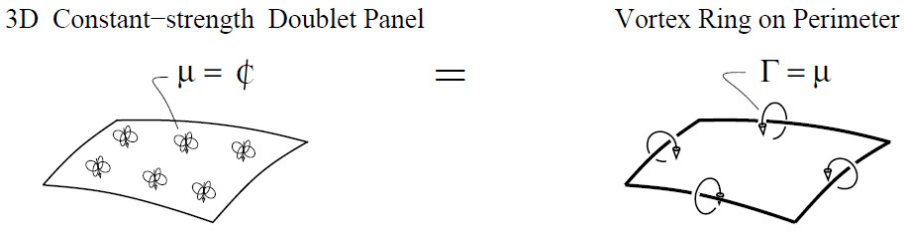


Figure 4.1.1: Comparison of 3D Source-Doublet to 3D Vortex Ring [15].

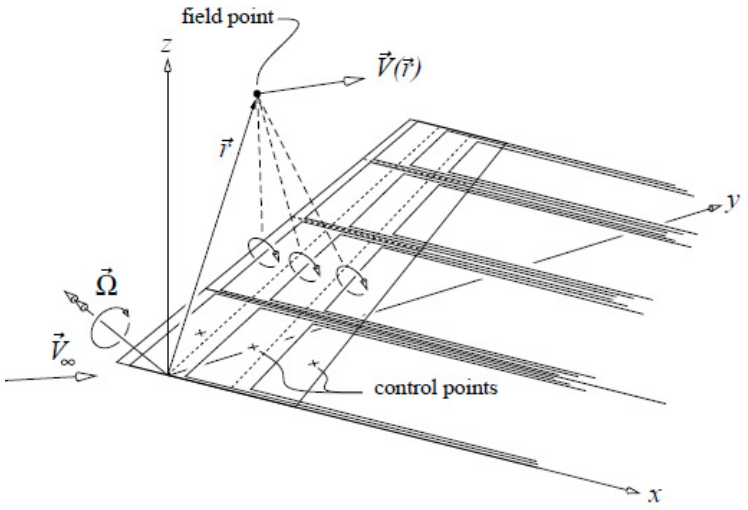


Figure 4.1.2: Vortex Lattice wing [15].

4.2 Coupling AVL with MatLab

Two vortex lattice method codes were used for this analysis, Athena Vortex Lattice (AVL), written by Mark Drela [14], and VLAERO+, available through Analytical Methods, Inc. [3]. The advantage of of AVL is that it is open-source and relatively easy to use in a batch-mode as previously demonstrated with MATLAB in the XFOIL analysis. However, AVL, though a robust code with excellent capabilities, has limitations in airplane geometries and velocities; limited to Mach = 0.3 with limited Prandtl-Glauert corrections up to Mach=0.6. In contrast, as one might expect from a commercial software package, the VLAERO+ code is quite capable batch-mode and a variety of geometries and flow conditions with the ability to import compressible flow results to tune the numerical model and allow stability calculations to be made in supersonic flow. However, due to the flow regime of this author's work, the added benefits of the VLAERO+ code where merely an exercise in validating the AVL results, and therefore will not be discussed further.

Similar to the XFOIL-MatLab routines discussed previously (Ch. 3), the AVL-MatLab routines were designed with similar goals in mind: the code needs to be capable of running a batch process to obtain information about a design in a timely fashion, and further optimize the design as needed. First, the main file for the AVL code, *name.avl* (Refer to Appendix B), is a geometry file much like the airfoil geometry file, or coordinate file, as used in XFOIL. Therefore, some information of the airplane geometry needs to be defined in a quantitative sense of X, Y, Z coordinates. This can either be performed by hand or through the use of a CAD program, first by outlining the planform of the proposed airplane and then measuring distances from some common origin to each location of major design details: e.g. the location of the wing root airfoil (x,y,z), the wing tip airfoil (x,y,z), the aileron location (x,y,z) , etc. Though necessary, this can be tedious and time consuming. To handle this situation, a parametric modeling program was developed by the author, whereby the airplane could be defined in the *Roysdon_Aero_Analysis.xls*, or the airplane could be modeled in a CAD program. The final x,y,z locations could then be imported into MatLab, which could then create the AVL geometry file. Defining the airplane characteristics in the Excel file is much easier to implement, however the advantages are obvious in the more advanced technique with the incorporation of the CAD software.

With the AVL geometry file defined, the mass properties file, *name.mass* (Refer to Appendix B), should be created using the assumptions tabulated in the mass properties section of the Excel spreadsheet. The remaining file for AVL, *name.run* (Refer to Appendix B), sets the run conditions and is created by MatLab and updated upon each iteration.

Finally, the MatLab script can be presented and discussed. This script was designed to run in both, batch and optimization modes, providing the user maximum flexibility in his or her design. The batch-mode is created such that the airplane can be run through a sweep of the following variables, either individually or sequentially:

- alpha
- beta
- velocity
- altitude
- control surface deflection
  - aileron
  - flap
  - elevator
  - rudder

Or in the case of a vee-tail a combination of rudder and elevator.

The result is an ASCII file containing the flow conditions as well as forces and moments for each run case. Ideally, the batch mode would be run to create 3D and 4D look-up tables for a simulator. For example for the drag coefficient look-up table CD $= [\alpha, \beta, velocity]$, where α = [-6:2:12], β = [-6:2:6], and three velocities, producing a C_D value for each alpha and beta at each individual velocity. The 4D table would be delta coefficients, for example the delta coefficient for drag due to elevator deflection, $C_{D\delta e}$, where the elevator deflection is [-10:2:10], and the look-up table would resemble $[\alpha, \beta, velocity, \delta_e]$.

From these result files, the script creates surface plots of all of the cases and saves the results to a Microsoft Word document for quick reference. This becomes a valuable aid in determining if the results are as expected, i.e. stable or unstable. The MatLab script is also designed to save the 6DOF results to a MatLab reference file, *aerodeck.mat* (Refer to Appendix B), for future use as aerodynamic values in a linear or non-linear 6DOF simulator. This last point is paramount in the creation of a 6DOF simulator, as the values of a well-posed or well-tuned VLM model will provide highly accurate results from which control laws can be developed in a linear simulation; that is until more accurate data is available via CFD, wind tunnel testing or flight testing. The user must keep in mind however that VLM code is limited to attached flow, therefore the resulting data should be used with caution until higher-fidelity data can be used to tune the VLM results. Within the linear region, the airplane design can be further investigated and compared to known values and quantities. To supply a quick reference, a table was built (Refer to Appendix A) which contained the 6DOF aero coefficients based on the information provided in Nelson [39] and Roskam [46] and can be seen in the figure below, Figure 4.2.1.

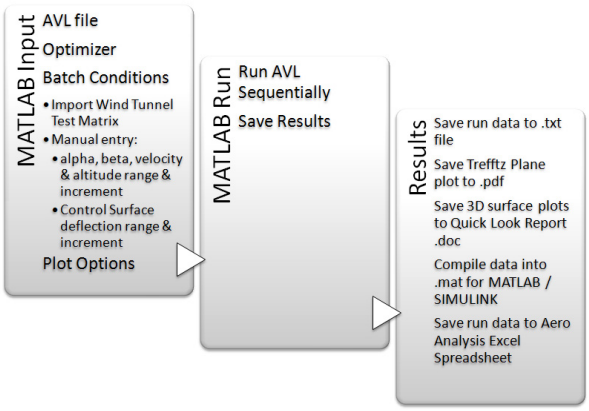


Figure 4.2.1: Block Diagram of the AVL to MatLab calculation flow.

In the case of the optimization, a greatest descent optimization algorithm is used against a set of design variables, or design space, and the optimization is carried out. The primary constraints on the optimization are wing and tail geometries, such that a specific quantity, like endurance, can be met. However this is strictly an exercise, as the real optimization occurs later in the airplane design process when more detailed information on mass properties and propulsion are known. A full discussion of the optimization routine is out of scope for this work, however, examples can be found in the literature for similar routines coupled to VLM codes, though not specifically AVL.

4.3 Results of the Coupled Routine

Again, the results of the batch-mode are staggering. For example, a typical AVL session run from the command prompt — provided that the geometry, mass and run files are already set-up, which in itself can take nearly a full 8 hours to complete — for a single run condition at a discrete alpha and beta values, can take up to 5 minutes for a proficient user to run, evaluate and plot the results. In contrast, the AVL Batch can be run for a sweep of the above listed parameters, a total of 16,000 test cases for the test-case airplane, in less than 10 hours on a personal laptop computer. The same calculation performed manually at the command prompt would have taken over 55 days if performed sequentially, without hesitation or breaks. Clearly the AVL batch-mode, like the XFOIL example given previously, has proven its value in the design process for the Aeronautical Engineer. Of course smaller data sets are also possible, and more probable early in the design stage, this is just an example to demonstrate what is typically needed for a 6DOF non-linear simulator and the time that is required to build the simulator. As before, the results can be exported to

the *Roysdon_Aero_Analysis.xls* spreadsheet for tuning the inputs of the spreadsheet simulator.

Figure 4.3.1 shows the configuration which was investigated for this analysis.

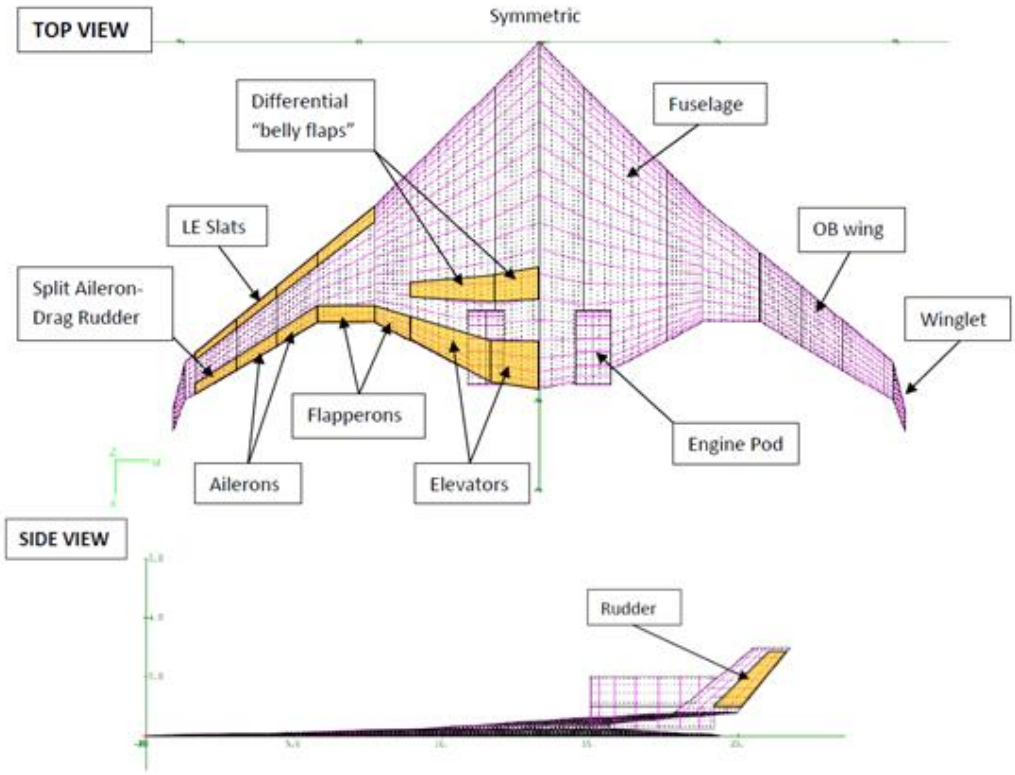


Figure 4.3.1: AVL model of the Roysdon BWB.

The following results are shown as a 3D surface plot for the Yawing Moment coefficient as a function of α and β for three different velocities. And the second plot is the delta coefficient (for yawing moment) with respect to the individual control surfaces. These variations over velocity are of particular interest when creating a 6DOF non-linear simulator and the autopilot gain schedules, specifically when the coefficient changes with increasing dynamic pressure, q. This allows the 6DOF simulator

to interpolate between velocities to generate the desired flight response. Of course, as the flight regime transitions through Mach 1, the separations between velocities must be much tighter to provide a more accurate representation of the coefficients at or around Mach 1. For the figures below, the velocity spacing was:

$$\delta_{vel} \;=\; [60,\; 100,\; 140] \;\; knots \tag{4.3.1}$$

Though the velocity spacing in this case is not necessary due to the dynamic pressure remaining nearly constant. The MatLab-AVL code was developed to be flexible, such that it could be used in slow UAV simulation, or transonic business jet simulation [1]. In the case of a simulation of a transonic to supersonic vehicle, the table would be similar to:

$$\delta_{vel} \;=\; [0.20,\; 0.50,\; 0.70,\; 0.80,\; 0.85,\; 0.90,\; 0.95,\; 0.98,\; 1.0,\; 1.1,\; 1.2,\; 1.5] \;\; Mach \tag{4.3.2}$$

Though all 6DOF coefficients were generated, the following subset will be shown as examples.

In Figure 4.3.6 (a), the top curve represents the predicted lift coefficient at stall, $C_{l\ stall}$, the second curve is the local lift coefficient at zero angle of attack, C_l, and the third is essentially the total lift distribution $C_l \frac{c}{c_{ref}}$. Where c is the local chord length, and c_{ref}, also commonly known as $\bar{c}$, is the mean aerodynamic chord.

Figure 4.3.6 (a) also presents the results of the MatLab-AVL optimization routine, where the result is a compromise of airfoil geometry, airfoil thickness, wing sweep and

[1] Note that if the transonic case were to be evaluated, the author recommends that VLAERO+ be used and tuned with either CFD or Wind Tunnel data.

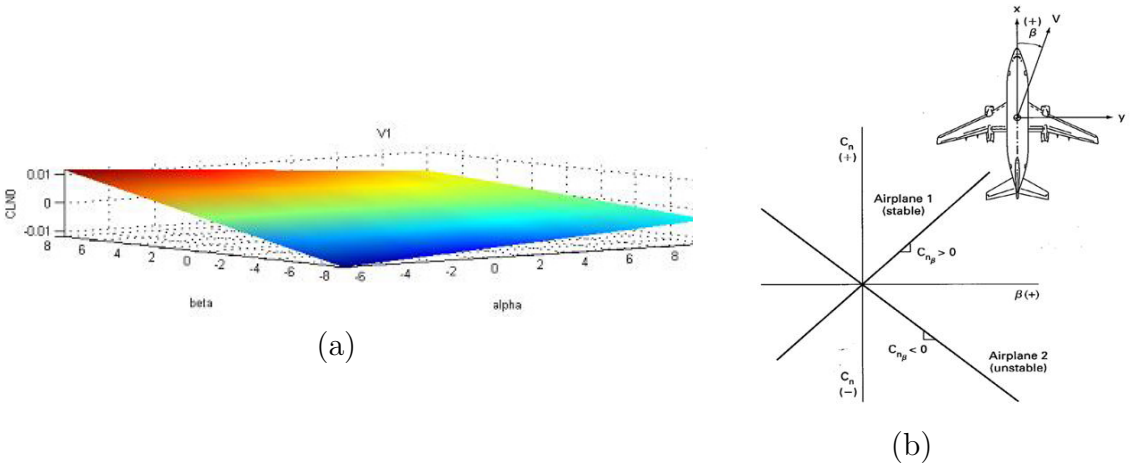


Figure 4.3.2: AVL results for three velocities with respect to Yawing Moment vs. α & β (a), yawing moment comparison to literature [39] (b).

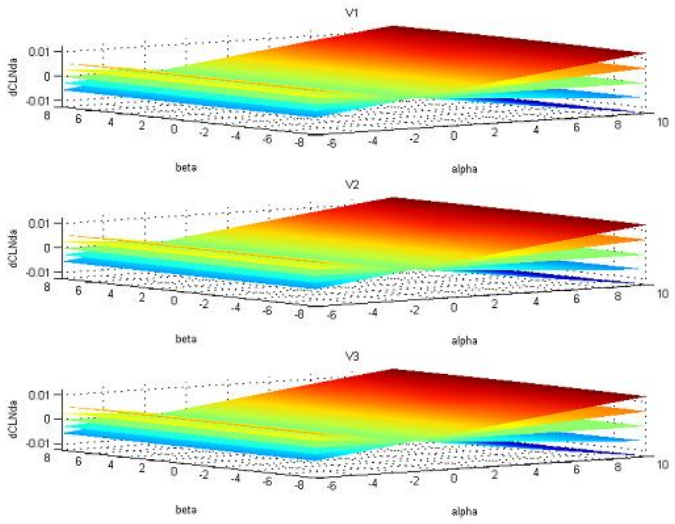


Figure 4.3.3: Yawing Moment as a function of: δ_{ail}.

wing twist. Comparing the initial VLM results with those of the Boeing X-48, we find that the BWB numerical results are nearly identical to a BWB of similar geometry.

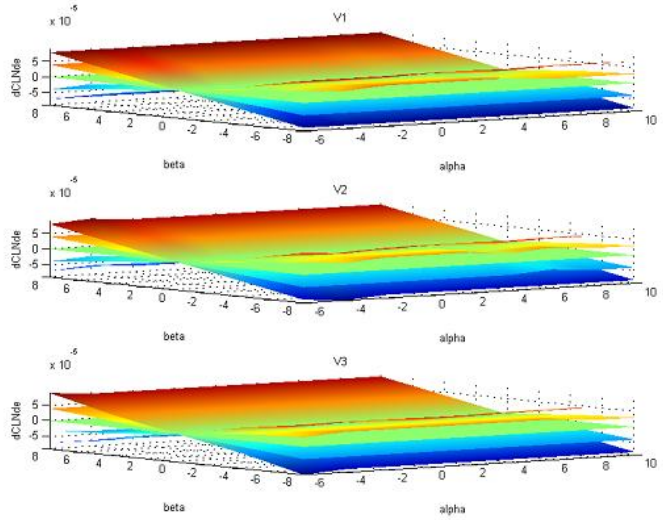


Figure 4.3.4: Yawing Moment as a function of: δ_{ele}.

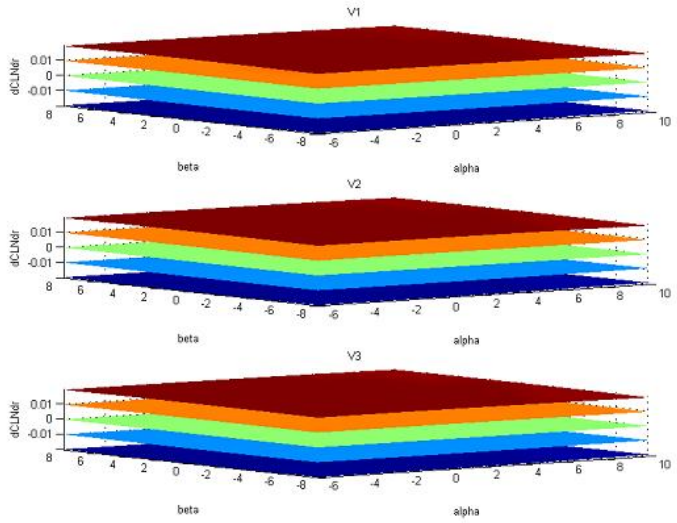
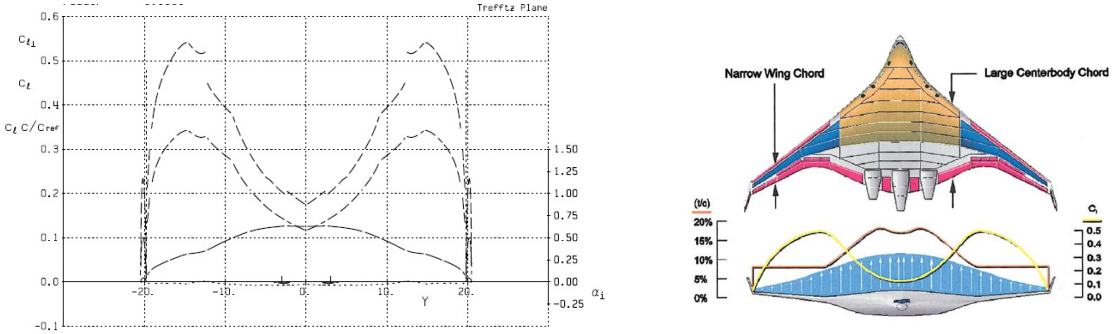


Figure 4.3.5: Yawing Moment as a function of: δ_{rud}.



(a) (b)

Figure 4.3.6: AVL lift distribution results (a), comparison to literature [31](b).

Chapter 5

CFD Analysis

5.1 History of Computational Fluid Dynamics

From 1687, with the publication of Isaac Newton's Principia, to the mid-1960's, advancements in fluid mechanics were made with pioneering experiments and basic theoretical analysis. In nearly all cases these advancements were done through the simplification of the fluid dynamic models of the flow, to closed-form solutions of the governing equations. Up to this time the advancement of fluid flow equations and theory were significant, while the application of these equations in hand-calculations were limited to "academic" or simplified models of fluid flows. However, during the mid-1960's, the advent of the modern computer provided the ability to simultaneously calculate hundreds and even millions of equations at once, handling the calculations of the governing equations in their "exact" form [4], [5].

The physical aspects of any fluid flow are governed by three principles:

- Mass is conserved.
- $F = ma$ (Newton's second law) $\rightarrow$ Momentum Equation.
- Energy is conserved.

These simple aspects lead to the development of the Navier-Stokes equations (Eq 5.1.1 thru 5.1.3), wherein *mass*, *momentum*, and *energy* are solved. From these equations, a limitless number of fluid dynamic applications applications can be defined and solved.

$$\frac{D\rho}{Dt} + \rho\nabla \cdot \vec{u} \;=\; 0 \tag{5.1.1}$$

$$\rho\frac{D\vec{u}}{Dt} \;=\; -\nabla p - \nabla \cdot \tau + \rho\vec{g} \tag{5.1.2}$$

$$E \;=\; E_{kinetic} + E_{potential} + E_{mechanical\ work} + E_{heat\ transfer} + ... \tag{5.1.3}$$

From these equations, simplifications can be applied and, for example, the boundary layer for a 2D case can be investigated. By applying mass conservation and x-momentum conservation in the laminar boundary layer, the N-S equations reduce to a simple expression of the boundary layer thickness δ for a given velocity U, air parameter ν and length scale l.

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0 \tag{5.1.4}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + nu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \tag{5.1.5}$$

$$\delta \sim \sqrt{\frac{\nu l}{U_o}} \quad \Rightarrow \quad \frac{\delta}{l} \sim \sqrt{\frac{\nu}{U_o l}} \sim \frac{1}{\sqrt{R_e l}} \tag{5.1.6}$$

Assuming that the static pressure is constant over the boundary layer cross-section, we find that:

$$p \sim \rho U_o^2 \quad \Rightarrow \quad \frac{\partial p}{\partial y} \sim \frac{\delta^2}{l^2}\rho U_o^2 \sim 0 \tag{5.1.7}$$

This example is used because it is important to understand that though CFD is an excellent tool for 3D analysis of complex fluid dynamic flows, the equations applied at the fluid-wall interface are typically based on 2D theoretical and experimental results. Meaning, that it is important for the CFD user to understand the limitations of the code which is being used, particularly in the selection the physics solvers and turbulence models, as the equations are likely to based on 2D results which are applied to a 3D domain. Furthermore, though very accurate solutions can be obtained for various fluid flows, the price which is paid for increased accuracy, is longer computational time. It is up to the user to define what numerical models are most appropriate for their application: subsonic, transonic, supersonic, rotational flow, etc.. Figure 5.1.1, taken from Green [18], is provided as a reference for the various numerical models.

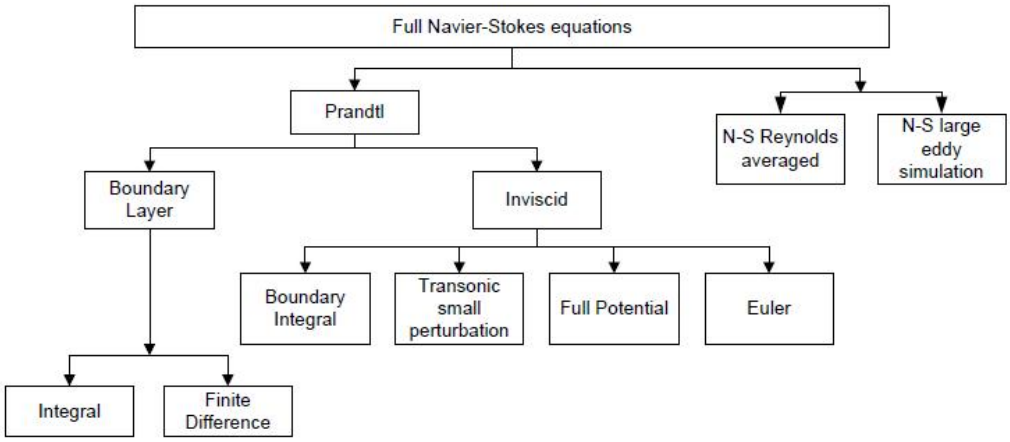


Figure 5.1.1: Hierarchy of CFD Turbulence Models.

5.2 Analysis of the BWB

The higher-order code in use is STAR-CCM+ v4.04, a product of CD-Adapco. Initially a half-model was tested for mesh dependencies and aerodynamic data comparison to the lower-order methods (see Figure 5.2.2), after which a full model was created. It was found that good correlation exists in four axes (F_x, F_z, M_x, M_y), between the half-model CFD results and the full-model lower-order code predictions; with an average offset of less than 5%. A brief description of the conditions is provided below:

- Mesh: structured, surface prism cells, with a fast adaptation to the domain
  - Cell size: 0.001 m (min), 1.0 m (max)
  - Cell count: 9.2 million
- Solvers
  - Reynolds Averaged Navier-Stokes
  - 2nd Order segregated flow, convection/ diffusion
  - Turbulent
    - SST (Menter) K-omega turbulence model
    - Compressibility correction (for Mach 0.735 test cases)
    - Reynolds number correction (for Mach 0.183 test cases)
    - All y+ Wall Treatment
- Flow Conditions
  - Standard atmosphere air at 10 km
  - 0.50 Mach
  - Angle of attack = 0, Angle of side slip = 0

The initial test cases were based on a less than optimum cruise condition, Mach 0.50, because the lower-order results were obtained from in incompressible solver. Once a correlation was obtained between the lower-order method and the CFD, a

full test matrix was examined. First looking at the higher Mach numbers, at the optimized cruise condition, Mach 0.735. Followed by more computationally expensive, low Mach numbers, on the order of 0.183 Mach, representing the takeoff and landing flight regimes. In this flight regime, the BWB is prone to lateral-directional instabilities due to high angles-of-attack combined with asymmetric gust loads or cross-wind induced wing stall. Cases were investigated in the $\alpha = 10^o - 16^o$, and $\alpha = 30^o - 40^o$ deep-stall range, see Figure 5.2.1. The engine pods were removed on the full model to minimize number of elements in the mesh domain, and the effect on the stability is assumed to be minimal. This is considered an adequate assumption based on literature.

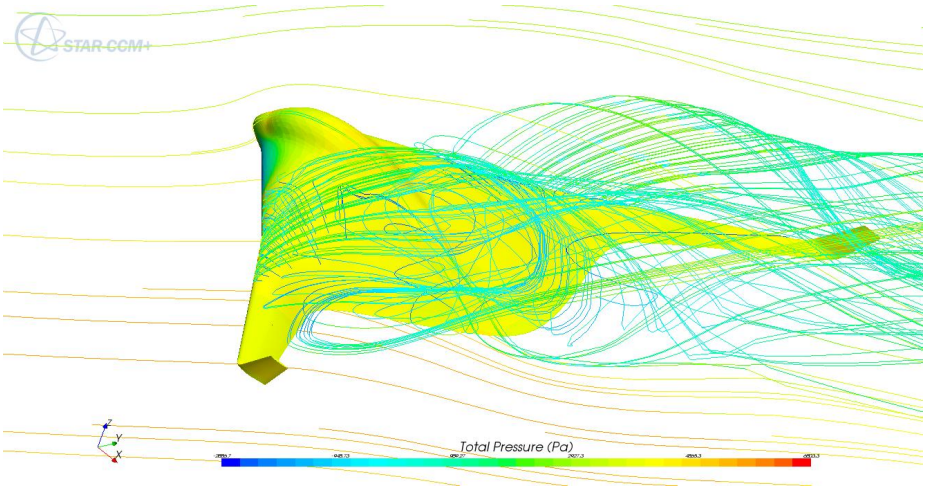


Figure 5.2.1: Deep-stall stability: Back view.

The following figures (Figures 5.1 thru 5.2.4) present the test cases, airplane geometry and mesh geometry, as well as the comparison to lower-order method results. Figures 5.2.5 and 5.2.6 compare the baseline BWB model to the Lear 23 lateral-directional stability, and Figures 5.2.7 and 5.2.8 compare the BWB with the Lear 23, where the BWB employs a novel control surface stability augmentation system called belly-flaps. These belly-flaps, a plane-flap with a variable chord length of 2-5% of the root

chord and located on the BWB centerbody at 60% of the root chord with a span of 25% of the wing, are the result of months in research and analysis by this author to produce a laterally and directionally stable BWB configuration. The results below show that when two belly-flaps, one on the right hand side of the centerline of the vehicle and one on the left, are differentially deployed at 90 degrees — meaning one at 2% and one at 5% — produce enough roll and yaw authority to maintain stability in landing and takeoff configurations, and compare well with the lateral-directional stability of the Lear 23.

Flight Condition	Configuration	Velocity (mach)	aoa	aoss
Cruise	*clean*	0.735	[0,2,6]	[0,2]
Take-off	*clean*	0.183	[0,2,4,8]	[0,2,6]
Stall (guess)	*clean*	0.098	[0,2,6]	[0,2,6]
Take-off	*flapped (belly flap)*	0.183	[0,2,4,8]	[0,2,6]

Table 5.1: CFD test cases.

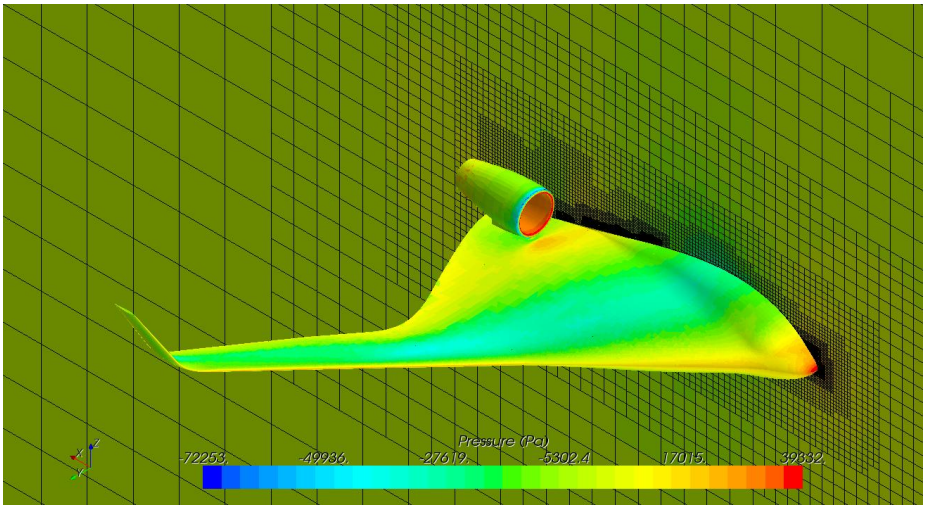


Figure 5.2.2: BWB CFD model: mesh and surface pressure.

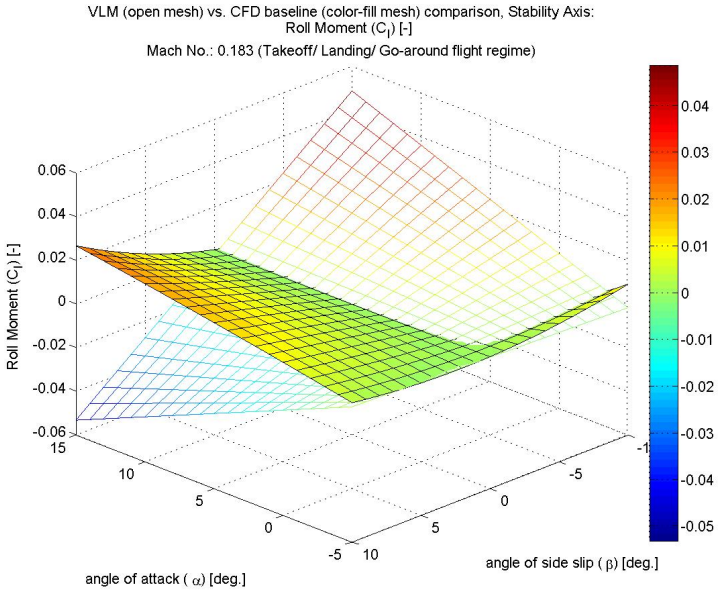


Figure 5.2.3: Roll Moment Coefficient Comparison: VLM vs. CFD baseline configuration.

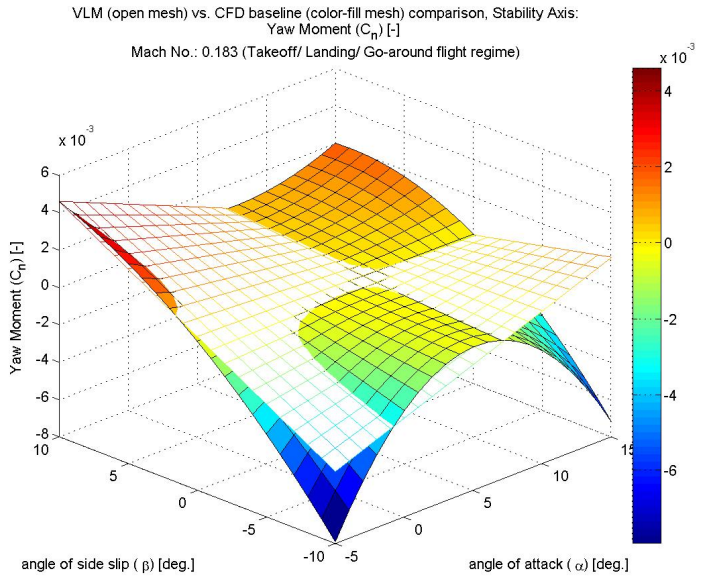


Figure 5.2.4: Yaw Moment Coefficient Comparison: VLM vs. CFD baseline configuration.

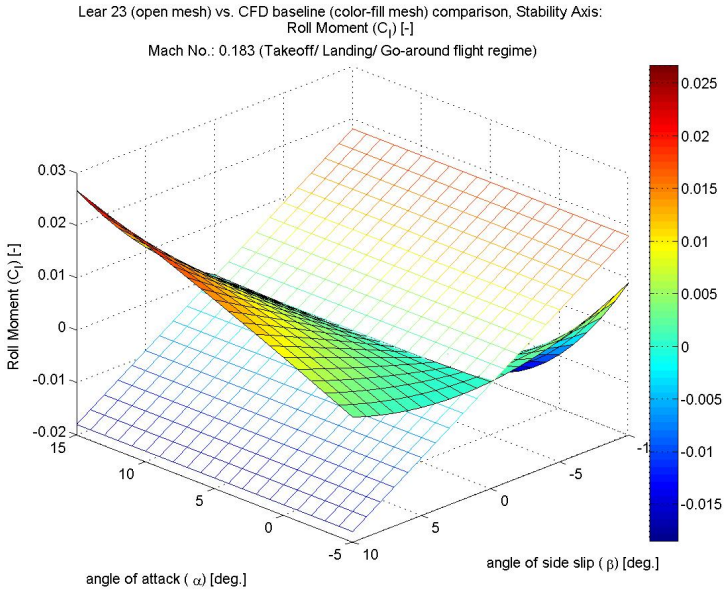


Figure 5.2.5: Roll Moment Coefficient Comparison: Lear 23 vs. CFD baseline configuration.

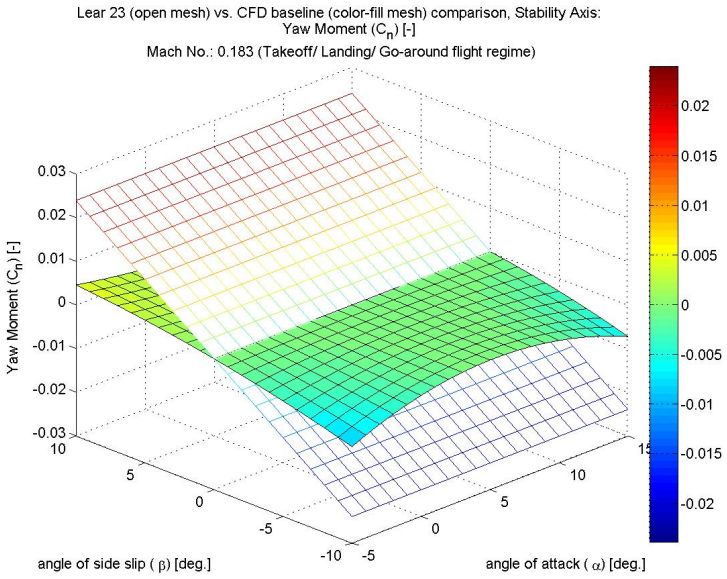


Figure 5.2.6: Yaw Moment Coefficient Comparison: Lear 23 vs. CFD baseline configuration.

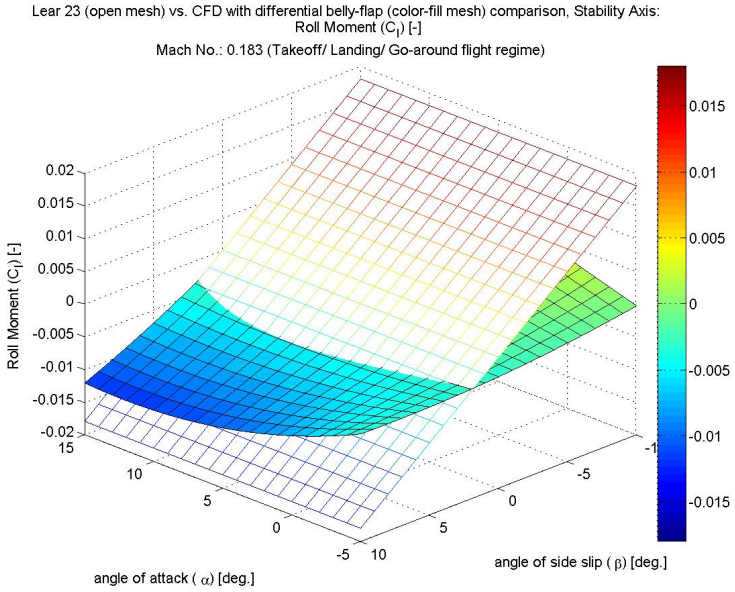


Figure 5.2.7: Roll Moment Coefficient Comparison: Lear 23 vs. CFD belly-flap configuration.

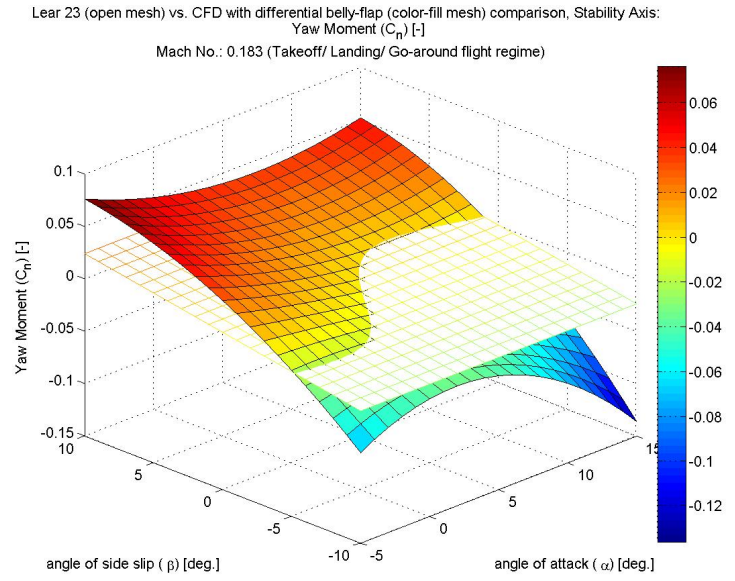


Figure 5.2.8: Yaw Moment Coefficient Comparison: Lear 23 vs. CFD belly-flap configuration.

For the purposes of this work, the author has only presented a few examples where CFD is applied to the airplane design process. In general, the CFD theory and methodologies will be left for the reader to investigate in other available literature, as it is not the goal of this work to provide exhaustive descriptions and derivations on this topic. An excellent introduction to CFD is provided in Reference [53].

Chapter 6

Propulsion Modeling and Estimates

To begin the construction of a numerical propulsion model, the equations from Roskam [45] and Raymer [42] were used. Several polynomial equations were programed into Excel to provide a basis from which to draw initial conclusions about the engine selection. Though the physical characteristics of a jet engine versus a propeller are quite different, the numerical modeling is the same. In either case, a database is needed for the numerical modeling of the propulsion system, such that it can be used in the 6DOF non-linear simulator.

A propulsion database, commonly referred to as an "engine deck", is a set of values corresponding to all possible operating conditions of the propulsion system. The engine deck provides information of thrust and fuel flow for a given altitude, RPM and airspeed. Often, as a starting point, extrapolated data can be obtained with equations from [42] and [45], a complete engine deck can be assembled in an Excel

spreadsheet (Appendix B).

This chapter covers two propulsion systems, the propeller in section one, and the jet engine in section two. The only real difference between the two modeled systems is the data contained in each. The engine-deck, as it relates to the simulator, is identical.

6.1 Propeller Model

6.1.1 Propellers in Literature

Both Raymer (Ch.10 of [42]) and Roskam (Ch.7 of [44], and Vol 2 of [45]), provide excellent resources and equations from which to begin a numerical model of a piston-propeller or turbo-prop system. However, both authors do not cover the characteristics of a propeller fixed to an electric motor. Though the propeller characteristics will remain the same, there is a difference between the operating conditions of an air breathing engine versus an electric motor. Drela [16] covers the propeller-electric motor system quite nicely in the documentation provided with his QPROP code. Additionally, a literature search revealed that there was tabulated data [54] available for a large variety of common-off-the-shelf (COTS) electric motors with combinations of various propellers. Using the data from this source, new values could be extrapolated to the desired operating range or motor selection, and a propulsion database was assembled, see Appendix B.

6.1.2 Validation with QPROP

To validate the extrapolated data, a numerical approach using QPROP was applied; similar to the XFOIL-MatLab (Ch.3) and AVL-MatLab (Ch.4) methods described previously. QPROP, developed by Mark Drela [16], is an executable program which generates electric motor-propeller data, when provided a set of input quantities regarding the electrical system, motor, propeller, and flight conditions. QPROP uses blade-element theory to generate the aerodynamic results of a rotating blade, and combines these results, or calculations, with the electrical calculations for electric motor efficiency, voltage and current. The result is similar to the above results, whereby QPROP is run for a batch or set of conditions and the results are compiled into a test file and Excel spreadsheet. In this work, the QPROP results were used to tune the engine deck, with the assumption that QPROP provides data of higher fidelity than the derived data in Excel (Appendix B).

Blade Element Theory

The blade element theory applies the following assumptions:

1. The actuator is an infinitely thin disc with area S
2. No resistance from the actuator is transfered to air passing through it.
3. Energy is transferred as pressure energy distributed evenly over the actuator area.
4. Velocity of the air through the disc is constant over the entire actuator area.
5. All energy supplied to the actuator is transferred to the air.

Figure 6.1.1 represents a disc actuator in rest in a fluid. Ahead of the disc the fluid moves at speed V_0 and has a pressure of p_0. As the fluid accelerates toward the

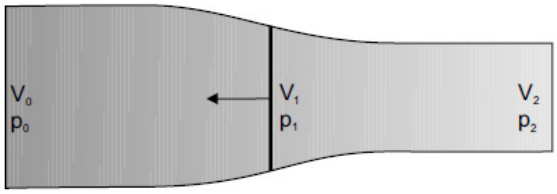


Figure 6.1.1: Blade Element Theory.

disc, the velocity increases, V_1, and the pressure decreases ,p_1. Behind the disc fluid velocity is V_2 and the pressure is p_2.

Applying the equation:

$$P_{engine} = P_{propeller} \tag{6.1.1}$$

and arguing that the pressure differences between p_0, p_1 and p_2 are small; the mass of fluid passing through the disc in unit time is $p_0 S V_1$. Since the increase of rearward momentum of this mass of fluid is the mass of fluid passing through the disc multiplied by the difference in fluid velocity between V_0 and V_2, the thrust on the disc is shown to be:

$$T = p_0 S V_1 (V_2 - V_0) \tag{6.1.2}$$

Characterizing the propeller in terms of thrust and power coefficients, C_T and C_P, respectively, we can apply the blade element theory to real propellers in terms of advance ratio λ and blade Reynolds number Re, based on propeller geometry, where:

$$C_T \;=\; C_T(\lambda, Re, geometry) \tag{6.1.3}$$

$$C_P = C_P(\lambda, Re, geometry) \tag{6.1.4}$$

$$\lambda = \frac{V}{/}\Omega R \tag{6.1.5}$$

$$Re = \frac{\rho \Omega R c_{ave}}{\mu} \tag{6.1.6}$$

where, R is the propeller radius, V is the flight velocity. By applying equation 6.1.5, the equations for thrust and torque can be manipulated into functions of propeller rotation rate Ω and velocity, V:

$$T(\Omega, V) = \frac{1}{2}\rho(\Omega R)^2 \pi R^2 C_T = \frac{1}{2}\rho V^2 \pi R^2 \frac{C_T(\lambda, Re)}{\lambda^2} \tag{6.1.7}$$

$$Q(\Omega, V) = \frac{1}{2}\rho(\Omega R)^2 \pi R^3 C_P = \frac{1}{2}\rho V^2 \pi R^3 \frac{C_P(\lambda, Re)}{\lambda^2} \tag{6.1.8}$$

Applying the blade theory to the DC motor-propeller matching, we apply the electric motor characteristics in terms of voltage v, current i, resistance r, shaft rotation rate Ω and the motor speed constant K_V [RPM/volt], and the resultant terms of propeller thrust T and propeller torque Q. By applying the conservation of energy together with the electric motor equations, we find the motor torque Q_m and motor and propeller rotation rate Ω in terms of i and v:

$$Q_m(i) = (i - i_o)/K_V \tag{6.1.9}$$

$$\Omega(i, v) = (v - ir)K_V \tag{6.1.10}$$

and by combining 6.1.9 and 6.1.10, we find the shaft power P_{shaft} and the motor efficiency η_m:

$$P_{shaft}(i,v) \;=\; Q_m\Omega \;=\; (i-i_o)(v-ir) \tag{6.1.11}$$

$$\eta_m(i,v) \;=\; P_{shaft}/Pelectric \;=\; (1-i_o/i)(1-ir/v) \tag{6.1.12}$$

By manipulating the motor values Q_m, P_{shaft} and η_m in terms of current, i

$$i(\Omega,v) \;=\; (v-\frac{\Omega}{K_V})\frac{1}{r} \tag{6.1.13}$$

we can find the relationships as functions of shaft rotation rate and voltage applied:

$$Q_m(\Omega,v) \;=\; [(v-\frac{\Omega}{K_V})\frac{1}{r}-i_o]\frac{1}{K_V} \tag{6.1.14}$$

$$P_{shaft}(\Omega,v) \;=\; [(v-\frac{\Omega}{K_V})\frac{1}{r}-i_o]\frac{\Omega}{K_V} \tag{6.1.15}$$

$$i(\Omega,v) \;=\; [1-\frac{i_o r}{v-\Omega/K_V}]\frac{\Omega}{vK_V} \tag{6.1.16}$$

Finally, the optimum operating condition for the motor-propeller combination can be found when the torque for the motor Q_m and propeller Q are in equilibrium:

$$Q_m(\Omega,v) \;=\; Q(\Omega,V) \tag{6.1.17}$$

By applying the torque matching condition of equation 6.1.17 we can, for example, deduce the motor power required as a function of flight velocity, in terms of

voltage and current. For more information on the torque matching methods, refer to Reference [16].

6.1.3 Validation with wind tunnel testing

A wind tunnel test was performed to determine the near flight condition capabilities of the system, validating both the extrapolated and calculated data from the previous sections. For this test, the complete electric power propulsion system was used to determine what losses are within the entire system, and a set of eight propellers were tested with one electric motor. To collect data, the propulsion system was run at various conditions of %RPM command — in terms of propeller rotation — as well as various airspeeds and side-slip conditions. Data was obtained via a data acquisition system [37] connected to a laptop computer, and a complete database was assembled. These results were compiled and used to tune the engine deck. Once the engine deck was completed, it was compiled into a MATLAB file, *enginedeck.mat*, for future use in a 6DOF simulator.

6.1.4 Propeller modeling results

The results of the previous sections are presented in Figure 6.1.2, for one propeller-motor combination at 100% RPM. By applying the equations from Raymer for a propeller (based on a piston engine), the thrust versus velocity was calculated and closely aligned with the graphs in Ch.10 of [42]. However, what is immediately apparent is the lapse rate of the piston engine. The data extrapolated from the test by Wergeland, were adjusted for dynamic conditions using Raymer's equations and closely resemble the same second-order curve-fit, with some deviation due to test

conditions and extrapolation for the chosen motor and propeller: the Wergeland [54] test was performed at static conditions. The results from the QPROP calculation are nearly linear and very closely match the results collected in the wind tunnel.

This relatively simple comparison also applies to the other test cases for various %RPM, motor size and propeller combinations. For this authors work; five propellers, two motors, and five %RPM combinations were tested and analyzed in total. It can be concluded that QPROP is an adequate numerical tool for predicting operating conditions of DC motor-propeller matching. Similarly, the test results from Wergeland [54], can also be used as a reliable resource, provided a correction is applied to correct for dynamic conditions. The calculated results, based on piston engine-propeller equations from Raymer [42], show the non-linearity that we expect from air-breathing engine-propeller systems. And though Raymer's equations provide a good estimate for the present analysis using a DC motor, the results should be used with caution.

6.2 Turbojet Model

6.2.1 Turbojets in Literature

Both Raymer (Ch.10 of [42]) and Roskam (Ch.6 of [44], and Vol 2 of [45]) provide excellent resources and equations from which to begin a numerical model of a jet engine. However, for a more comprehensive understanding of jet engines, the author referred to Mattingly's work [34] on the individual elements of the jet engine and

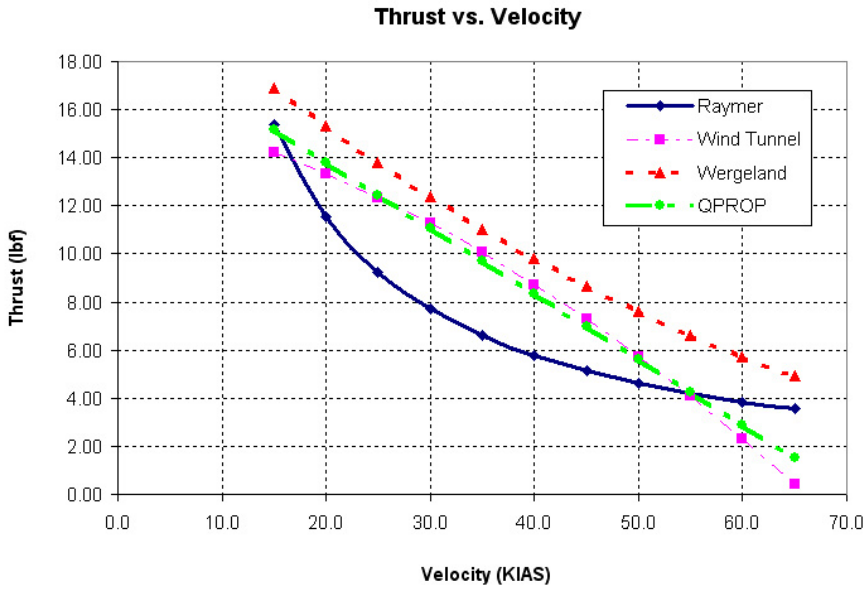


Figure 6.1.2: Propeller analysis results for constant RPM (100% RPM).

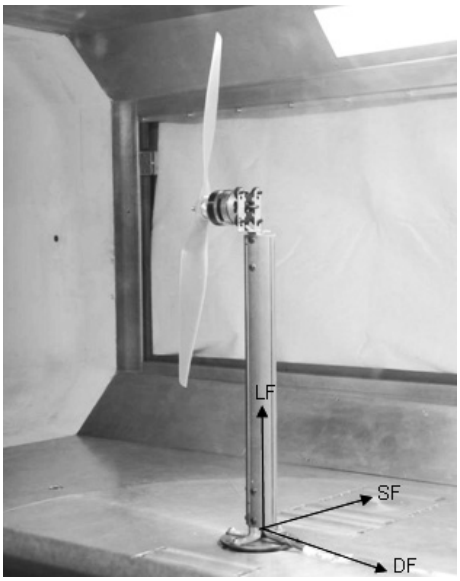


Figure 6.1.3: Propeller wind tunnel test set-up.

included examples which help increase the accuracy of engine modeling. Mattingly also provides a nice command line-based code which allows the development of a numerical jet engine model.

In most cases of jet engine modeling (based on the author's industry experience) a manufacturer will provide an engine-deck to a customer, thereby alleviating the numerical work to be performed by the engineer. The engine deck can then be translated into a format as presented in Section 6.1, for the integration into a 6DOF non-linear simulator. Appendix B provides a comprehensive view of the engine-deck.

Chapter 7

Mass Moments of Inertia Modeling & Estimates

Both Raymer (Ch.15 of [42]) and Roskam (Ch.6 of [44], and Vol 5 of [45]) provide excellent methodologies and equations for calculating the mass properties of an airplane. Additional references are found in [49], [35], [36] and [41], and this chapter adapts these authors equations to a spreadsheet, for design work and trade studies.

7.1 Mass Property Calculations using Spreadsheets

To estimate the mass properties and rotational moments of inertia of the vehicle, a section of the *RoysdonAeroAnalysis.xls* excel spreadsheet was assembled with a list of basic shapes — sphere, trapezoid, square rod, cone, etc. — to represent basic airplane shapes for wings, fuselages and payloads (Refer to Appendix A). Within this module was the ability to enter the specific values for each shape, e.g. length, width, height, mass, etc., to obtain a result of the mass properties.

Some of the airplane component examples are:

- Wing: could be estimated by a *Rectangular Parallelepiped*.
- Fuselage: could be estimated by a *Cylinder* with a *Cone* on either end.
- Engine: piston = *Cube*, jet = *Cylinder*.
- Various Electrical Components: could be estimated as a *Cube* or *Sphere*.

These examples were tabulated in the spreadsheet as shown in Figure 7.1.1, below:

Mass Properties Calculations

Inertias for Common Geometries

Circular Cylinder Shell			inertias taken at the mass center				
length (z) =	72.00	in.	Ixx =	3419.92	lbf*in^2	23.7494	lb*ft^2
radius (x,y) =	2.00	in.	Iyy =	3419.92	lbf*in^2	23.7494	lb*ft^2
mass (m) =	7.88	lbf.	Izz =	31.52	lbf*in^2	0.2189	lb*ft^2
			mass center (xf2) =	36.00	in.	3.0000	ft.
			mass center (yf2) =	1.00	in.	0.0833	ft.
			mass center (zf2) =	1.00	in.	0.0833	ft.
Uniform Slender Rod			inertias taken at the mass center				
length (z) =	60.00	in.	Ixx =	150.00	lbf*in^2	1.0417	lb*ft^2
mass (m) =	0.50	lbf.	Iyy =	150.00	lbf*in^2	1.0417	lb*ft^2
			Izz =	-	lbf*in^2 -		lb*ft^2
			mass center (xf2) =	-	in. -		ft.
			mass center (yf2) =	-	in. -		ft.
			mass center (zf2) =	30.00	in.	2.5000	ft.
Sphere			inertias taken at the mass center				
radius (x,y,z) =	2.00	in.	Ixx =	0.80	lbf*in^2	0.0056	lb*ft^2
mass (m) =	0.50	lbf.	Iyy =	0.80	lbf*in^2	0.0056	lb*ft^2
			Izz =	0.80	lbf*in^2	0.0056	lb*ft^2
			mass center (xf2) =	1.00	in.	0.0833	ft.
			mass center (yf2) =	1.00	in.	0.0833	ft.
			mass center (zf2) =	1.00	in.	0.0833	ft.
Rectangular Parallelepiped			inertias taken at the mass center				
length (z) =	5.00	in.	Ixx =	8.33	lbf*in^2	0.0579	lb*ft^2
width (y) =	5.00	in.	Iyy =	4.83	lbf*in^2	0.0336	lb*ft^2
height (x) =	2.00	in.	Izz =	4.83	lbf*in^2	0.0336	lb*ft^2
mass (m) =	2.00	lbf.	mass center (xf2) =	1.00	in.	0.0833	ft.
			mass center (yf2) =	2.50	in.	0.2083	ft.
			mass center (zf2) =	2.50	in.	0.2083	ft.
Conical Shell			inertias taken at the mass center of the cone (not the cone base)				
length (z) =	60.00	in.	Ixx =	100.50	lbf*in^2	0.6979	lb*ft^2
radius of the base (x,y) =	2.00	in.	Iyy =	100.50	lbf*in^2	0.6979	lb*ft^2
mass (m) =	0.50	lbf.	Izz =	1.00	lbf*in^2	0.0069	lb*ft^2
			mass center (xf2) =	1.00	in.	0.0833	ft.
			mass center (yf2) =	1.00	in.	0.0833	ft.
			mass center (zf2) =	40.00	in.	3.3333	ft.
Elliptic Paraboloid (rounded cone)			inertias taken at the mass center of the cone (not the cone base)				
length (z) =	60.00	in.	Ixx =	100.33	lbf*in^2	0.6968	lb*ft^2
width of the base (y) =	2.00	in.	Iyy =	100.75	lbf*in^2	0.6997	lb*ft^2
height of the base (x) =	3.00	in.	Izz =	1.08	lbf*in^2	0.0075	lb*ft^2
mass (m) =	0.50	lbf.	mass center (xf2) =	1.50	in.	0.1250	ft.
			mass center (yf2) =	1.00	in.	0.0833	ft.
			mass center (zf2) =	40.00	in.	3.3333	ft.

Figure 7.1.1: Mass Properties for Common Geometries.

Using this module (Figure 7.1.1, a calculation could be performed for each major and/or minor airplane component, and tabulated such that each virtual component

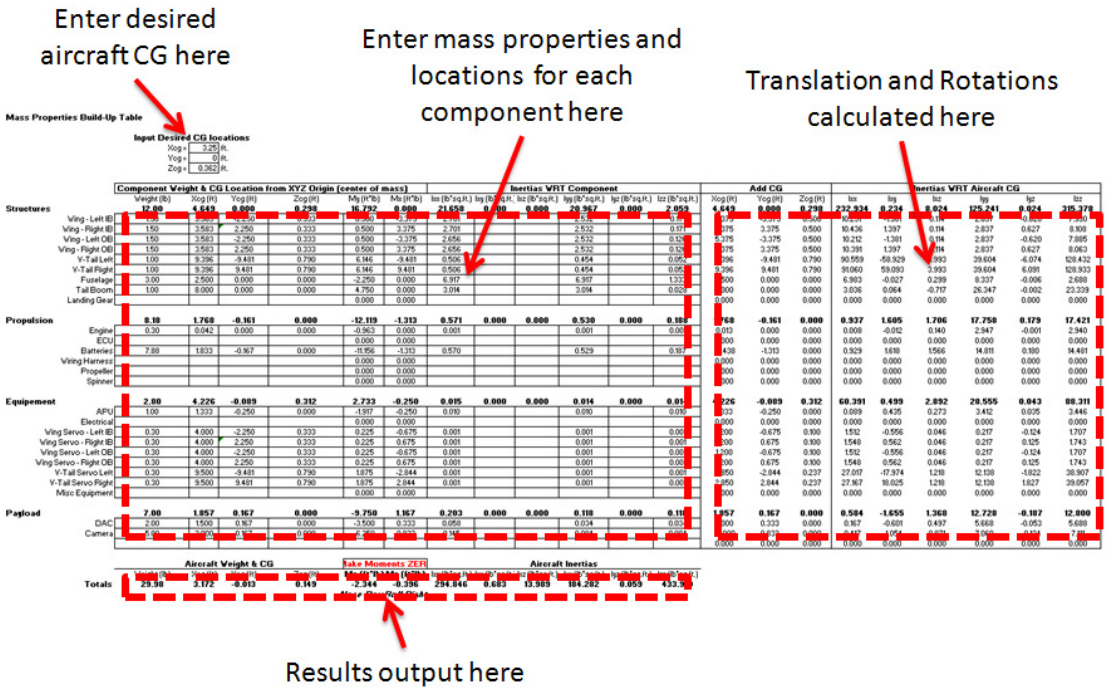


Figure 7.1.2: Mass Properties for Common Geometries.

could be placed into their respective x, y, z locations. The table (Figure 7.1.2) performs the translation and rotation calculations for all the individual masses and moments of inertias, using the following equations:

$$m^A = \sum_{k=1}^{n} m^C \tag{7.1.1}$$

$$\begin{aligned} X^A_{CG} &= \textstyle\sum_{k=1}^{n} X^C_{CG} \\ Y^A_{CG} &= \textstyle\sum_{k=1}^{n} Y^C_{CG} \\ Z^A_{CG} &= \textstyle\sum_{k=1}^{n} Z^C_{CG} \end{aligned} \tag{7.1.2}$$

$$\begin{aligned} M_X^A &= \sum_{k=1}^{n} M_X^C = \sum_{k=1}^{n} [(X_{CG}^C - X_{CG}^{AC}) \cdot m^C] \\ M_Y^A &= \sum_{k=1}^{n} M_Y^C = \sum_{k=1}^{n} [(Y_{CG}^C - Y_{CG}^{AC}) \cdot m^C] \\ M_Z^A &= \sum_{k=1}^{n} M_Z^C = \sum_{k=1}^{n} [(Z_{CG}^C - Z_{CG}^{AC}) \cdot m^C] \end{aligned} \tag{7.1.3}$$

$$\begin{aligned} I_{xx}^G &= I_{xx}^C \cdot m^C((Y_{CG}^C - Y_{CG}^A)^2 + (Z_{CG}^C - Z_{CG}^A)^2) \\ I_{xy}^G &= I_{xy}^C \cdot m^C((X_{CG}^C - X_{CG}^A) \cdot (Y_{CG}^C - Y_{CG}^A)) \\ I_{xz}^G &= I_{xz}^C \cdot m^C((X_{CG}^C - X_{CG}^A) \cdot (Z_{CG}^C - Z_{CG}^A)) \\ I_{yy}^G &= I_{yy}^C \cdot m^C((X_{CG}^C - X_{CG}^A)^2 + (Z_{CG}^C - Z_{CG}^A)^2) \\ I_{yz}^G &= I_{yz}^C \cdot m^C((Y_{CG}^C - Y_{CG}^A) \cdot (Z_{CG}^C - Z_{CG}^A)) \\ I_{zz}^G &= I_{zz}^C \cdot m^C((X_{CG}^C - X_{CG}^A)^2 + (Y_{CG}^C - Y_{CG}^A)^2) \end{aligned} \tag{7.1.4}$$

$$\begin{aligned} I_{xx}^A &= \sum_{k=1}^{n} I_{xx}^G \\ I_{xy}^A &= \sum_{k=1}^{n} I_{xy}^G \\ I_{xz}^A &= \sum_{k=1}^{n} I_{xz}^G \\ I_{yy}^A &= \sum_{k=1}^{n} I_{yy}^G \\ I_{yz}^A &= \sum_{k=1}^{n} I_{yz}^G \\ I_{zz}^A &= \sum_{k=1}^{n} I_{zz}^G \end{aligned} \tag{7.1.5}$$

where m^A is the total mass of the airplane and m^C mass of the component.

$$(X_{CG}^A,\ Y_{CG}^A,\ Z_{CG}^A)$$

are the CG locations in the airplane reference frame, and

$$(X_{CG}^C,\ Y_{CG}^C,\ Z_{CG}^C)$$

are the CG locations in the component reference frame.

$$(M_X^A,\ M_Y^A,\ M_Z^A)$$

are the sum of moments, translated from the component reference frame, to the airplane reference frame.

$$(I^G_{xx},\ I^G_{xy},\ I^G_{xz},\ I^G_{yy},\ I^G_{yz},\ I^G_{zz})$$

are the component inertias translated to the airplane reference frame, and

$$(I^A_{xx},\ I^A_{xy},\ I^A_{xz},\ I^A_{yy},\ I^A_{yz},\ I^A_{zz})$$

are the total inertias in the airplane reference frame.

The result is a table which contains the complete, but preliminary, total mass properties and rotational moments of inertia of the entire airplane as a unit. This provides a starting point for further stability and dynamic analysis in AVL and other such analyses. Additionally the table provides flexibility to move and/or adjust values (mass, inertia, location) to estimate a balanced airplane, prior to building a higher fidelity model. Refer to Appendix B for Figures and more information.

7.2 Mass Property Calculations using CAD

The following stage of the airplane design process should be approached carefully, as in all stages in the airplane design process. This, if not performed earlier in the design process, will serve as a baseline for many other evaluations in the airplane design process, that is the development of the 3D CAD model. As with other design processes, the CAD model should be developed with a well though out build process to allow flexibility in future uses. Starting from the location of sketch planes and 2D sketches, to the lofting of surfaces and later cutting of bodies to create individual parts from

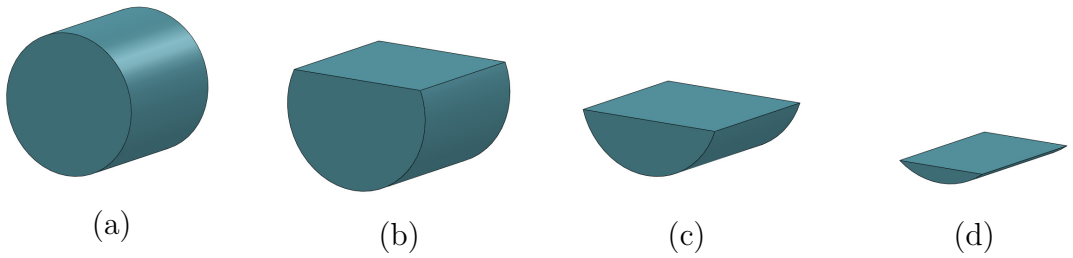
the final airplane exterior geometry, known as the Outer Mold Line (OML). In this case the author prefers the use of SolidWorks specifically for its ability to create and easily manage configurations, or derived parts, from a master part or OML. Meaning: when the OML is complete, the wing can be cut from the OML and saved within the same CAD file as a *configuration*, such that if an airfoil or wing dihedral is updated, this update propagates to the sub-components. This is a convenient and powerful tool, as it facilitates the design of *one* airplane within *one* file. This also provides a simple file management system of the various sub-components for manufacturing and rapid prototyping. Additionally, creating sub-components of the complete OML may be used for structural analysis models and CFD flow domains, using solvers like *ANSYS Mechanical* and *Star-CCM+*, respectively. Of course, any structural analysis and computational fluid dynamics package may be used, these are however just two examples of powerful and robust software packages which communicate with *Solid-Works* directly in such a way that when the model is updated, so too is the mesh updated in the analysis package.

The above discussion introduces the need for a clean and well-built OML from which to derive sub-components; now the assignment of mass properties can be addressed.

One of the nice features of using a 3D CAD package is the ability to design an airplane assembly in 3D space with each individual component. Depending on the level of fidelity needed in the model — typically based on the demands of manufacturing — the model can actually be quite simple. From the baseline OML, the sub-components can be cut to shape and size, leaving the component model as a

solid: for example, a manufactured wing commonly has a top and bottom skin with a spar and many ribs, however in the CAD model the wing can remain one solid shape. From this generic shape, the material density properties can be assigned to it and the shape can be cut and updated to make the mass property results more accurate to the true properties — if known. In the case where the mass properties are know for a particular component, the model can be tuned to manage the mass locations, and therefore provide a higher fidelity CAD mass and inertia model.

From this mass model, the mass properties can be derived for various configurations, again using the idea of sub-components, and a mass properties database can be created, *massprops.mat* for example (Refer to Appendix B). A fuel tank is a good example of this configuration build: The fuel tank can be a sub-component of an airplane assembly, such that discrete fuel states can be created, e.g. 10 gal., 20 gal., 30 gal. and so on (See Figures 7.2.2 thru 7.2.4 for examples). In this way, a configuration approach can be taken in which the airplane now resembles a variable mass and the mass properties can be created for the airplane versus fuel state. These results can be used to populate a database or generate polynomials for airplane CG and rotational moments of inertia for various flight conditions, and therefore be used in flight simulation calculations.



(a) (b) (c) (d)

Figure 7.2.1: CAD model of fuel tank configurations: 15 *gal.* (a), 10 *gal.* (b), 5 *gal.* (c), 1 *gal.* (d).

15 gal. Tank								
Fuel Qty (gal)	Weight (lbs)	Xcg (in)	Ycg (in)	Zcg (in)	Ixx (lbs*in^2)	Iyy (lbs*in^2)	Izz (lbs*in^2)	Ixz (lbs*in^2)
1	18	20.0	0.0	8.0	0.189	0.220	0.282	n/a
5	45	20.0	0.0	8.6	0.453	0.543	0.657	n/a
10	78	20.0	0.0	9.1	0.882	0.945	1.012	n/a
15	112	20.0	0.0	10.5	1.230	1.344	1.486	n/a

Figure 7.2.2: Variable Mass Properties table for 15 gallon fuel tank.

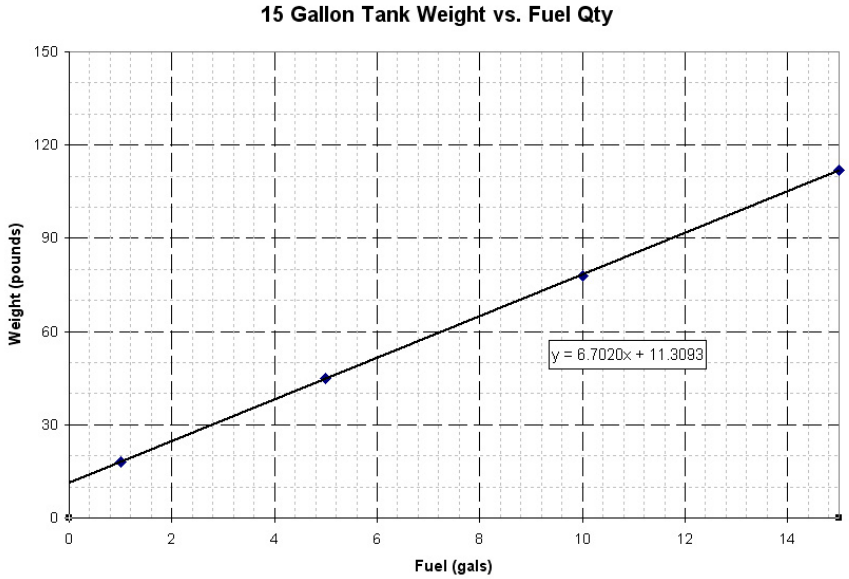


Figure 7.2.3: Variable Mass Properties graph of weight versus fuel quantity for 15 gallon fuel tank.

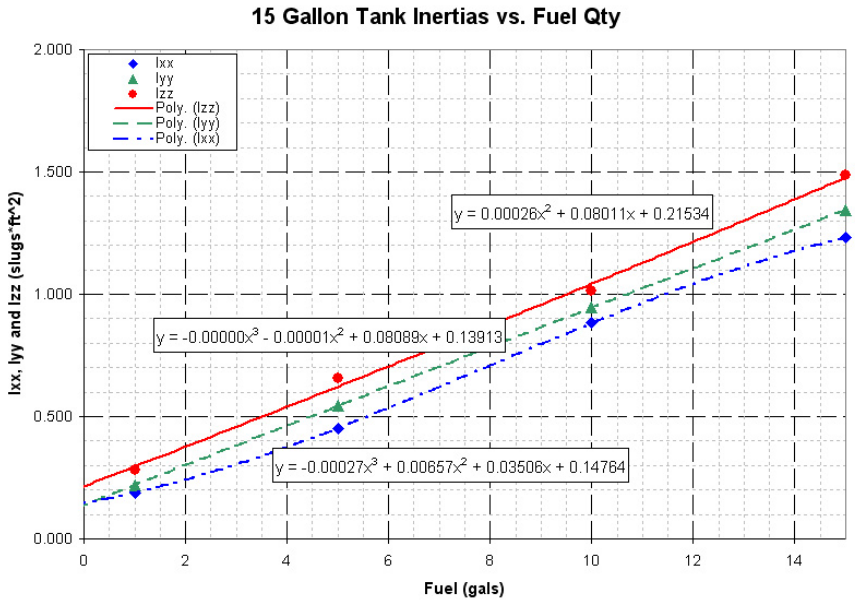


Figure 7.2.4: Variable Mass Properties graph of inertias versus fuel quantity for 15 gallon fuel tank.

Chapter 8

Simulation

The following sub–sections address the various components of a flight simulator built in MatLab & Simulink. The first section discusses the construction and uses for a linear simulation, and the second section discusses the construction and evaluation of a full non-linear simulation.

8.1 Linear & Trim Simulation

The preceding sections and sub–sections have described how to create the necessary building blocks, or numerical models, necessary for a high–fidelity simulator, e.g. the *aero-deck*, *engine-deck*, *mass-props look-up-table*. This section discusses the methods to build a linear simulator using the following models: aero, engine, mass properties, and actuator(s) models. It is of course possible to build a linear simulator much before this stage in the airplane design process, and in fact this was done in the Excel spreadsheet. However now, as will be shown in the following paragraphs, it is necessary to have more flexibility in the simulator, and increased capabilities, for the purpose of generating transfer functions and modern control laws using state space

methods. These results will be used in development of the autopilot logic loops and gains, and eventually in the 6DOF non-linear simulator sub–routines.

Though several programs work well for the modeling of a vehicle numerically, the author found it easiest to model the vehicle in MatLab. MatLab provides several robust sub-routines, which allow the user to focus his or her time on the development of a specific mathematical routines or numerical models. This time saving advantage is crucial during the development process and maximizes the engineer's ability to be creative instead of becoming lost in the mundane tasks specific to programming. Additionally, writing the simulation in a language which is common to data analysis, be it wind tunnel testing or flight testing, inherently provides flexibility to link these databases and tune numerical models, thereby increasing the model accuracy and performance.

8.1.1 Trim Simulation

Determining the trimmed flight condition, is the first step to obtaining a set airplane State Vectors and Matrices. This is performed by creating a trim tool, which uses the inputs for flight conditions and performs an optimization against a cost function to obtain a state vector which describe the airplane state. The inputs include; velocity, altitude, throttle setting, roll, pitch and yaw. From these conditions the trim tool performs an optimization using the *fminsearch* function in MatLab against the cost function (eq. 8.1.3) to obtain the state vector (eq. 8.1.1) which contains the airplane 6DOF motion, including control surface deflections, (eq. 8.1.2). This state vector is then the input to the linear tool.

The following steps are performed in the trim tool:

1. User specifies a flight condition, initial guess for the optimization vector and other trim tool controls.
2. Flight path constraints are calculated.
3. The airplane state is calculated.
4. A scalar cost function is calculated from the airplane state.
5. A minimization function adjusts the optimization vector to generate a cost as close to zero as possible via a simplex algorithm.
6. Post processing, display and storage of the solution.

The equations for the trim function follow, and the state vector equations based on the flight condition is presented in Tables 8.1 & 8.2:

$$\vec{X} = \left\{ \begin{array}{l} V_T \\ \alpha \\ \beta \\ \phi \\ \theta \\ \psi \\ P \\ Q \\ R \\ P_N \\ P_E \\ P_D \\ \omega_{eng} \end{array} \right\} \tag{8.1.1}$$

$$\vec{X} = \left\{ \begin{array}{l} \delta_{throttle} \\ \delta_{elevator} \\ \delta_{aileron} \\ \delta_{rudder} \end{array} \right\} \tag{8.1.2}$$

$$J = \dot{V}_T^2 + 10(\dot{\alpha}^2 + \dot{\beta}^2) + 5(\dot{P}^2 + \dot{Q}^2 + \dot{R}^2) \tag{8.1.3}$$

Table 8.1: Trim Tool States: Level Turn.

State	Straight and Level	Level Turn
ϕ	0	$tan^{-1}(\frac{G\ cos\beta}{cos\alpha - G\ sin\alpha\ sin\beta})$ $G = \frac{\psi V_T}{g}$
θ	α	$tan^{-1}\ (\frac{ab + sin\gamma\sqrt{a^2 - sin^2\gamma + b^2}}{a - sin^2\gamma})$ $a\ =\ sin\alpha\ cos\beta$ $b\ =\ sin\phi\ sin\beta + cos\phi\ sin\alpha\ cos\beta$ $sin\gamma\ =\ 0$
P	0	$\dot{\psi}\ sin\theta$
Q	0	$R\ tan\phi$
R	0	$\frac{\dot{\psi}\ cos\theta}{tan\phi\ sin\phi + cos\phi}$

Table 8.2: Trim Tool States: Rate of Climb.

State	Straight and Level	Rate of Climb
ϕ	0	0
θ	α	$tan^{-1}\ (\frac{ab+sin\gamma\sqrt{a^2-sin^2\gamma+b^2}}{a-sin^2\gamma})$ $a\ =\ sin\alpha\ cos\beta$ $b\ =\ sin\phi\ sin\beta + cos\phi\ sin\alpha\ cos\beta$ $sin\gamma\ =\ \frac{\dot{h}}{V_T}$
P	0	0
Q	0	0
R	0	0

8.1.2 Linear Simulation

The linear tool is a MatLab-based application that determines a set of linear models for an airplane using non-linear models and a trim position from the trim tool. By employing a linear tool on a specific airplane state, one can calculate the feedback for a particular control loop. The result includes a state vector, control vector, initial output and flight condition.

The Linear tool performs the following tasks:

1. User specifies name of the file containing the output from the trim tool, *trimToolOut*
2. Trim data is loaded from *trimToolOut* structure and broken back out to the same structures used in the trim tool.
3. Model data is loaded into the workspace.
4. Set up for the Matlab's *numjac* function is performed.
5. The *numjac* function is run.
6. Post processing, display and storage of the solution.

The linear model uses the following set of equations:

$$\begin{aligned} \dot{\vec{x}} &= A\vec{x} + B\vec{u} \\ \dot{\vec{y}} &= C\vec{y} + D\vec{u} \end{aligned} \tag{8.1.4}$$

By applying the same state vector Eq. 8.1.1, as used in the trim tool, and the results from the trim tool, Eq. 8.1.2, the linear tool can calculate the output vectors. The output vector is a collection of parameters that are usually some type of feedback for a control loop. The linear tool provides four outputs.

$$\vec{Y} = \left\{ \begin{array}{l} a_n \\ a_y \\ \phi \\ h \end{array} \right\} \tag{8.1.5}$$

Heading and altitude are pulled directly from the $\vec{X}$ with only a sign change made for altitude. The accelerations are:

$$\begin{array}{rcl} A_n &=& -\dot{W} + QU - PV + gcos\phi\ cos\theta \\ A_y &=& \dot{V} + RU - PW - gsin\phi\ cos\theta \end{array} \tag{8.1.6}$$

where:

$$\begin{array}{rcl} U &=& V_t\ cos\alpha\ cos\beta \\ V &=& V_t sin\beta \\ W &=& V_t sin\alpha\ cos\beta \\ \dot{V} &=& \dot{V}_t\ sin\beta + \dot{\beta} V_t\ cos\beta \\ \dot{W} &=& \dot{V}_t\ sin\alpha\ cos\beta + \dot{\alpha} V_t\ cos\alpha\ cos\beta - \dot{\beta} V_t\ sin\alpha\ sin\beta \end{array} \tag{8.1.7}$$

Note that these accelerations are measured at the airplane CG, and if needed, can be translated to the location of an autopilot sensor, which is not located at the airplane CG.

By applying a batch-mode method, it is possible to assign a range of flight conditions and run them sequentially, first through the trim tool and then through the linear tool. This result is the dataset for a lookup table in a feedback loop for application in a non-linear simulator and the development of an autopilot code.

8.2 Non-linear Simulation MatLab and Simulink

All of the preceding chapters and sections have, up to this point, lain the foundation for this final section: The creation of the six-degrees-of-freedom full non-linear simulator. Each assumption and calculation, good or bad, will reveal its self in the results that follow, and therefore reinforce the necessity for attention to detail and diligence in both calculations and educated guesses in all of the previous work.

Similar to the well-tuned numerical tools used in a laboratory to predict laboratory results, within the limits of the tool, so too can the flight simulator create and predict flight conditions within the limits of its numerics, as well as conditions slightly outside these limits—with some degradation in fidelity. The non-linear simulator provides a tool for vehicle analysis prior to both manufacturing and flight test, and in some cases even prior to wind tunnel testing. A good non-linear simulator also provides a mechanism for pilot training and post-flight performance evaluation, as well as the recreation and evaluation of a flight mishap or accident.

> Special Note:
>
> First, if the case is investigated for flight prediction, the latter example is really only possible after careful tuning of the simulator with flight data, and inherently has limitations in the vehicle flight dynamics. As most accidents are a result of highly non-linear effects during flight, unless modeled correctly, these effects will not be correctly represented in the simulation. Some examples include: a wing separation in flight, large flow separation or flow asymmetry, etc..

Second, if the case is investigated where there exists flight data from which to run the simulator (Euler angles, rates, velocities), a prediction of events could be made up to and including the flight anomaly. This is really the purpose for using the flight simulator in an accident investigation, provided the simulator is well-tuned.

The following sections introduce and discuss the various components of the 6DOF non-linear simulator.

8.2.1 Mechanical Actuators

The actuator model is a necessary component of the flight simulator as it amplifies, delays and otherwise distorts the commands given it by the autopilot. While in most cases of smaller UAV's, meaning, less than a few hundred pounds, the actuator can be modeled as a simple linear actuator with a specific phase delay and time-ramp from one position to the next. However, it is possible to also determine a transfer function of a specific actuator operating under a specific condition, or alternatively, completely model the non-linearity of the actuator sub-system. The afore mentioned example is typically the case for more complex actuators found on large airplane and UAV's, and often time the actuator manufacturer will provide this model under the term's of an non-disclosure agreement. For the case of the test vehicle of this work, the actuator model was a simple linear model that represented the characteristics of the actuators selected for manufacturing. Data for the actuator was modeled using the manufacturer specification for the actuator, followed by tuning the model with bench tests in a controlled environment using specific position commands to obtain specific results for time rate of change and position held.

Here it is also necessary to mention that the actuator model is not complete unless the connection to the control surface is also modeled. Any mechanical advantage — the ratio between the actuator arm length and the control surface arm length — should be modeled to accurately represent the physical system within the numerical model. Therefore it is quite possible to have a separate actuator model for each control surface. Alternatively, if the actuators used are all identical, it is much simpler to model each control surface with the same actuator and apply a gain to the system to represent any mechanical advantage that may be present. Additionally, the dead-band and backlash should be modeled for each surface. The dead-band is the region around zero actuator deflection wherein the actuator remains at or near zero until a command larger than the dead-band region is given. In a physical sense, a significantly small dead-band will result in an actuator continuously searching for a zero position and therefore using up unnecessary electrical energy; which in the case of electrically powered UAV's, including solar powered UAV's, energy conservation is paramount. Larger dead-bands are common in both analog and digital actuators and are typically on the order of ± 0.1 degree, depending on the actuator and its resolution. Backlash, on the other hand is the additional free play that exists everywhere as result of exterior mechanical linkage and/or interior gear slop within the actuator. This problem exists in all physical systems and can only be minimized by various methods, but never completely eliminated.

- Initial state: zero deflection
- Inputs
  - Deflection command (from the autopilot)
- Outputs

– Deflection result (for 6DOF calculations)

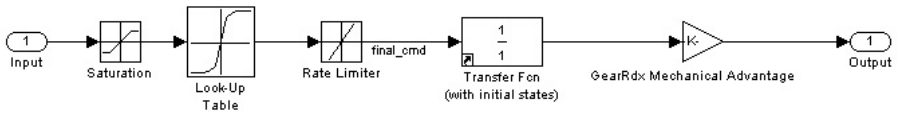


Figure 8.2.1: Simulink actuator model.

8.2.2 Aerodynamics

The aerodynamics block applies the *aerodeck.mat*, which was developed in Ch.4, and applies the variables to lookup tables in the Simulink environment. This allows the simulator to access the discrete data in the lookup table, and interpolates between data points to represent the current flight condition, while also applying the transformation from coefficient form to forces and moments.

As an example, in the following list the axial force is represented by a drag coefficient due to angle of attack (α), angle of side-slip (β), and Mach, as well as delta coefficients due to control surface deflections ($\delta_{aileron}$, $\delta_{elevator}$, δ_{rudder}). The input to this block are the Euler angles, Mach, and control surface angles. The table lookup then provides the resultant force (or moment in the case of moment coefficients). Each of the forces and moments are then combined, and sent as output to the upper level. The following is a breakdown of the aero-block structure.

Aerodynamics Block:

- Forces

  - Axial $\rightarrow$ F_X
    - C_Do (due to α, β, Mach)
    - C_D $\delta_{aileron}$ (due to α, β, Mach)
    - C_D $\delta_{elevator}$ (due to α, β, Mach)
    - C_D δ_{rudder} (due to α, β, Mach)
    - $C_{D\ skin\ friction}$ (due to Mach)
  - Side $\rightarrow$ F_Y
    - (similar to C_D)
  - Normal $\rightarrow$ F_Z
    - (similar to C_D)

- Moments
  - Roll $\rightarrow$ M_X
    - (similar to C_D)
  - Pitch $\rightarrow$ M_Y
    - (similar to C_D)
  - Yaw $\rightarrow$ M_Z
    - (similar to C_D)

The application of this structure can be seen in the following figure:

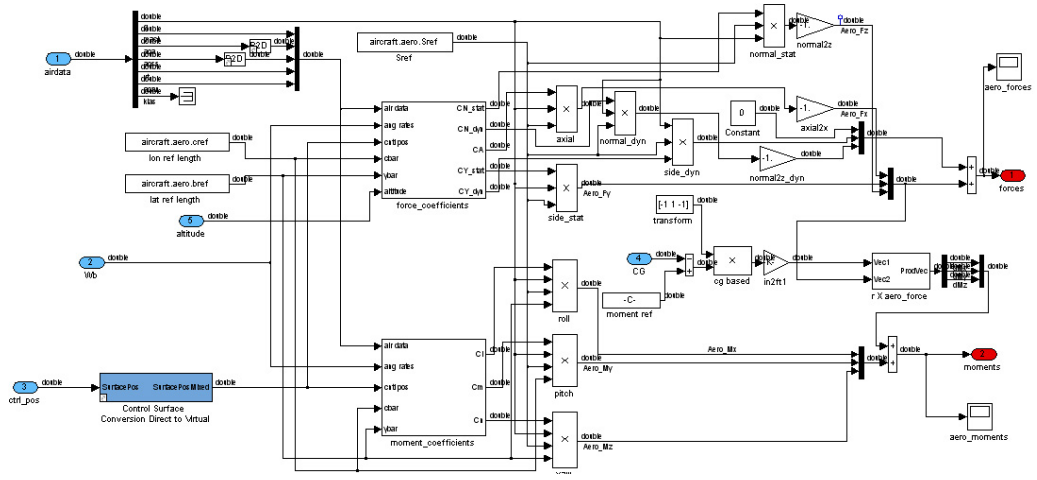


Figure 8.2.2: Simulink aerodynamics model.

8.2.3 Mass Properties & Inertias

The mass properties block applies the *massprops.mat*, which was developed in Ch.7, and applies the variables to lookup tables in the Simulink environment. This allows the simulator to access the discrete data in the lookup table, and interpolates between data points to represent the current flight condition for CG location and moments of inertia.

The following figure displays the case of a variable mass simulation, where the CG and MOI are calculated based on fuel state.

- Initial state
  - Fuel quantity
  - Airplane Weight
  - Airplane CG
- Inputs
  - Fuel Flow (from the engine block)
- Outputs
  - Weight (for 6DOF calculations)
  - CG (for 6DOF calculations)
  - MOI (for 6DOF calculations)
  - Fuel Quantity (for the autopilot)

8.2.4 Propulsion

The propulsion block applies the *enginedeck.mat*, which was developed in Ch.6, and applies the variables to lookup tables in the Simulink environment. This allows the

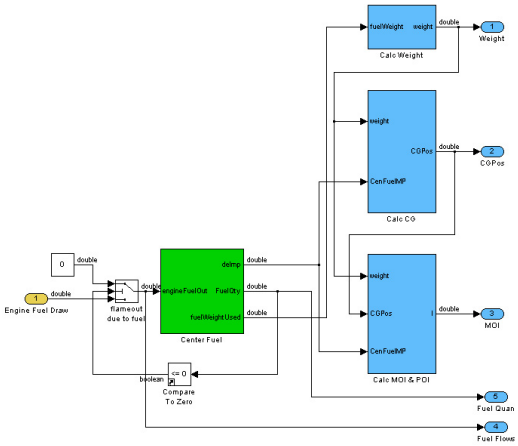


Figure 8.2.3: Simulink mass properties model.

simulator to access the discrete data in the lookup table, and interpolates between data points to represent the current flight condition for fuel flow, EGT, and thrust, as a function of throttle command in %RPM. This block requires the following:

- Initial state: none
- Inputs
  - Throttle Command (from the autopilot)
  - Altitude (from the environment block)
  - Mach (from the environment block)
  - CG (from the mass properties block)
- Outputs
  - %RPM (for autopilot)
  - Engine Momentum (for 6DOF calculations)
  - Forces (for 6DOF calculations)
  - Moments (for 6DOF calculations)
  - Fuel Flow (for mass properties calculations)
  - EGT (for autopilot)

As presented in the following figure:

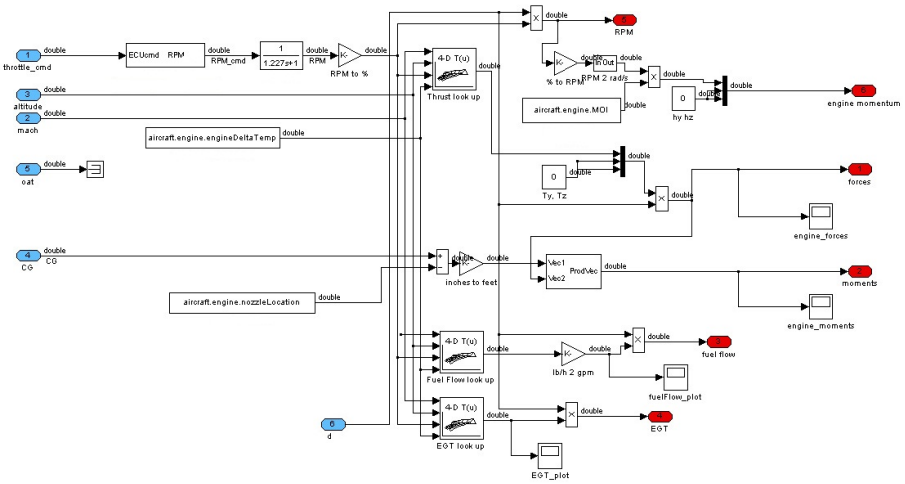


Figure 8.2.4: Simulink propulsion model.

8.2.5 Environment: Earth, Wind & Turbulence

The environment block applies a terrain model, wind and turbulence model, atmospheric model, and gravitational force model. These models provide the necessary information to simulate the airplane dynamics in a "natural" environment, which is capable of replicating real conditions. This is a necessary model when testing the airplane dynamics in the presence of wind gusts, wind shear, and turbulence. Additionally, it provides the air data necessary for calculating the dynamic pressure for aerodynamic coefficients, based on altitude and non-standard day conditions.

- Initial state
  - Wind Velocity & Direction
  - Turbulence intensity
  - Initial Altitude, Latitude & Longitude
- Inputs
  - Altitude (from the 6DOF quaternion block)
  - Velocity (from the 6DOF quaternion block)
  - Euler Angles (from the 6DOF quaternion block)

  - Latitude (from the 6DOF quaternion block)

- Outputs
  - Above Ground Level (for autopilot)
  - Euler angles (for various blocks)
  - Euler rates (for various blocks)
  - Temperature (for various blocks)
  - Pressure (for aero block)
  - Speed of Sound (for various blocks)
  - Air density (for aero block)
  - Gravitational Force (for 6DOF)

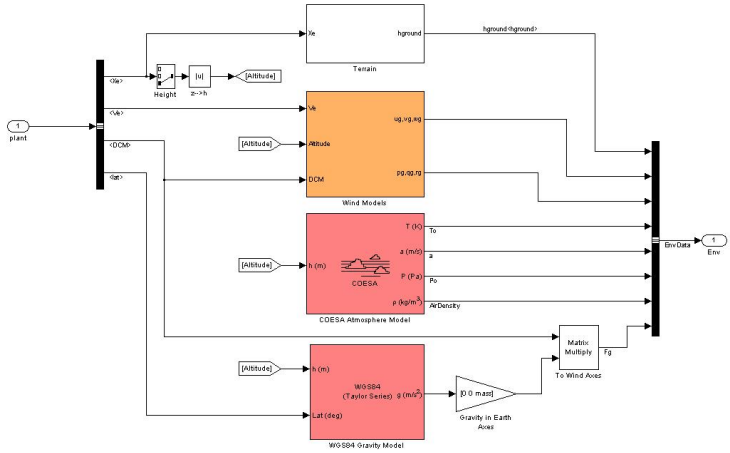


Figure 8.2.5: Simulink environment model.

8.2.6 6DOF Quaternion Calculation

The 6DOF quaternion applies the 6DOF Eulerian equations of a body-fixed coordinate frame (X_b, Y_b, Z_b) about an Earth-fixed reference frame (X_e, Y_e, Z_e), with a correction term which guarantees a non-zero calculation. The model assumes that the rotations occur at the mass center of the body, and that the body is rigid.

The following equations describe the translational motion of a rigid body in the body-fixed frame, where the applied forces $[F_x \ F_y \ F_z]^T$.

$$\underline{F_b} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m(\dot{\underline{V_b}} + \underline{\omega} \times \underline{V_b})$$

$$\underline{V_b} = \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix} \quad , \quad \underline{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

The rotational dynamics of the body-fixed frame are given below, where the applied moments are $[L \ M \ N]^T$, and the inertia tensor I is with respect to the origin O.

$$\underline{M_B} = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I\dot{\underline{\omega}} + \underline{\omega} \times I\underline{\omega}$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

To determine the Euler rates in the body-fixed coordinate frame, the body-fixed angular velocity vector $[p\ q\ r]^T$ and the the rate of change of the Euler angles, $[\dot{\phi}\ \dot{\theta}\ \dot{\psi}]^T$ can be applied.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} +$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = J^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Inverting J then gives the required relationship to determine the Euler rate vector.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = J \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & (sin\phi\ tan\theta) & (cos\phi\ tan\theta) \\ 0 & cos\phi & -sin\phi \\ 0 & \frac{sin\phi}{cos\theta} & \frac{cos\phi}{cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

To translate the Eulerian angles into the Quaternion angles, the integration of the rate of change is given below. The quaternion angle representation is use to avoid divide by zero terms in the angles. Thus, in the case where ϵ tends to a nonzero

value, the K gain drives the norm of the quaternion state vector to unity.

$$\begin{bmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & -q & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} + K\epsilon \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

where:

$$\epsilon = 1 - (q_0^2 + q_2^2 + q_3^2 + q_4^2)$$

The 6DOF quaternion model is represented in Simulink as follows:

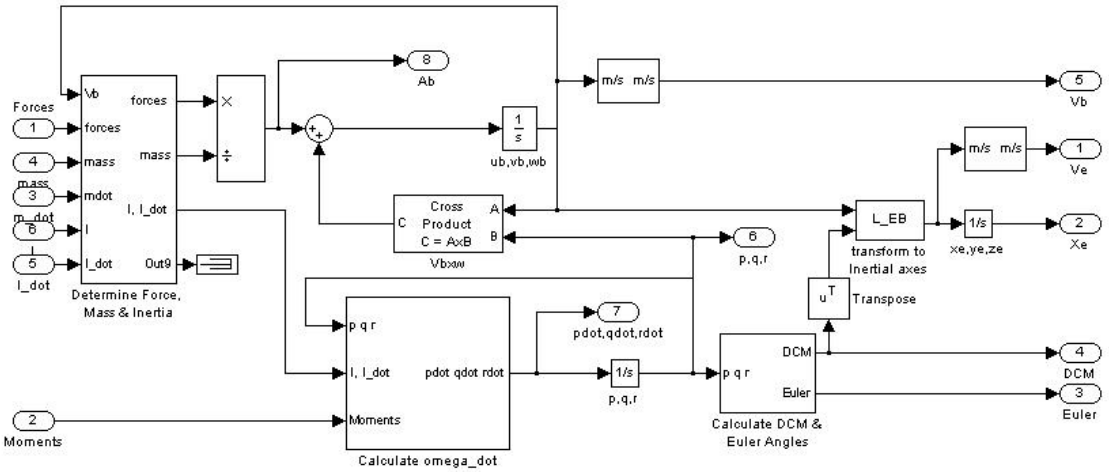


Figure 8.2.6: Simulink 6DOF quaternion model.

8.2.7 Avionics: Guidance, Navigation, & Control

The airplane avionics is possibly the most interesting block of all of the blocks in the Simulink structure. This is where the basic loops of the autopilot can be created,

and manipulated, to create the desired guidance and control characteristics prior to translating the block diagram structure into C-code.

As can be seen in the Figure 8.2.7, this block applies three main blocks: navigation (a three-axis inertial measurement unit (IMU)), guidance, and an autopilot. These three items will be discussed in the following sections.

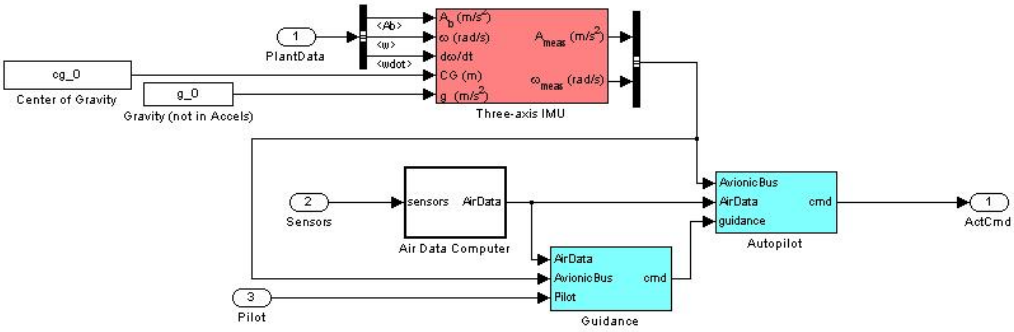


Figure 8.2.7: Simulink avionics model.

Navigation: Inertial Measurement Unit

The IMU contains both a three-axis accelerometer and a three-axis gyroscope.

The *Three-Axis Accelerometer* block implements an accelerometer on each of the three axes. The ideal measured accelerations ($\underline{A_i}_{meas}$) include the acceleration in body axes at the center of gravity ($\underline{A_b}$), lever arm effects due to the accelerometer not being at the center of gravity, and, optionally, gravity in body axes can be removed.

$$\underline{A_i}_{meas} \;=\; \underline{A_b} = \underline{\omega_b} \times (\underline{\omega_b} \times \underline{d}) + \underline{\dot{\omega}_b} \times \underline{d} - \underline{g} \tag{8.2.1}$$

where $\underline{\omega_b}$ are body-fixed angular rates, $\underline{\dot{\omega}_b}$ are body-fixed angular accelerations and

$\underline{d}$ is the lever arm. The lever arm is defined as the distances that the accelerometer group is forward, right and below the center of gravity.

$$\underline{d} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = \begin{bmatrix} -(x_{acc} - x_{CG}) \\ y_{acc} - y_{CG} \\ -(z_{acc} - z_{CG}) \end{bmatrix}$$

The orientation of the axes used to determine the location of the accelerometer group $(x_{acc}, y_{acc}, z_{acc})$ and center of gravity (x_{CG}, y_{CG}, z_{CG}) is from the zero datum (typically the nose) to aft, to the right of the vertical centerline and above the horizontal centerline. The x-axis and z-axis of this measurement axes are opposite the body-fixed axes producing the negative signs in the lever arms for x-axis and z-axis.

Measured accelerations ($\underline{A}_{meas}$) output by this block contain error sources and are defined as:

$$\underline{A}_{meas} = \underline{\underline{A}_i}_{meas} \cdot \underline{A}_{SFCC} + \underline{A}_{bias} + noise \tag{8.2.2}$$

where $\underline{A}_{SFCC}$ is a 3-by-3 matrix of scaling factors on the diagonal and misalignment terms in the non-diagonal, and $\underline{A}_{bias}$ are the biases.

The *Three-Axis Gyroscope* applies a gyroscope to each of the three flight axes, and measures the body rates ($\underline{\omega_b}$), errors and non-linearities of each signal, by the following equation:

$$\underline{\omega}_{meas} \;=\; \underline{\omega}_b \cdot \underline{\omega}_{SFCC} + \underline{\omega}_{bias} + Gs \cdot \underline{\omega}_{g\ sens} + noise \tag{8.2.3}$$

where $\underline{\omega_{meas}}$ are the measured body rates, $\underline{\omega_{SFCC}}$is a 3-by-3 matrix of scaling factors on the diagonal and misalignment terms in the non-diagonal,$\underline{\omega_{bias}}$ are the biases, (Gs) are the gravitational forces on the gyroscope, and $\underline{\omega_{g\ sens}}$ are the gravitational sensitive biases.

Guidance

The guidance block applies the user input from both a joystick command and a discrete signal command from the *Desktop Sim GUI*, which allows the command of very simple to very advanced maneuvers (see Figure 8.2.8). The guidance block also applies a simple range calculation to determine the distance traveled, by applying the following equation:

$$Range \ = \ \sqrt{(x_1 - x_2)^2 \ + \ (y_1 - y_2)^2 \ + \ (z_1 - z_2)^2} \tag{8.2.4}$$

Sensors

The *Sensors* block applies virtual transducers to replicate the sensors on board a real autopilot unit. These consist of the angle and rate sensors, pressure altitude sensor (from a pitot-static probe) and an airspeed correction. The result is a signal combined with some white noise, which is passed to the autopilot block.

Autopilot

The *autopilot* block contains the necessary equations, or loops, for commanding the various control surfaces and throttle %RPM for a desired flight condition, by applying similar equations as presented in Section 8.1.1 for straight and level flight, turning flight or straight and climbing flight. The autopilot determines the output based on

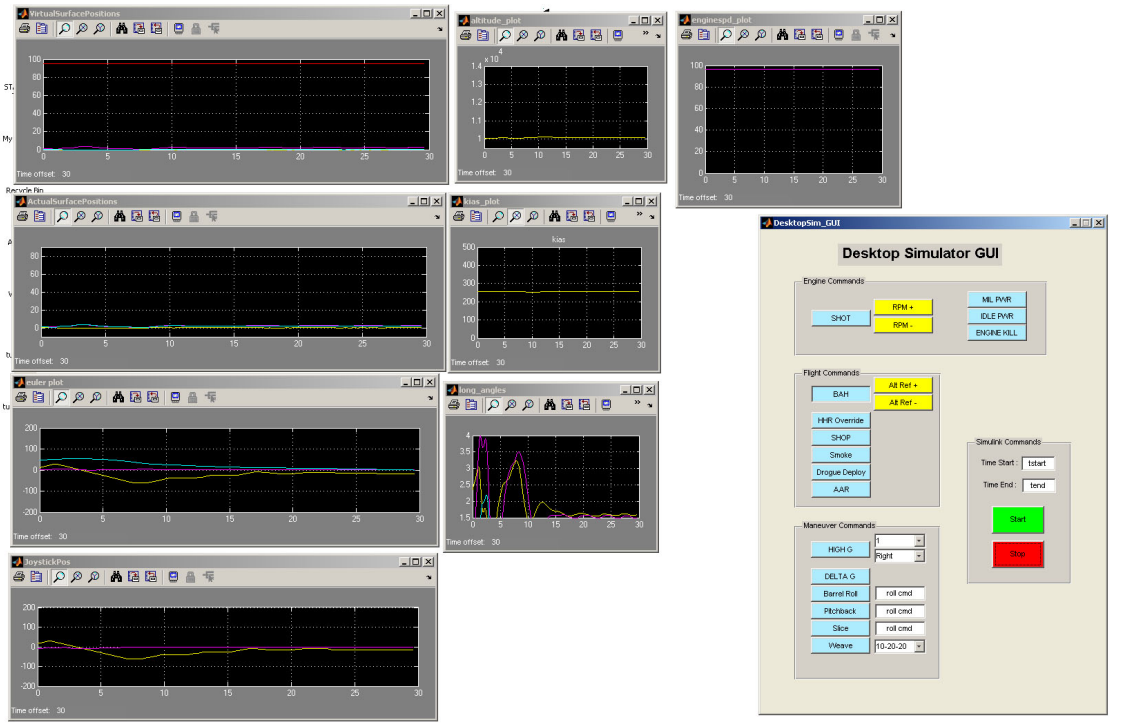


Figure 8.2.8: Simulink scopes with Desktop Simulator GUI.

a set of Mach and altitude scheduled gains, to the various control loops, based on the input of flight condition provided it by the *sensors* block.

The following figure presents a simple autopilot structure which maintains the the airplane six-degrees-of-freedom motion.

8.2.8 Visual Interface for Non-Linear Simulators

This section, though not necessary for engineer, presents a nice feature to have in the final 6DOF non-linear simulator, the visual simulation. Mainly, if not for any other reason, the visual interface is what will be seen by the your manager, the customer for the project, or even the end-user which is most likely a pilot to some degree. It is very unlikely that the event will arise where the end user will be interested in the

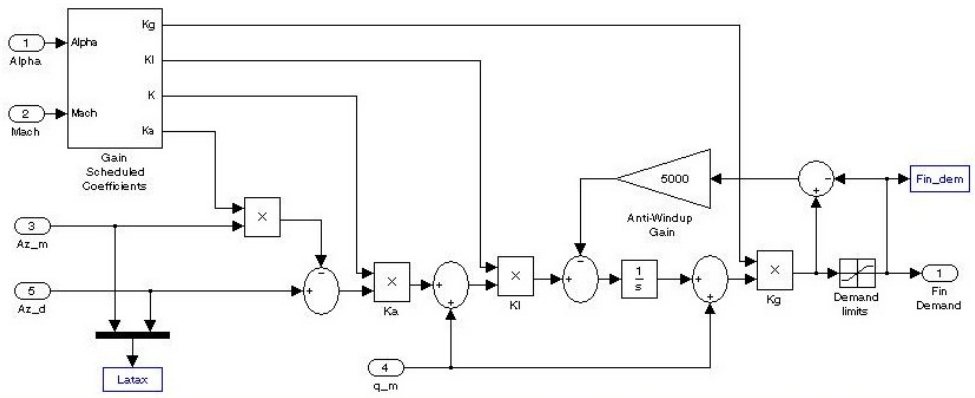


Figure 8.2.9: Simulink autopilot block diagram.

mathematics or software code used to develop the simulator. Therefore, this section introduces a rather simple method to build a physical/ visual simulation environment with a joystick, gages, buttons and a 3D airplane with a heads-up-display (HUD). To do this the author has chosen FlightGear, because it is easy to use and available as freeware from FlightGear.org. Additionally, a MatLab/Simulink package by AeroSim.org (ref) is available to university students free of charge. The AeroSim toolbox contains software examples and tools for building simulators. As this is an introduction, the specifics will not be discussed as these can be found in the literature, but the general overview should present enough information for the engineer to get started.

First, the 3D CAD model, OML only, should be exported as a .stl file. This is the basic geometry file which can be read by Flight Gear. Though it is possible to make a model with finer detail and complete geometry, the file size becomes an issue and cannot be opened with the versions of FlightGear that are supported by AeroSim.

Both AeroSim and FlightGear (Version 0.9.8) should be installed. Within the

FlightGear airplane directory, create and name a directory to hold the new airplane and supporting files: for examples, see other airplane directories within FlightGear. The remaining instructions are within the AeroSim documentation and included examples.

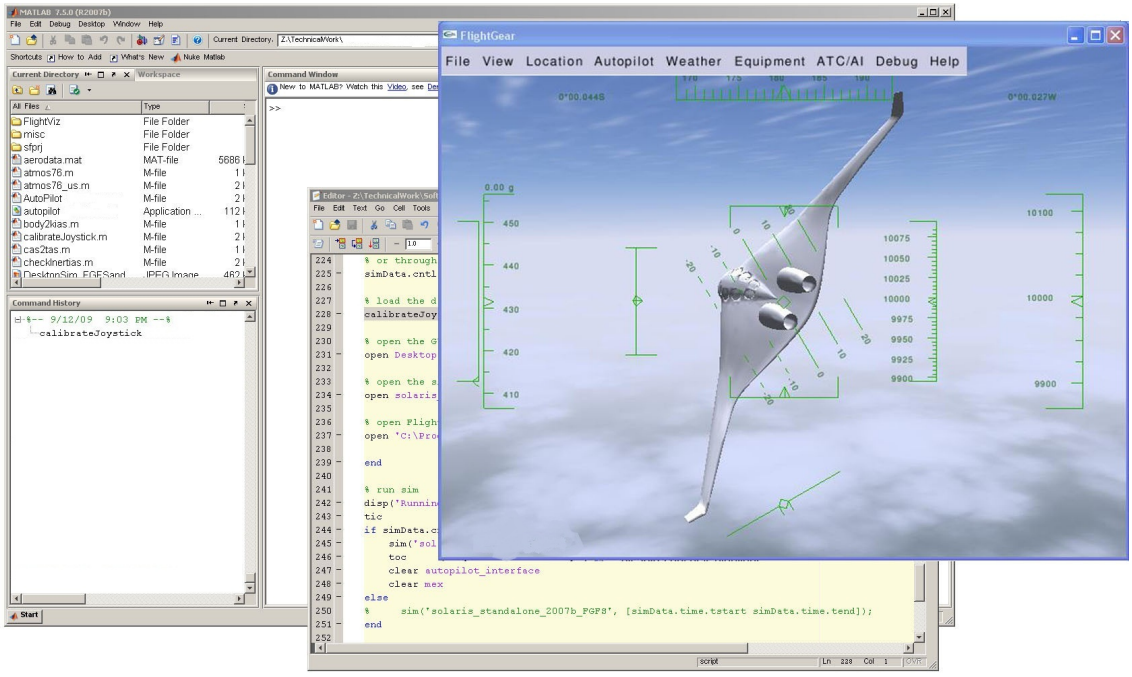


Figure 8.2.10: MatLab-Simulink with FlightGear Flight Sim.

8.3 Testing Validation and Verification: Methods and Techniques

A brief discussion on simulation test techniques follows:

Provided the non-linear simulator is well designed, a series of simple test can be

examined to determine the performance of the vehicle and the response of the avionics. Two examples will be provided here, a pitch response test and a roll response test. Both cases are common maneuvers performed during flight training in manned airplanes, therefore the maneuver requirements and expectations are well defined in literature.

8.3.1 Pitch Response Test

In the first case, the vehicle is subject to a 10 degree pitch input, which will create a 2G pitch-up maneuver, followed by a constant pitch climb maneuver at full throttle. This test will examine the system response to a discrete command, as well as the long-term effects on the system, namely decreasing airspeed.

At a first glance, the pitch response appears to be considerably slower than what was expected and/or desired. According to literature, the desired response is on the order of 2-2.5 seconds, and in the case of a UAV this could be reduced. However in the case of a manned vehicle, it might be more desirable to have a slower response so as to not disturb the passengers on board. In the first figure, we can see that the rise time is nearly 2 seconds and did not maintain pitch. A further look at the control surfaces, and the Normal Force result, it was determined that the pitch response at this flight condition was limited by a 3.5G limiter placed in the autopilot. This autopilot issues was later removed, but provides here a good example of what one might expect to see, versus what actually results.

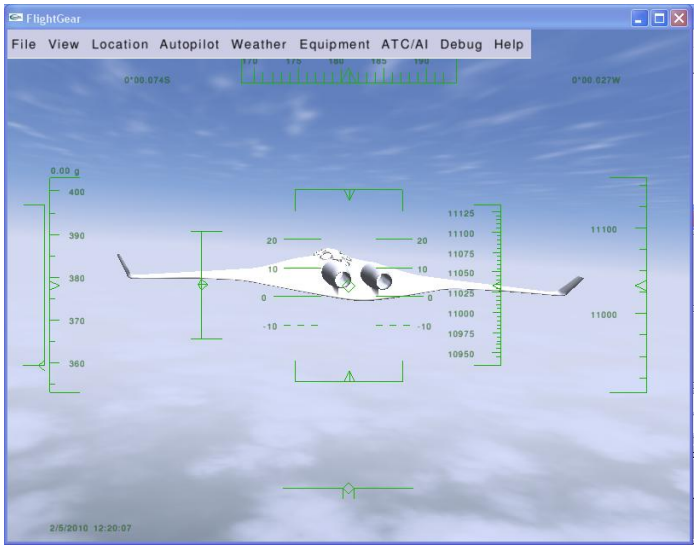


Figure 8.3.1: MatLab-Simulink pitch command results: FlightGear Visual Interface.

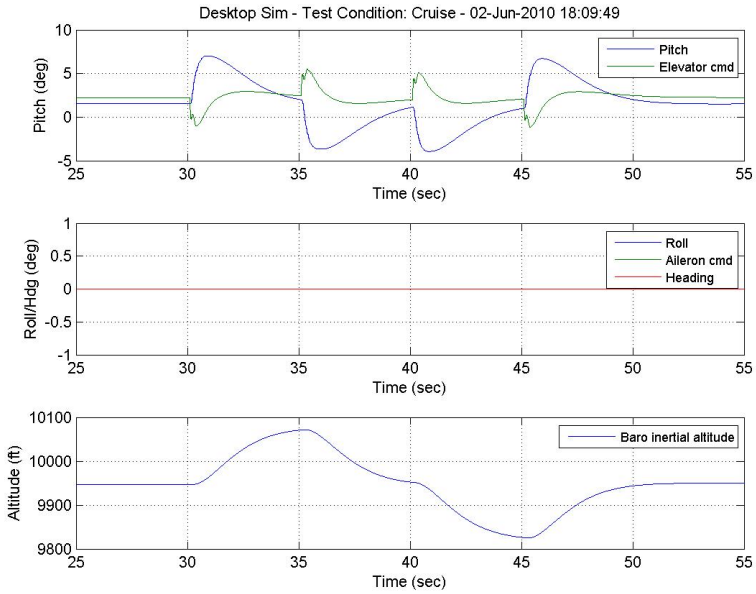


Figure 8.3.2: MatLab-Simulink pitch command results: Euler Angles, Altitude.

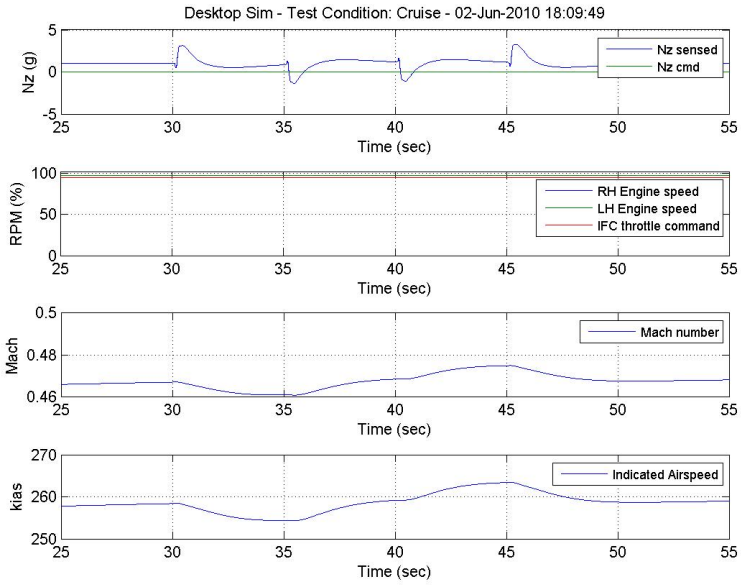


Figure 8.3.3: MatLab-Simulink pitch command results: Normal Force, %RPM, Velocity.

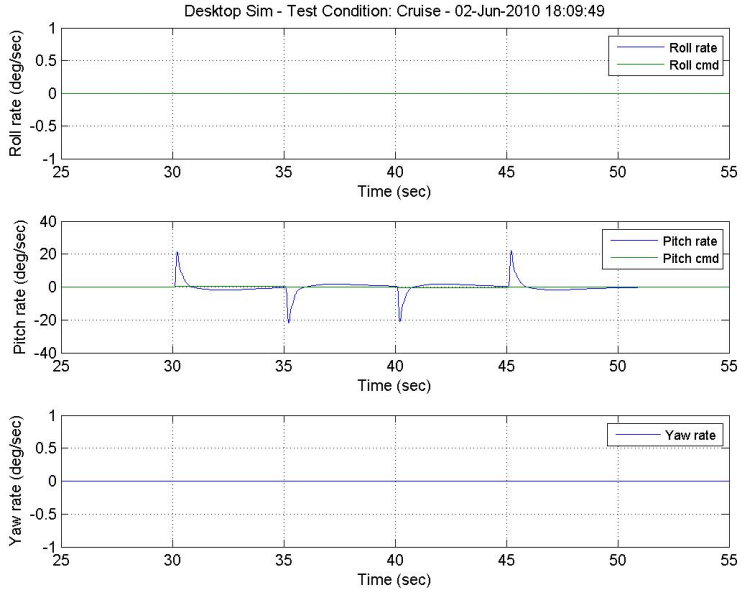


Figure 8.3.4: MatLab-Simulink pitch command results: Euler Rates.

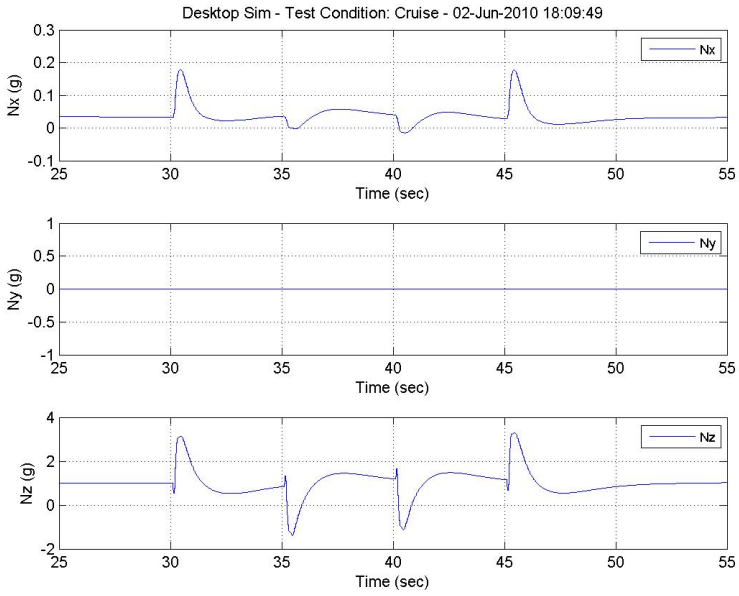


Figure 8.3.5: MatLab-Simulink pitch command results: Forces.

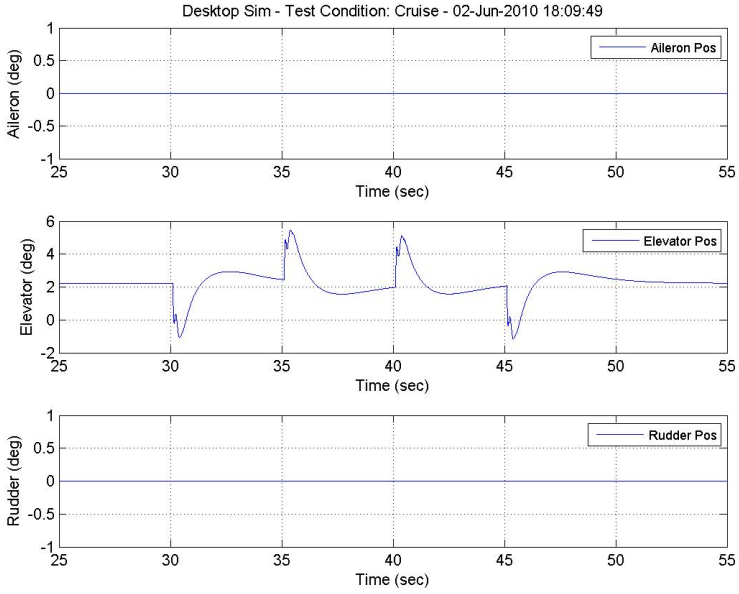


Figure 8.3.6: MatLab-Simulink pitch command results: Control surface commands and response.

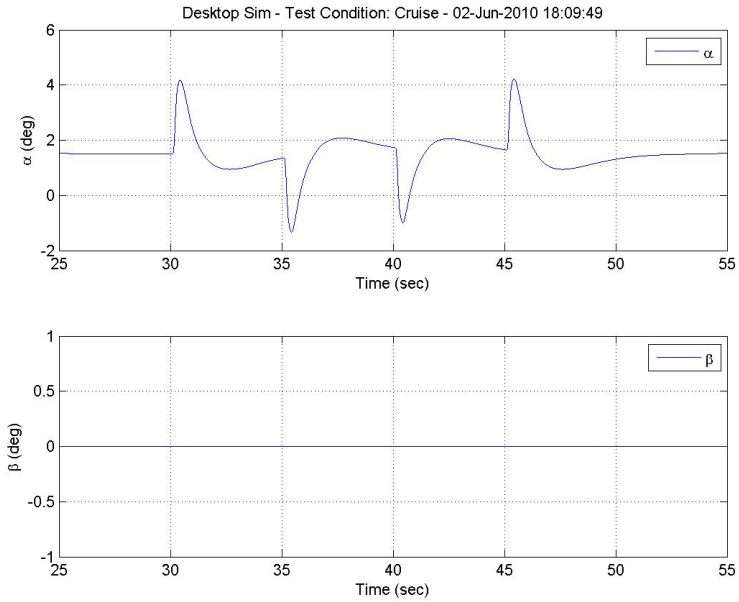


Figure 8.3.7: MatLab-Simulink pitch command results: Angle of attack and side slip.

8.3.2 Roll Response Test

The second case examines the system response to a 60 degree roll input followed by a constant altitude hold maneuver at full throttle. This test will examine the system response to the discrete roll command, the system's ability to damp the roll rate at 60 degrees, and the ability to hold altitude with rudder and throttle input, throttle being the more dominant factor for altitude hold in this maneuver.

The roll response appears to be as desired. The system responded quickly to the roll command, achieved the roll position at a reasonable rate — limited by the autopilot— with minimal overshoot and oscillation. And the system promptly integrated the yaw angle and yaw rate to maintain altitude throughout the maneuver with a minimal and acceptable gain in altitude throughout the maneuver. This is an excellent, but not perfect, execution of a roll command for a 2G altitude hold turn

maneuver. Altitude gain would have been reduced if the energy state was reduced, i.e. the throttle was reduced to hold altitude.

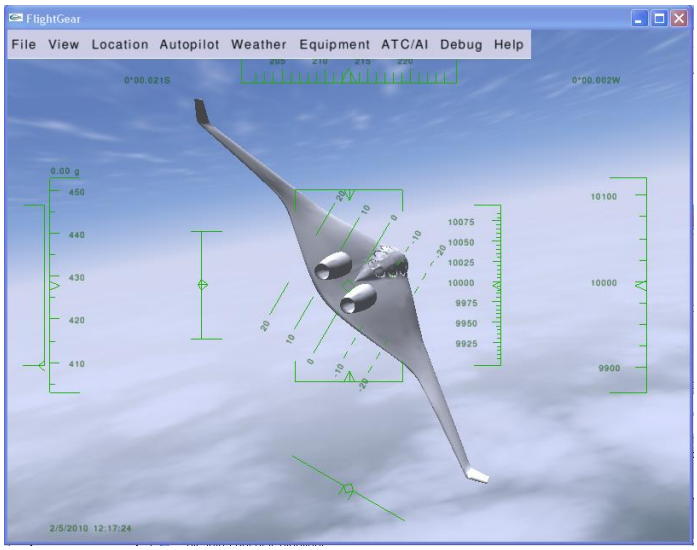


Figure 8.3.8: MatLab-Simulink roll command results: FlightGear Visual Interface.

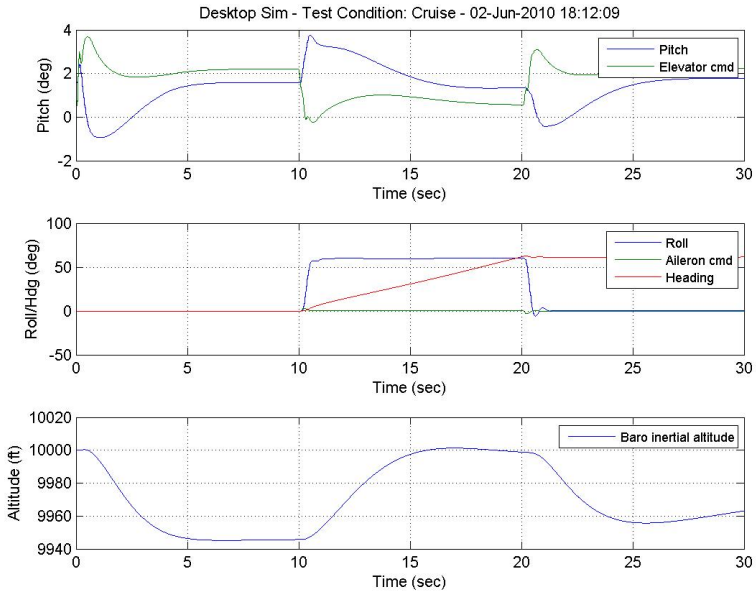


Figure 8.3.9: MatLab-Simulink roll command results: Euler Angles, Altitude.

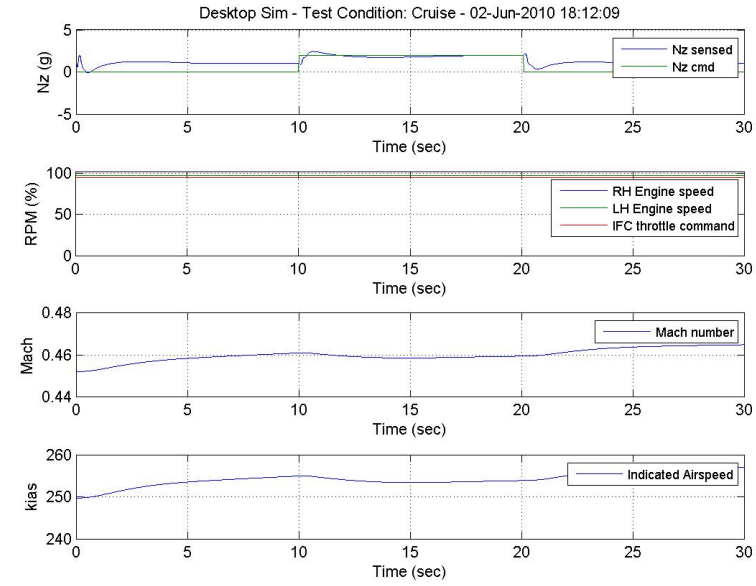


Figure 8.3.10: MatLab-Simulink roll command results: Normal Force, %RPM, Velocity.

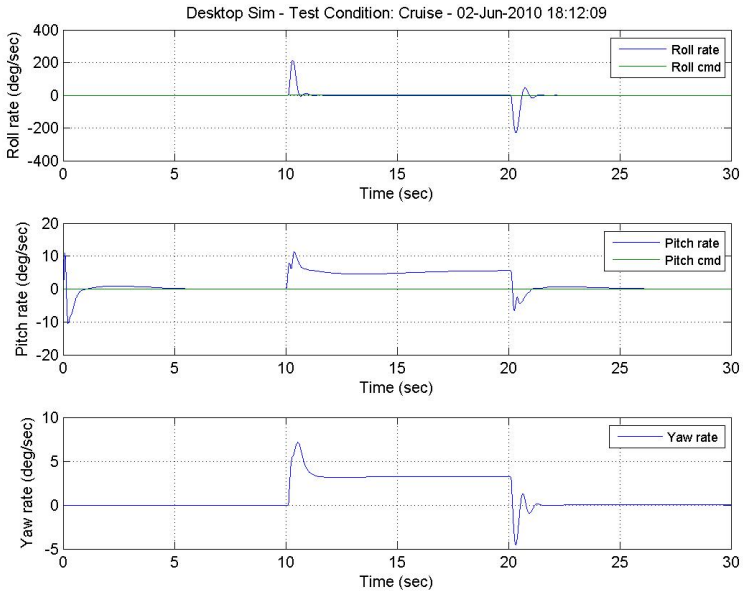


Figure 8.3.11: MatLab-Simulink roll command results: Euler Rates.

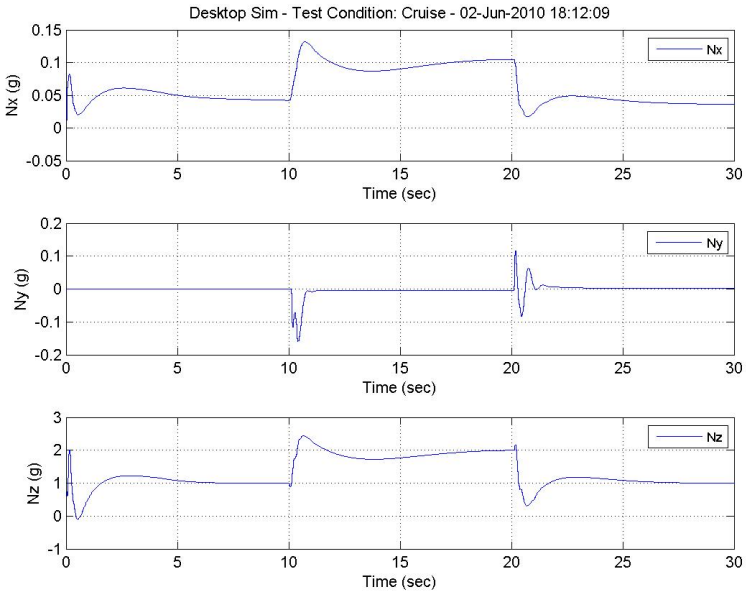


Figure 8.3.12: MatLab-Simulink roll command results: Forces.

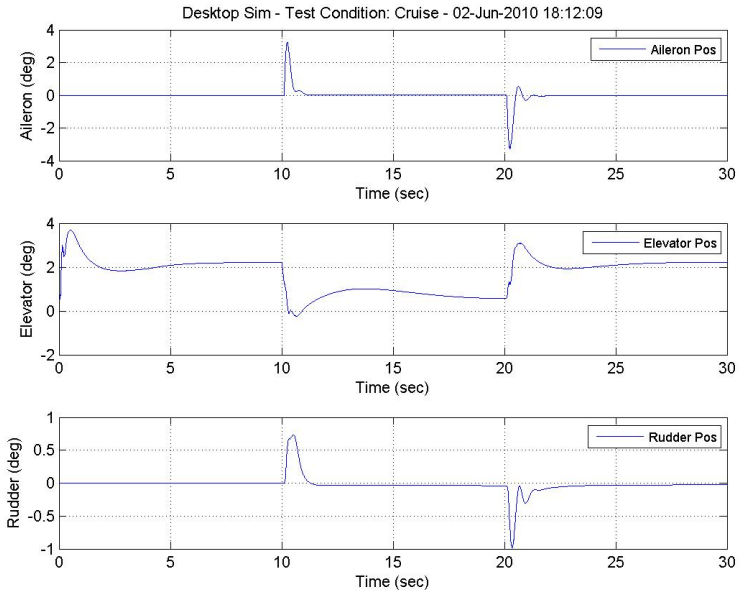


Figure 8.3.13: MatLab-Simulink roll command results: Control surface commands and response.

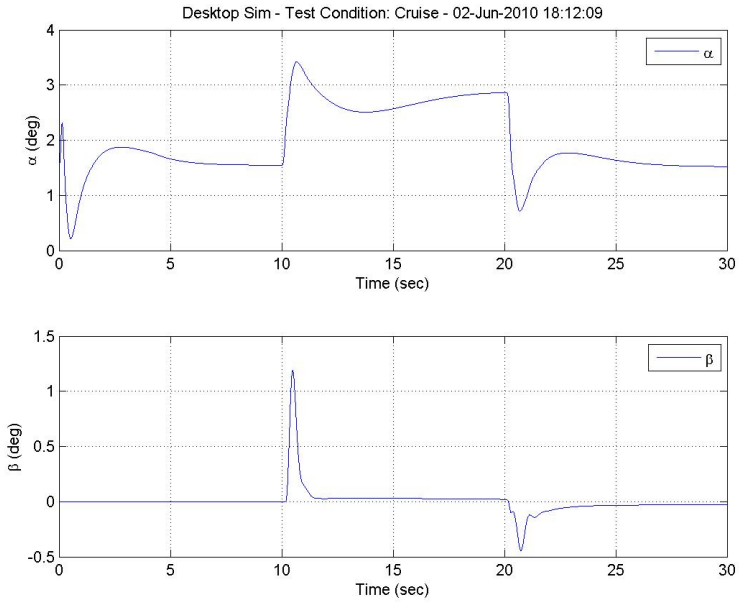


Figure 8.3.14: MatLab-Simulink roll command results: Angle of attack and side slip.

Chapter 9

Concluding Remarks

9.1 Summary

This thesis reported the investigation, design and practical implementation of a 6DOF non-linear simulator.

The main purpose of this thesis was to find a set of industry codes that were relatively inexpensive, and combine these codes into a tool which was capable of a rapid development (in less than few weeks) of a new airplane configuration. The investigation revealed a large quantity of literature from which to compare results, but more importantly this investigation revealed the relatively few codes necessary for the conceptual and preliminary design phases of airplanes.

It was demonstrated that a specific airfoil could be selected based on requirements, without prior knowledge of the airfoil-family capabilities. Additionally, this airfoil could be moderately optimized to meet the specific needs of the configuration.

The airplane geometry could be initially designed in a spreadsheet, to meet the various requirements, and then optimized in a vortex-lattice-method code, Athena Vortex Lattice (AVL). Coupling AVL with MatLab, one could perform an optimization on the airplane geometry, to derive a configuration based on a design requirement or goal. This analysis additionally provided the aero data necessary for both linear and non-linear simulation. The aero data was validated using full-Navier-Stokes 3D computational fluid dynamic simulation results, with recommendations for further tuning the numerical models with wind tunnel data.

The mass properties of the vehicle were shown to be easy to estimate, initially, with increased fidelity in CAD modeling; and thereby creating the configuration specific polynomial-based mass property equations required for simulation.

Two methods for propeller propulsion modeling were demonstrated; the first, derived from equations presented in literature, and the second using a blade-element-theory code. In both cases the results closely matched a wind tunnel test data for the same propeller-motor combinations which were investigated. It was also shown that a similar approach could be applied to a jet-engine simulation using equations and codes from Mattingly [34].

Finally, the linear and non-linear simulation models were presented, with a detailed description of the major components of the simulators; highlighting the ease with which the previous analysis data could be implemented and tested in a full non-linear environment. As well as the potential and necessity for non-linear simulators in the airplane design process, particularly in the process prior to expensive testing

and manufacturing of a finished product.

The following contributions were made during the course of the project towards the development of future UAV's and small airplane:

- Batch and Optimization routines for XFOIL using MatLab.
- Batch and Optimization routines for AVL using MatLab.
- Aerodynamics Module for Simulink based on AVL-MatLab Batch mode results.
- Propulsion Module for Simulink based on QPROP-MatLab Batch mode results.
- Comprehensive, all-inclusive, spreadsheet approach to airplane design.

9.2 Recommendations for Future Work

The author encourages the reader to use the methods supplied in this thesis as a basis from which to further tune the numerical models with experimental data. In all cases, the numerical models are only as good as the conditions and data supplied to it. It is therefore imperative to tune a non-linear simulator prior to "edge of the envelope" testing.

Furthermore, for pilot training, testing of a ground control station, and testing of airplane systems, it is necessary to build a hardware-in-the-loop (HWIL) simulator. While a software-in-the-loop (SWIL) is an excellent tool for accelerated testing the airplane flight dynamics and guidance, navigation, and control algorithms. An HWIL is capable of running the airplane systems in real-time, which provides a true pilot-training tool and system evaluation tool. A SWIL is capable of evaluation similar to an HWIL, only if the software is run on a separate machine. Serious limitations

exist when SWIL software is run in on the same machine as an existing operating system like $Windows^{TM}$— if real-time performance is desired. In the case of this thesis work, the author evaluated the airplane performance on an accelerated basis (for flight dynamic analysis and control command response evaluation) and therefore an SWIL running on $Windows^{TM}$ was acceptable.

References

[1] Abbott, I.H., von Doenhoff, A.E., *Theory of Wing Sections: Including a Summary of Airfoil Data*, Dover Publications, 1959.

[2] Aerosonde, *Aerosonde Weather Tracking UAV*, 2003, <http://www.aerosonde.com>.

[3] Analytical Methods, Inc. <http://www.am-inc.com>.

[4] Anderson, J., *Fundamentals of Aerodynamics*, McGraw-Hill, 4th ed., 2005.

[5] Anderson, J.D. Jr., "Computational Fluid Dynamics – An Engineering tool?", *Numerical/Laboratory Computer Methods in Fluid Dynamics*, Edited by A.A. Pouring, ASME, 1976, pp. 1-12.

[6] Ashley, H., Landahl, M., *Aerodynamics of Wings and Bodies*, Dover Publications, 1985.

[7] Barlow, J.B., Rae, W.H., Pope, A., *Low-Speed Wind Tunnel Testing*, Wiley-Interscience, 3rd ed., 1999.

[8] Day, R.A., *How to Write & Publish a Scientific Paper*, Oryx Press, 5th ed., 1998.

[9] Designation Systems. *A-series Drones*, 2003, <http://www.designation-systems.net/dusrm/app1/a-target.html>.

[10] Designation Systems. *Radio Plane OQ-14/TDD*, 2003, <http://www.designation-systems.net/dusrm/app1/oq-14.html>.

[11] Designation Systems. *MQ-1 Predator*, 2003, <http://www.designation-systems.net/dusrm/app2/q-1.html>.

[12] Designation Systems. *MQ-4 Global Hawk*, 2003, <http://www.designation-systems.net/dusrm/app2/q-4.html>.

[13] Drela, M., Youngren, H., XFOIL 6.94 User Guide. MIT Aero & Astro. 10 Dec 2001. <http://web.mit.edu/drela/Public/web/xfoil/>.

[14] Drela, M., Youngren, H., AVL 3.26 User Primer. MIT Aero & Astro. 29 Apr 2006. <http://web.mit.edu/drela/Public/web/avl/>.

[15] Drela, M., Panel Method Review. MIT Aero & Astro. 12 June 2008. <http://web.mit.edu/drela/>.

[16] Drela, M., QPROP User Guide. MIT Aero & Astro. 6 July 2007. <http://web.mit.edu/drela/Public/web/qprop/>.

[17] Drela, M., ASWING User Guide. MIT Aero & Astro. 8 Sept 2008. <http://web.mit.edu/drela/Public/web/aswing/>.

[18] Green, J.E., "Numerical Methods in aeronautical fluid dynamics—An introduction", Edited by P.L. Roe, *Numerical Methods in aeronautical fluid dynamics*, 1982, pp. 1-32.

[19] Hacker, D., *The Bedford Handbook*, Bedford St. Martin's, 5th ed., 1999.

[20] Hays, A.P., "Spreadsheet Methods for Aircraft Design",American Institute of Aeronautics & Astronautics Aircraft Design, Systems and Operations Conference, July 1989, AIAA-89-2059.

[21] Herbert, H.E, Lamar, J.E., "Production Version of the Extended NASA-Langley Vortex Lattice FORTRAN Computer Program", Vol. II. Source Code, 1982, NASA TM 83304.

[22] Hoak, D.E., Ellison, D.E. *et.al.*, USAF Stability & Control DATCOM, Flight Control Division, Air Force Flight Dynamics Laboratory, Wright Patterson Air Force Base, Ohio, Vol 1-3., April 1979, AFFDL-TR-79-3032.

[23] Hoerner, S.F., *Fluid Dynamic Drag*, Hoerner, 1992.

[24] Hoerner, S.F., Borst, H.V., *Fluid Dynamic Lift*, Hoerner, 2nd ed., 1985.

[25] Katz, J., Plotkin, A., *Low-Speed Aerodynamics*, Cambridge University Press, 2nd ed., 2001.

[26] Kimberlin, R.D., *Flight Testing of Fixed-Wing Aircraft*, AIAA Education Series, 2003.

[27] Klein, V., Morelli, E.A., *Aircraft System Identification: Theory And Practice.* AIAA Education Series, 2006.

[28] Lamar, J.E., Gloss, B.B., "Subsonic Aerodynamic Characteristics of Interacting Lifting Surfaces With Separated Flow Around Sharp Edges Predicted by a Vortex-Lattice Method", 1975, NASA TN D-7921.

[29] Lamar, J.E., Herbert, H.E., "Production Version of the Extended NASA-Langley Vortex Lattice FORTRAN Computer Program", Vol. I. Users Guide, (requires update packet, July, 1984), 1982, NASA TM 83303.

[30] Lazzara, D.S., Haimes, R., Wilcox, K., "Multifidelity Geometry and Analysis in Aircraft Conceptual Design", 19th American Institute of Aeronautics & Astronautics Computational Fluid Dynamics Conference, June 2009, AIAA-2009-3806.

[31] Liebeck, R.H., "Design of the Blended Wing Body Subsonic Transport", American Institute of Aeronautics & Astronautics Journal of Aircraft, Vol. 41, No. 1, January-February 2004. AIAA-9084-368.

[32] Margason, R.J., Lamar, J.E., "Vortex-Lattice FORTRAN Program for Estimating Subsonic Aerodynamic Characteristics of Complex Planforms", 1971, NASA TN D-6142.

[33] The MathWorks. <http://www.mathworks.com>.

[34] Mattingly, J.D., "Elements of Gas Turbine Propulsion", McGraw-Hill, 1996.

[35] McCloud, D., "Weight and Balance Design Considerations for Flight Test Aircraft", Presented at the 63rd Annual Conference of Society of Allied Weight Engineers, Inc. Newport Beach, California, 17-19 May, 2004. SAWE Paper No. 3322, Category No. 10.

[36] McGarry, J., *et.al.*, "Design of an Engine Inertia Measuring Device", Presented at the 63rd Annual Conference of Society of Allied Weight Engineers, Inc. Newport Beach, California, 17-19 May, 2004. SAWE Paper No.3340, Category No. 6.

[37] Medusa Research, Inc. *Power Analyzer Plus, Operation Manual.* 2006. <http://www.MedusaProducts.com>.

[38] NASA, "Vortex Lattice Utilization Workshop", May 1976, NASA SP-405.

[39] Nelson, R., *Flight Stability and Automatic Control*, McGraw-Hill, 2nd ed., 1997.

[40] Nicolai, L., *Fundamentals of Aircraft Design*, Mets, Revised Edition, 1984.

[41] Peterson, W.L., "Mass Properties Measurements in the X-38 Project", Presented at the 63rd Annual Conference of Society of Allied Weight Engineers, Inc. Newport Beach, California, 17-19 May, 2004. SAWE Paper No.3325, Category 6.

[42] Raymer, D.P., *Aircraft Design: A Conceptual Approach*, AIAA Education Series, 4th ed., 2006.

[43] Raymer, D.P., "Modern Use of Spreadsheet Methods for Aircraft Design, Sizing, and Performance Analysis", American Institute of Aeronautics & Astronautics Aerospace Sciences Meeting, Reno, NV, 2004, AIAA-2004-0534.

[44] Roskam, J., Lan, C.E., *Airplane Aerodynamics and Performance*, Darcorporation, 1997.

[45] Roskam, J., Lan, C.E., *Airplane Design*, Darcorporation, Vol. 1-8, 2nd ed., 2003.

[46] Roskam, J., Lan, C.E., *Airplane Flight Dynamics and Automatic Flight Controls*, Darcorporation, Vol.1-2, 2003.

[47] Schrenk, O., "A simple approximation method for obtaining the spanwise lift distribution", National Advisory Committee for Aeronautics, April 1940, naca-tm-948.

[48] Soderman, P., Aiken, T., "Full-Scale Wind-Tunnel Tests of a Small Unpowered Jet Aircraft With a T-Tail", NASA Ames Research Center, November 1971), NASA TN D-6573.

[49] Society of Allied Weight Engineers, "Weight Engineer's Handbook", Society of Allied Weight Engineers Inc., May 2002.

[50] Strunk , W. Jr., White, E.B., *The Elements of Style*, Longman Publishers, 4th ed., 2000.

[51] Thomas, F., *Fundamentals of Sailplane Design*, College Park Press, 3rd ed, Trans. Judah Milgram, 1999.

[52] University of Illinois at Urbana-Champaign, Airfoil Coordinate Database. <http://www.ae.illinois.edu/m-selig/ads/coord_database.html>.

[53] Von Karman Institute for Fluid Dynamics, "Introduction to computational fluid dynamics—Lecture Series", *VKI Annual Lecture Series*, 2009.

[54] Wergeland, F., *The Great Motor Test*. 5th Ed. 2003-2004. <http://www.flyingmodels.org>.

[55] Zipfel, P.H. *Modeling and Simulation of Aerospace Vehicle Dynamics*, AIAA Education Series, 2001.

(This page intentionally blank.)

Appendix A

Excel Spreadsheets

(Data included on CD-ROM.)

Appendix B

XFOIL and AVL files

(Data included on CD-ROM.)

Appendix C

Non-Linear Simulation Files

(Data included on CD-ROM.)

(This page intentionally blank.)