



**Application and Validation of Off-Nominal
Aircraft Performance Models**

THESIS

Mark W. Vahle, Major, USAF
AFIT-ENY-MS-17-M-297

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENY-MS-17-M-297

APPLICATION AND VALIDATION OF OFF-NOMINAL AIRCRAFT
PERFORMANCE MODELS

THESIS

Presented to the Faculty
Department of Aeronautics and Astronautics
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Aeronautical Engineering

Mark W. Vahle, B.S.A.A.E., M.A.S., M.S.F.T.E.
Major, USAF

February 22, 2017

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENY-MS-17-M-297

APPLICATION AND VALIDATION OF OFF-NOMINAL AIRCRAFT
PERFORMANCE MODELS

THESIS

Mark W. Vahle, B.S.A.A.E., M.A.S., M.S.F.T.E.
Major, USAF

Committee Membership:

R. Cobb, PhD
Chair

J. Agte, PhD
Member

D. Kunz, PhD
Member

M. Reeder, PhD
Member

Abstract

Several real-world scenarios demand the exploration aircraft performance and flying qualities (P&FQ) in off-nominal or adverse conditions. These topics include environmentally-induced events (shear or wake vortex encounters), stalls or departures, on-board system failures, and structural failures including battle damage. When faced with on-board system failures (i.e. control surface failures), recent developments in multistate design have shown great promise in improving system reliability through early stage design consideration of off-nominal performance. Multistate design involves representing aircraft flight controls as different elements having a cumulative effect on the entire system. Meaning the performance rates of each element composing the system can contribute to the degradation of the entire system performance. Using knowledge of element reliability, an aircraft design can be optimized to increase overall system availability. Three primary areas that affect the accuracy (uncertainty) of multistate design decisions are 1) the choice of control input, 2) the degree in which one can model response to that input and 3) prediction of component failure rates.

Accurately modeling the response to a control input typically requires a high-fidelity aerodynamic model and simulation routine with the ability to predict off-nominal aircraft performance. Unfortunately, the modeling data that is typically used to develop these simulation routines, i.e. aerodynamic data, control derivatives, mass and inertia properties, etc., is either a) generically generated from simplified geometric analysis assuming everything is working properly, or b) flight test data collected only from the nominal state of performance.

Utilizing a variable stability in-flight simulator (IFS), nominal and off-nominal

P&FQ data was collected and compared against data from a high-fidelity 6-DoF simulation software package.

Presented in this research is a methodology to verify and validate off-nominal aircraft performance utilizing a low-cost open-source flight dynamics simulator (JSBSim). The central objective of the research was a determination of the degree of accuracy in which one can model off-nominal performance in order to make early stage design decisions required by the multistate design process. Results indicate that nominal and off-nominal P&FQ could be predicted inside Federal Aviation Administration (FAA) simulator qualification criteria at low-cost and with a reasonable amount of effort.

Acknowledgements

First and foremost, I would like to thank Lt Col Jeremy Agte for his inspiration and guidance through this project. His previous work in multistate design optimization through simulation provided an extremely strong foundation, in which this research was built upon.

Additional thanks to my thesis adviser, Dr. Richard Cobb, for ensuring that this research project was the best that it could be by providing insight, motivation and encouragement. I would also like to acknowledge my Test Pilot School (TPS) test management project (TMP) team: Captain Samantha Thorn, Captain Mark Gabbard, Captain Matthew Welch, and Major Gregory Marcus for their assistance in 12 hours of flight test and over 200+ hours of work collecting and analyzing the data from the test cases. Without their tireless efforts, this thesis would not be what it is today. I'd also like to thank 1Lt Joshua Kim, for his keen insight into JSBSim.

Mark W. Vahle

Table of Contents

	Page
Abstract	iv
Acknowledgements	vi
List of Figures	x
List of Tables	xvi
I. Introduction	1
1.1 Background	1
1.2 Research Problem Motivation and Description	3
1.3 Research Objectives and Scope	4
1.4 Case Study: Twin-Engine Learjet 25D	5
1.5 6-DoF Simulation Aircraft Model	6
1.6 Chapter Summary	7
II. Literature Review	9
2.1 Aircraft Design Optimization	9
2.1.1 Multistate Design Optimization	11
2.2 System ID and Model Verification	15
2.2.1 Modeling Aircraft with Failed Systems/Battle Damage	17
2.3 Aerodynamics of Aircraft with Failed Systems	18
2.3.1 Parameter Estimation and Identification	22
2.4 Flight Test and Simulation Integration	26
2.4.1 Types of Simulation	27
2.5 Data Analysis	28
2.5.1 Filtering	29
2.5.2 Smoothing	29
2.5.3 Evaluation Criteria	30
2.6 Chapter Summary	33
III. Research Methodology	34
3.1 Simulation Model State and Input Vectors	34
3.2 Aircraft Capabilities and Limitations	36
3.3 Post-flight Calculations	39
3.3.1 Variations in Aircraft Center of Gravity	40
3.3.2 Mass Moment and Product of Inertia	41
3.3.3 Non-standard Temperature Effects	42
3.4 Experiment Design	43

	Page
3.4.1 Simulation Control Integration	45
3.4.2 Planned Test Maneuvers	46
3.5 Methodology.....	47
3.5.1 Trim Method	48
3.5.2 Nominal Open-Loop Flying Qualities.....	52
3.5.3 Baseline Performance Maneuver	61
3.6 Chapter Summary	68
IV. Results	69
4.1 Data Presentation	69
4.2 Trim.....	70
4.2.1 Nominal trim	71
4.2.2 Off-Nominal Trim	73
4.3 Nominal Longitudinal and Lateral-Directional Modes.....	78
4.3.1 Short Period Dynamics.....	78
4.3.2 Phugoid Response	81
4.3.3 Roll Response.....	82
4.3.4 Dutch Roll Dynamics	83
4.3.5 Spiral Stability	84
4.3.6 Open-Loop Flying Qualities (OLFQ) Summary	86
4.4 Performance Maneuver Case Studies	87
4.4.1 Data Presentation/Analysis	87
4.4.2 Nominal.....	88
4.4.3 Rudder Failure	92
4.4.4 Aileron Failure	97
4.4.5 Engine Failure	101
4.4.6 Engine and Aileron Failure	106
4.4.7 Engine and Rudder Failure	111
4.4.8 Aileron and Rudder Failure	115
4.4.9 Elevator Failure	118
4.5 Chapter Summary	118
V. Conclusions and Recommendations	119
5.1 Results Summary	120
5.2 Conclusions.....	123
5.3 Recommendations for Future Work	126
Appendix A. JSBSim 6-DoF Simulation Model	129
A Introduction	129
A.1 Equations of Motion	130
A.2 Aircraft Modeling	131
A.3 MATLAB Integration	135

	Page
Appendix B. JSBSim C++ Code Modification	137
Appendix C. Learjet 25D Modeling Data	145
Appendix D. JSBSim Learjet 25D XML Aircraft Configuration Code	148
Appendix E. JSBSim Learjet 25D XML Autopilot Configuration Code	167
Appendix F. Supporting Figures	179
Bibliography	231
Vita	237

List of Figures

Figure		Page
1.	Calspan Corporation Learjet 25D In-Flight Simulator [53].	6
2.	Learjet VSS Cockpit [53]	37
3.	Learjet Aileron System [12]	38
4.	Learjet Elevator System [12]	38
5.	Learjet Rudder System [12]	39
6.	Doublet Input Example	54
7.	Step Input Example	57
8.	Baseline Performance Maneuver	62
9.	Performance Maneuver Control History Isolation	65
10.	VSS Control Channel Isolation	66
11.	Maneuver Quality Assessment (Example)	67
12.	Nominal Trim: Pitch and Elevator-Stabilizer Deflection	71
13.	Nominal Trim: Net Thrust	72
14.	Off-Nominal Trim: Pitch and Elevator-Stabilizer Deflection	74
15.	Off-Nominal Trim: Rudder Deflection and Bank Angle	75
16.	Off-Nominal Trim: Rudder Deflection and Bank Angle (bias removed)	76
17.	Off-Nominal Trim: Net Thrust	77
18.	Short Period Dynamics: Pitch Response	79
19.	Short Period Dynamics: Pitch Rate Response	80
20.	Roll Response	82
21.	Dutch Roll Dynamics: Yaw Rate	83

Figure		Page
22.	Spiral Stability	85
23.	Nominal Configuration (500 fpm): Control Deflections	89
24.	Nominal Configuration (500 fpm): Body-Axis Rates	90
25.	Nominal Configuration (500 fpm): Thrust	91
26.	Rudder Failed Configuration (1000 fpm): Control Deflections	93
27.	Rudder Failed Configuration (1000 fpm): Thrust	94
28.	Rudder Failed Configuration (1000 fpm): Individual Engine Thrust	95
29.	Rudder Failed Configuration (1000 fpm): Angle of Sideslip	96
30.	Ailerons Failed Configuration (500 fpm): Control Deflections (Bias Present)	98
31.	Ailerons Failed Configuration (500 fpm): Control Deflections (Bias Removed)	99
32.	Ailerons Failed Configuration (500 fpm): Thrust	100
33.	Single Engine Right-Hand Turn into Idle Engine (500 fpm): Control Deflections	102
34.	Single Engine Right-Hand Turn into Idle Engine (500 fpm): Split Thrust	103
35.	Single Engine Left-Hand Turn away from Idle Engine (500 fpm): Control Deflections	104
36.	Single Engine Left-Hand Turn away from Idle Engine (500 fpm): Split Thrust	105
37.	Single Engine and Aileron Failed (RHT) at 500 fpm: Deflections	107
38.	Single Engine and Aileron Failed (RHT) at 500 fpm: Split Thrust	108

Figure	Page
39. Single Engine and Aileron Failed (LHT) at 500 fpm: Control Deflections	109
40. Single Engine and Aileron Failed (LHT) at 500 fpm: Split Thrust	110
41. Single Engine and Rudder Failed (RHT) at 500 fpm: Control Deflections	112
42. Single Engine and Rudder Failed (RHT) at 500 fpm: Split Thrust	113
43. Single Engine and Rudder Failed (LHT) at 500 fpm: Control Deflections	114
44. Single Engine and Rudder Failed (LHT) at 500 fpm: Split Thrust	115
45. Ailerons and Rudder Failed at 500 fpm: Control Deflections	116
46. Ailerons and Rudder Failed at 500 fpm: Individual Engine Thrust	117
A.1. JSBSim Switch Component	134
A.2. JSBSim S-Function Example	136
F.1. Nominal Configuration (1000 fpm): Control Deflections	179
F.2. Nominal Configuration (1000 fpm): Body-Axis Rates	180
F.3. Nominal Configuration (1000 fpm): Thrust	181
F.4. Nominal Configuration (2000 fpm): Control Deflections	182
F.5. Nominal Configuration (2000 fpm): Body-Axis Rates	183
F.6. Nominal Configuration (2000 fpm): Thrust	184
F.7. Rudder Failed Configuration (500 fpm): Control Deflections	185
F.8. Rudder Failed Configuration (500 fpm): Body-Axis Rates	186

Figure		Page
F.9.	Rudder Failed Configuration (500 fpm): Thrust	187
F.10.	Rudder Failed Configuration (500 fpm): Individual Engine Thrust	188
F.11.	Rudder Failed Configuration (2000 fpm): Control Deflections	189
F.12.	Rudder Failed Configuration (2000 fpm): Body-Axis Rates	190
F.13.	Rudder Failed Configuration (2000 fpm): Thrust	191
F.14.	Rudder Failed Configuration (2000 fpm): Individual Engine Thrust	192
F.15.	Aileron Failed Configuration (500 fpm): Rates	193
F.16.	Aileron Failed Configuration (1000 fpm): Control Deflections	194
F.17.	Aileron Failed Configuration (1000 fpm): Rates	195
F.18.	Aileron Failed Configuration (1000 fpm): Thrust	196
F.19.	Aileron Failed Configuration (2000 fpm): Control Deflections	197
F.20.	Aileron Failed Configuration (2000 fpm): Rates	198
F.21.	Aileron Failed Configuration (2000 fpm): Thrust	199
F.22.	Single Engine Right-Hand Turn into Idle Engine (500 fpm): Rates	200
F.23.	Single Engine Left-Hand Turn away from Idle Engine (500 fpm): Rates	201
F.24.	Single Engine Right-Hand Turn into Idle Engine (1000 fpm): Control Deflections	202
F.25.	Single Engine Right-Hand Turn into Idle Engine (1000 fpm): Rates	203
F.26.	Single Engine Right-Hand Turn into Idle Engine (1000 fpm): Individual Engine Thrust	204

Figure	Page
F.27. Single Engine Left-Hand Turn away from Idle Engine (1000 fpm): Control Deflections	205
F.28. Single Engine Left-Hand Turn away Idle Engine (1000 fpm): Rates	206
F.29. Single Engine Left-Hand Turn away from Idle Engine (1000 fpm): Individual Engine Thrust	207
F.30. Single Engine and Aileron Failed (RHT) at 500 fpm: Rates	208
F.31. Single Engine and Aileron Failed (LHT) at 500 fpm: Rates	209
F.32. Single Engine and Aileron Failed (RHT) at 1000 fpm: Control Deflections	210
F.33. Single Engine and Aileron Failed (RHT) at 1000 fpm: Rates	211
F.34. Single Engine and Aileron Failed (RHT) at 1000 fpm: Split Thrust	212
F.35. Single Engine and Aileron Failed (LHT) at 1000 fpm: Control Deflections	213
F.36. Single Engine and Aileron Failed (LHT) at 1000 fpm: Rates	214
F.37. Single Engine and Aileron Failed (LHT) at 1000 fpm: Split Thrust	215
F.38. Single Engine and Rudder Failed (RHT) at 500 fpm: Rates	216
F.39. Single Engine and Rudder Failed (LHT) at 500 fpm: Rates	217
F.40. Single Engine and Rudder Failed (RHT) at 1000 fpm: Control Deflections	218
F.41. Single Engine and Rudder Failed (RHT) at 1000 fpm: Rates	219

Figure		Page
F.42.	Single Engine and Rudder Failed (RHT) at 1000 fpm: Split Thrust	220
F.43.	Single Engine and Rudder Failed (LHT) at 1000 fpm: Control Deflections	221
F.44.	Single Engine and Rudder Failed (LHT) at 1000 fpm: Rates	222
F.45.	Single Engine and Rudder Failed (LHT) at 1000 fpm: Split Thrust	223
F.46.	Ailerons and Rudder Failed (500 fpm): Rates	224
F.47.	Ailerons and Rudder Failed (1000 fpm): Control Deflections	225
F.48.	Ailerons and Rudder Failed (1000 fpm): Rates	226
F.49.	Ailerons and Rudder Failed (1000 fpm): Split Thrust	227
F.50.	Ailerons and Rudder Failed (2000 fpm): Control Deflections	228
F.51.	Ailerons and Rudder Failed (2000 fpm): Rates	229
F.52.	Ailerons and Rudder Failed (2000 fpm): Split Thrust	230

List of Tables

Table		Page
1.	Table of PID Autopilots	47
2.	Test Point Matrix	48
3.	Longitudinal Trim Metric [25, p. 55]	52
4.	Engine Inoperative Trim [25, p. 58]	52
5.	Short Period programmable test input (PTI)	54
6.	Pitch Response and Short Period Dynamics [25, p. 57]	55
7.	Phugoid Dynamics [25, p. 57]	56
8.	Roll Response PTI	58
9.	Roll (Aileron) Response [25, p. 58]	58
10.	Dutch Roll PTI	59
11.	Yaw (Rudder) Response [25, p. 58] and Dutch Roll (Yaw Damper OFF) [25, p. 59]	59
12.	Spiral Stability [25, p. 58]	60
13.	Baseline Performance Maneuver	61
14.	Table of Autopilot Switch Settings	64
15.	Performance Maneuver Tolerances	66
16.	FAA <i>14 CFR 60</i> Criteria Summary [25]	68
17.	Nominal Trim Theil's inequality coefficient (TIC)	73
18.	Off-Nominal Trim TIC	78
19.	Short Period Modal Parameters	80
20.	Short Period Dynamics TIC	81
21.	Phugoid Modal Parameters	81
22.	Roll Response Time Constants	82

Table	Page
23. Roll Response TIC	83
24. Dutch Roll Modal Parameters	84
25. Dutch Roll Dynamics TIC	84
26. Spiral Stability TIC	86
27. Lateral-Directional and Longitudinal Mode Solution Comparison	86
28. Nominal Configuration TIC (500 fpm)	92
29. Rudder Failed Configuration TIC (1000 fpm)	97
30. Single Engine TIC (500 fpm)	106
31. Results Summary: Trim and open-loop flying qualities (OLFQ)	121
32. Results Summary: Performance Maneuver	122
C.1. Learjet 25D Geometric Parameters [45]	145
C.2. Learjet 25D Flight Control Parameters [12]	145
C.3. Learjet 25D Stability and Control Derivatives	146
C.4. Learjet 25D Stability and Control Derivatives (cont.)	147

List of Acronyms

CFD computational fluid dynamics

c.g. center of gravity

DoD Department of Defense

DoF degree of freedom

DATCOM data compendium

DAS data acquisition system

DARPA Defense Advanced Research Projects Agency

EPR exhaust pressure ratio

FAA Federal Aviation Administration

FBW fly-by-wire

FDM flight dynamics model

ID identification

IFS in-flight simulator

ISA International Standard Atmosphere

HALE high-altitude, long-endurance

HBL Hotbench Lite

HWIL Hardware-In-The-Loop

KIAS knots indicated airspeed

LHT left-hand turn

MALE medium-altitude, long-endurance

MEL minimum equipment list

MLE maximum likelihood estimation

MITL Man-In-The-Loop

MSS multistate system

MSL mean sea level

NASA National Aeronautics and Space Administration

NATO North Atlantic Treaty Organization

OLFQ open-loop flying qualities

P&FQ performance and flying qualities

PID proportional integral derivative

PTI programmable test input

RHT right-hand turn

RMS root-mean-square

RTO Research and Technology Organization

SAS stability augmentation system

SLUF straight, level, and unaccelerated flight

TIC Theil's inequality coefficient

T&E test and evaluation

TMP test management project

TPS Test Pilot School

UAV Unmanned Aerial Vehicle

US United States

USAF United States Air Force

USAF TPS United States Air Force Test Pilot School

USM United States Military

VISTA Variable Stability In-Flight Simulator and Test Aircraft

VSS Variable Stability System

VV&A verification, validation and accreditation

XML eXtensible Markup Language

APPLICATION AND VALIDATION OF OFF-NOMINAL AIRCRAFT PERFORMANCE MODELS

I. Introduction

This thesis documents research into prediction of off-nominal aircraft performance utilizing an open-source 6-degree of freedom (DoF) nonlinear simulator. This research includes real-world flight test integration for validation of the simulation results. The purpose of this research is to determine the degree of accuracy in which the simulation software predicts aircraft performance given a specific control failure. Control in this sense, pertains to any control surface (e.g. aileron, rudder, elevator) or thrust component (e.g. engine). For this thesis, failure modes involving the ailerons, rudder, elevators, and engine(s) were simulated and compared against real-world flight test data for validation. Better understanding of off-nominal performance leads to improvements in recovery training, flight control development, envelope limiting, trajectory planning and multistate aircraft design optimization.

1.1 Background

The primary goal of any initial design of an aircraft or new space system is to maintain operation in a state of optimum performance over a maximum operating envelope. The past has shown us that many large scale engineering endeavors have had to operate in an off-nominal condition for the majority of their designed life-span (e.g. Hubble Space Telescope, the Galileo spacecraft, or aircraft operating under a minimum equipment list (MEL)) [2]. Space systems tend to dominate the large-scale engineering endeavors that have had to overcome partial-system failures (due to their

inability to be serviced after launch), while most traditional aircraft have the ability to land and have a faulty component serviced or replaced prior to the next flight.

As aircraft are stretched into long-design cycles and operation well past their designed operational lifespan, the chance of component failure increases requiring increased time operating in an off-nominal state. For aircraft, this drives the requirement for a MEL, which provides pilots a list of equipment that is essential for safe flight of the aircraft (and also equipment that is not required for safe operation). Additionally, as aircraft are stretched to fly longer flights, there are fewer opportunities to provide regular service. As a result, aircraft designers must include performance robustness (i.e. the ability for the aircraft to overcome partial system failures and still continue the mission) into their design decisions to maximize the chance of mission accomplishment. Including performance robustness into an initial design has been studied on three fronts and will be discussed more in Chapter II. This research contributes to the front that accepts that component failures will occur given a large enough operational time-frame, and instead of improving reliability or adding redundant systems to compensate for that failure, let the designer predict off-nominal performance and then alter static design variables to increase performance robustness. It has been shown that this approach can increase overall mission expected performance while decreasing life-cycle costs [4].

Previous research has shown that incorporating partial system failures into the initial design process can improve performance robustness across a wide-range of failed states. One problem with designing for performance robustness is determining to what degree of accuracy can one model the aircraft in an off-nominal condition. With an accurate predictive capability, an iterative design approach can be accomplished to maximize performance given a wide range of off-nominal conditions (failed states). This research has direct applicability to the design of high-altitude, long-

endurance (HALE) platforms like the United States Air Force (USAF) RQ-4 Global Hawk, Facebook’s Aquila, Google’s Solara, and a plethora of vehicles currently in development for both military and civilian applications.

1.2 Research Problem Motivation and Description

As of May 2015, the combined service branches of the United States Military (USM) operated at least 678 drones of 18 different varieties [62]. The services identify a drone’s mission based on a tier system. Tiers 1 and 2 contain low and medium-altitude, long-endurance (MALE) platforms like the Gnat 750 and the MQ-9 Reaper, respectively. Tier 2+ contains HALE platforms like the RQ-4 Global Hawk and tier 3 contains low-observable HALE platforms. HALE platforms are designed to fly greater than 60,000 feet and have more than 24-hours of on-station time [59]. Of the operational USAF unmanned aerial vehicles (Unmanned Aerial Vehicle (UAV)s), the vehicles with the longest endurance capabilities, the MQ-1 and RQ-4, have a 36 hour maximum endurance capability [54]. Recent trends in the aerospace industry have included increasing the mission duration of these platforms and reducing the operating cost per flight-hour. Examples of these trends include several developmental ultra-long endurance UAVs (e.g. Aurora Flight Science’s Orion [66], Facebook’s Aquila [52], and Google’s Solara [6]).

The functionality and availability requirements of these platforms could not be satisfied with a single-point design methodology. As a result, these trends drive a requirement for a multistate design approach for aircraft requiring performance robustness across a wide-range of failed states, each requiring an accurate off-nominal aircraft model to predict performance and flying qualities (P&FQ).

1.3 Research Objectives and Scope

One of the main reasons that this field is lacking in exploration is the inability to safely induce a control surface failure while performing a test maneuver to compare with simulation results. The advent of digital fly-by-wire flight control systems, while not new, has allowed for the creation of variable stability aircraft, which can accomplish this control restriction within a safe operating envelope. These variable stability aircraft can alter their flight control laws real-time to 1) simulate handling qualities of other aircraft, 2) test a control law design, or in the case of this research, 3) limit the deflection of a flight control to study off-nominal performance. This research will utilize a high-fidelity nonlinear aircraft flight dynamics model coupled with a variable stability aircraft to determine to what degree of accuracy can these models predict off-nominal performance. The objectives for this research are as follows:

1. Compile and condense previously collected data from a system identification (ID) process on a test aircraft and build a high-fidelity simulation model of that aircraft.
2. Calibrate the simulation to match nominal performance data collected during flight test. Compare the accuracy of the simulator during nominal aircraft performance.
3. Induce off-nominal aircraft states and perform flight test maneuvers, collect aerodynamic and performance data, and compare with the performance predicted by the simulation software.
4. Determine the degree of accuracy with which the software predicted performance.
5. Determine if changes to the baseline model can or need to be made to increase

off-nominal accuracy.

Using *a priori* knowledge of the control inputs, how does one determine if that aircraft model is valid (or accurate)? This is the last phase of the system ID process, model validation. Morelli and Klein state, “the identified model must demonstrate that its parameters have physically reasonable values and acceptable accuracy, and that the model has good prediction capability on comparable maneuvers” [41]. This definition of **valid** is subjective and qualitative; a quantitative approach is used in this research to determine model validity.

1.4 Case Study: Twin-Engine Learjet 25D

One requirement of this research is the ability to validate the simulation results with real-world flight test data. This requirement, while providing credibility to the results, limits the validation mechanism to aircraft with variable stability flight control systems. Variable stability aircraft have the ability to alter their flight-controls and/or flight-control laws (mechanically or digitally) to study various flying and handling qualities. Extensive research has been conducted utilizing a variety of variable stability aircraft, with NASA being the primary government purveyor [30]. On the civilian side, the Calspan Corporation has vast experience working with IFS, aircraft with variable stability interfaces, and has been the primary innovator, developer and operator of these aircraft since 1947 [74].

The Calspan Corporation currently operates five IFS platforms, with the ability to vary much more than the stability of the aircraft [74]; of the five IFS platforms, two aircraft were available for this research. One of these was the F-16 Variable Stability In-Flight Simulator and Test Aircraft (VISTA) and the other was the Learjet 25D IFS. Since the main focus of this research applies to aircraft with long-endurance requirements, implying aircraft with higher aspect ratios, the Learjet 25D platform

was selected as the test platform.

The Learjet 25D is a twin-engine business jet, Figure 1, equipped with two General Electric CJ-610 turbine engines. The version tested is heavily modified with a fly-by-wire (FBW) flight control system, and is designed to take commands from either the pilot onboard or a remote operator on the ground [13]. The flight control laws can be altered in real-time to simulate the performance and handling characteristics of other aircraft, or in the case of this research, restrict the movement of specific control surfaces or the thrust of an engine. The Learjet IFS has a 4-DoF simulation capability, with fully programmable control over pitch, roll, yaw, and the x-axis (thrust).



Figure 1. Calspan Corporation Learjet 25D In-Flight Simulator [53].

1.5 6-DoF Simulation Aircraft Model

The simulation software used to model the Learjet was JSBSim. JSBSim is an open-source, nonlinear, six degree-of-freedom flight dynamics model, used for modeling flight dynamics and control of aircraft [8]. From a programmer’s perspective, JSBSim can be viewed as a collection of program code (mostly in C++ but also some

C routines), that loads a user-supplied eXtensible Markup Language (XML) aircraft configuration file containing aircraft parameters and then propagates a state vector given a history of control inputs. The open-source nature of JSBSim allows the user to run the simulation as either a “black-box” or to modify the source-code to suit a specific need. Further description of the modification and programming required for the simulation in this research is contained in Appendix A.

The Learjet aerodynamic parameters used in the analysis of this research are a product of many different sources. The stability and control derivative information was provided by the Calspan Corporation, collected from previous parameter identification missions and extracted at a single condition, 250 KIAS and 15,000 ft MSL [39]. Aircraft geometry and flight control limits were retrieved from the Learjet 25D Cockpit Reference Handbook [12] or through direct measurement. Further description of the test aircraft and aerodynamic parameters used in this research is contained in Appendix C.

1.6 Chapter Summary

Two major contributions to the field result from model validation in the off-nominal aircraft condition. First, models may be incorporated into high-fidelity visual simulators for aircrew training to control actuator failures. This ensures that if and when a failure occurs in flight, the aircrew’s expectation of aircraft performance matches that of the simulation experience. This effect can drastically improve safety and pilot decision-making ability during actual aircraft emergencies. Secondly, if a model can be proven valid for an off-nominal condition, then the aircraft static design variables can be adjusted to maximize off-nominal performance of the aircraft to complete its intended mission. The second effect involves the field of multistate aircraft design optimization for performance robustness.

Chapter I provided background information on the objectives and motivation for this research. Chapter II contains a review of the literature pertaining to multistate robust design, high-fidelity simulation and flight test integration. Chapter III contains the research methodology. Chapter IV presents the results, findings and analysis of the simulation results compared with the flight test results. This chapter also comments on the ability to meet the research objectives and any corrections that were made to the model after flight test to better match the simulation results. Chapter V provides an overview of the work performed, conclusions on the applicability and contributions of the research, and recommendations for future work.

Appendix A contains information regarding the simulation software, JSBSim. Appendix B describes the JSBSim C++ code modification. Appendix C contains the nominal aerodynamic parameters of the Learjet used in the simulation and the sources for each data point. Appendix D contains the Learjet 25D XML aircraft configuration code. Appendix E contains the Learjet 25D XML autopilot configuration code.

II. Literature Review

Chapter II presents the necessary framework and foundation for this research. The first section introduces the methods for optimizing aircraft design and performance. The second section covers the system ID process as it pertains to model validation and the research areas into off-nominal aircraft modeling. Thirdly, a review of parameter identification methods are discussed. The fourth section covers the research into the integration of simulation and flight test. Lastly, common data analysis techniques and simulation evaluation criteria are covered to improve understanding of the results and conclusions presented in Chapters IV and V.

2.1 Aircraft Design Optimization

As described, the objectives provided in Section 1.3 directly support the ultimate goal of improving our ability to accurately model behavior within multistate aircraft design optimization (defined in Section 2.1.1). Such optimization of aircraft design is often an iterative process of small changes in order to squeeze-out the last 10-15% of performance capability, without sacrificing mission requirements. This process is often accomplished after the aircraft has moved into the production phase. For example, many commercial airline companies are adding winglets to current aircraft to reduce induced drag, while having a minimal impact on overall weight [27]. Optimizing an aircraft's design prior to production involves prediction of the aircraft's performance through simulation and modeling. For aircraft requiring performance robustness in the wake of system or structural failures, a multistate approach is necessary. A review of relevant research has shown that design for aircraft robustness has been researched on three fronts. The first two areas involve optimizing control laws and component reliability. These two areas are applicable to the optimization of the post-production

aircraft. The third front involves treating an aircraft as a multistate system to optimize its static design variables before the system’s geometry and physical architecture have been set. This is the front most directly supported by this thesis, thus reliability improvements and control optimization will only be briefly introduced, mostly to provide context for the overall motivation. Greater detail, however, will be provided for multistate design optimization and multistate systems research (see 2.1.1).

First, reliability theory describes the probability of a system completing a specified task during an interval of time [61]. This theory utilizes failure rates of systems to predict the probability a system will complete a given mission or task. For a single component system, the probability that it will complete its function is simply the product of the failure rate and the time interval. A component’s performance threshold may be treated as binary (either as failed or operational) or there may be intermediate levels of performance. For example, a sewer system would have several intermediate performance levels as debris builds up on the inner walls, while a cable driven flight control may only be fully operational or failed (severed cable). Conventional improvements to reliability involve improving component reliability and improving system reliability through the incorporation of redundancy. Investigating the mode of failure of a single element and improving on the design to reduce the failure rate falls into the first category of reliability improvement methods. This method is heavily incorporated by the automotive industry, continually iterating of a component design with every successive yearly model release. Adding redundancy to a multistate system (MSS) can aid in performance robustness at the cost of extra weight and systems integration. This method for improving reliability is heavily used for space applications. For example, most satellite systems have multiple redundant electronic buses, communication relays, and backup power sources.

Second, a significant amount of research has been conducted in the field of control

design and optimization for performance robustness. This field of study, as applied to aircraft, utilizes on-board sensors or variances in performance to determine the presence of a failed component. Flight control laws can sometimes then be dynamically optimized or adjusted to compensate for a specific failure. Research into this field was not possible until the advent of FBW flight-control systems, which could be digitally reprogrammed in-flight. A detailed history of fault-tolerant and reconfigurable flight control studies is contained within [69].

2.1.1 Multistate Design Optimization

The third front in improving aircraft robustness involves multistate design optimization. Initial work on multistate design optimization began in the mid-1970s with Barlow and Wu [7], El-Neveih et al. [21], and Murchland [49], and Ross [64]. The work evolved from the need to improve system reliability by looking at a system as a collection of elements, each with varying levels of performance. An element is essentially a component of a larger system that is analyzed in its entirety, though the element may also be made of several parts or components. In an aircraft case, an aileron may be composed of several actuators, electrical relays, and hydraulic lines, but generally its reliability is analyzed as though it were a single element of a bigger system, with the system being the entire aircraft. The simplest form of a MSS is the binary case, where the MSS has two available states, perfectly functioning or completely failed.

Lisnianski and Levitin [42] describe a MSS as any system that is composed of different units having a cumulative effect on the performance of the entire system. Meaning the performance rates of each element composing the system can contribute to degradation of the entire system performance. Individual element performance degradation can be a result of fatigue, partial failures, exposure to unfavorable en-

vironmental conditions, etc. Each element can then be given a level of performance based on an expected state. In order to analyze the behavior of the larger system, one has to know the performance levels of each element. Mathematically, any system element j can have k_j different states (performance rates), and can be represented by Equation 1.

$$g_j = \{g_{j1}, g_{j2}, \dots, g_{jk_j}\} \quad (1)$$

The performance rate element j and state i is represented by g_{ji} . As a function of time, the performance rate of element j , $G_j(t)$, is a stochastic process, meaning for the MSS time interval, $[0, T]$, the performance rate is analyzed using statistical methods rather than an explicit value. The probability that a specific element is in a certain state is given by Equations 2 and 3:

$$p_j(t) = \{p_{j1}(t), p_{j2}(t), \dots, p_{jk_j}(t)\} \quad (2)$$

where each element has the probability

$$p_{ji}(t) = Pr\{G_j(t) = g_{ji}\} \quad (3)$$

The MSS system structure function can then be derived representing the performance of the entire MSS, Equation 4.

$$G(t) = \phi(G_1(t), \dots, G_n(t)) \quad (4)$$

This simple MSS model offers a way to compute system performance given the performance probabilities of each element. These methods are particularly useful for flow transmission systems, power generation, and data transmission system as shown

in [42].

For the MSS mathematical framework to be applied, each element needs to be *relevant* to the entire system and *coherent*. An element is *relevant* if some conditions of the entire system state depend on the state of the particular element. For the aircraft example, an aileron (element) failure would affect the roll capability of the entire aircraft (MSS), therefore being a relevant element. The state of the landing gear may not be relevant, assuming it failed in the up position, as it would not affect the aerodynamic properties of the aircraft. For a MSS to be *coherent*, the system requires: 1) A failure of all elements causes entire system failure, 2) all elements operational means the system is fully operational, 3) once the system has failed, no additional failure can cause the system to function again, and 4) no repair or addition of elements can cause system failure, while the system is operating [42]. In addition to the coherency requirements, MSS's can be categorized as homogeneous and non-homogeneous. A MSS is homogeneous if all of the system elements have the same number of states (with equal performance values). Using this definition, all binary systems would be homogeneous, with all elements being either *failed* or *operational*. Most complex systems, however, have a variety of elements each having individual levels of performance, yielding a non-homogeneous system.

Most past research (pre-2010) in multistate systems has been focused on ground-based parallel, series, or k-out-of-n systems [42]. Coherent structure theory evolved in the late 1950s through Moore and Shannon's work examining the reliability of relays within circuits [47]. These systems are typically the focus of mechanical or reliability engineering disciplines. Very few examples exist with applications of the MSS framework to the aerospace industry. However, with the addition of long-endurance requirements for several experimental platforms, the MSS methodology has been applied to the design of a long-endurance UAV by Agte, Borer and de Weck [4], where

the system elements were defined by the static design variables, control law variables, and operational parameters. The static design variables, x , represents the geometry of the aircraft (wing sweep, wing area, tail arm). The control law variables, u , represent variation to the system’s control law (i.e. control gains). The operational parameters, c , represent fixed variables that influence the behavior of the system (i.e. operating conditions). Using this info, system state performance, Equation 4 can be more explicitly represented by Equation 5:

$$G^s = \{G_1^s(x, u, c), G_2^s(x, u, c), \dots, G_j^s(x, u, c)\} \quad (5)$$

where G^s is a function of the state performance levels, rather than time. Treating the aircraft as a multistate system, resulted in an “improvement of 10-11% in system availability with minimal impacts on cost or performance [4],” when compared to literature baseline designs. Availability in this sense means the system (aircraft) has enough functionality to complete its intended mission. Agte et al. also showed this method useful in identifying components (elements) that may lead to aircraft (system) failure through performance analysis across the aircraft’s states [3]; this was accomplished using a twin-engine Beechcraft Super King Air coupled with a JSBSim aircraft model.

These advances in the field of multistate design have led to the requirement for high-fidelity off-nominal aircraft models. However, the advances rest on the assumption that the model correctly predicts aircraft performance with a wide variety of failed subsystems. This assumption limits the applicability of multistate design for aircraft robustness until it can be shown that a modeling method can correctly predict off-nominal performance. Verification and validation of the modeling method, as such, is the focus of thesis.

2.2 System ID and Model Verification

The process of observing how inputs lead to different outputs is known as system identification (ID) [41]. Applied to aircraft, system ID involves developing an accurate and comprehensive mathematical model of the system and validating the model with wind tunnel and/or flight test data. While wind tunnel testing can provide a significant amount of aerodynamic data about a particular aircraft, real-world flight test data is the most credible when developing and validating a model. This process involves the knowledge of the input, u , and the output, y , to solve for the system model (y can also be thought of as G in Equation 4). When the model is complete, it should be able to predict the outcome of future wind tunnel and flight tests with a high degree of accuracy. There are several motivations behind the development of high-fidelity models that match both wind tunnel and flight test. One motivation is flight simulators dependency on highly accurate models to train aircrew to operate the aircraft. Another motivation is the expansion of an aircraft's flight envelope or verifying that an aircraft meets a desired specification.

The process of identifying parameters necessary to create aircraft models has been underway since the late 1940s [46]. As a result, the field of system ID has been widely studied and developed. With current system ID methods, experiment designers can develop test plans that extract the maximum amount of useful data. The first step is determining how to collect measurements of the inputs and outputs of a system, and determining sources of error (or noise) in the data. These error sources must be addressed when comparing with the theoretical model of the system. The next step is determining what form the model should take and ensuring that each parameter gives meaningful information about input/output dependencies. The final step is identifying how good the model is and how to utilize the data collected from the model [41].

The majority of system ID processes involves estimating aerodynamic parameters for an aircraft in a nominal condition, i.e. all control inputs operational. These aerodynamic parameters (also known as stability and control derivatives) are then incorporated into a mathematical model of the aircraft, which then attempts to predict how an aircraft will perform given a known input. Most early work in the field involved linearizing the aircraft model about a known flight condition to ease the computational burden and simplify flight control law design. However, evolution of computational power over the past 20 years has allowed the retention of highly accurate nonlinear equations of motion. These high-fidelity models have proven, when properly calibrated, that nominal aircraft performance can be predicted to a high degree of accuracy, with very little computational cost. What is lacking in current research is validating the accuracy of these models for aircraft in an off-nominal condition. Several research fronts explore aircraft performance in off-nominal or adverse conditions. These conditions can include environmentally-induced events (shear or wake vortex events), stalls or departures, on-board system failures, and structural failures including battle damage. These research areas contribute to improvements in recovery training, flight control development, envelope limiting, trajectory planning and multistate aircraft design optimization. As performance robustness and safety becomes a more important priority for aircraft developers, the need for multiple sources of modeling data (experimental, theoretical, and computational) in these conditions has increased. Difficulties lie in combining these sources of data in the dynamic flight environment due to violation of the aerodynamic and physical symmetry assumptions. Additionally, challenges exist in validation of a designed model with real-world flight test due to technical and safety issues.

The research fronts can be divided into three main areas: high-angle-of-attack (including stalls and departures), environmentally-induced events (wind shear, wake

turbulence, adverse weather), and on-board system failures (including structural/-battle damage). A significant amount of research has been conducted in the high-angle-of-attack field. A good overview of high-AoA research can be found in [15] and [67]. This research has led to many advances in stall warning technologies, stall recovery maneuvers and training, and improvements in aerodynamic designs. Accurately modeling aircraft response to environmentally-induced events is also of significant interest to aircraft designers. This modeling process allows validation of aircraft design variables (structural tolerances, control robustness, operational limitations, etc.) when encountering adverse conditions. Abundant information is available on aircraft modeling in adverse environmental conditions. Some examples of adverse condition modeling include: low-level wind-shear [29], turbulence [44], and National Aeronautics and Space Administration (NASA)’s LEWICE (LEWIs ICE acceleration program) software used to predict aircraft performance during icing conditions [14]. This research has led to advances in aircraft stability augmentation systems and safety procedures for aircrew (e.g. increased takeoff/landing spacing for wake vortex avoidance, wind shear prediction and warnings from weather forecasters, icing tolerances, etc.). The last research area, and the focus of this thesis, involves modeling aircraft with failed systems or structural/battle damage.

2.2.1 Modeling Aircraft with Failed Systems/Battle Damage

The research herein focuses on failures or damage to the aircraft in flight. This can include on-board system failures, control surface failures, or actual airframe damage (due to battle damage or structural failure). Aircraft modeling in these off-nominal states has typically been limited to engine-out scenarios, control degradation (possibility due to hydraulic pressure loss), and sometimes hard-over/frozen flight controls. Aircraft are generally not modeled (for flying performance reasons) with structural

failures or battle damage. Modeling requirements for off-nominal cases are sometimes driven by certification requirements.

Certification requirements for the USAF are covered in the MIL-STD-1797B, *Flying Qualities of Piloted Aircraft*. For example, for engine-out controllability, the requirement is “on multi-engine aircraft it shall be possible to recover safely from stalls with the critical engine inoperative” [17]. Therefore, if sufficient thrust and control authority exists after an engine failure and stall, then the aircraft passes the requirement. Additional aircraft failures resulting in “degradation in flying qualities is permitted only if the probability of encountering a lower level [of failure] than specified is sufficiently small. The contractor shall determine, based on the most accurate available data, the probability of occurrence of each aircraft failure state per flight within the operational and service flight envelopes” [17, p. 44]. Therefore, the aircraft contractor is only required to show the probability of the failure occurring and level of effect it would have on the aircraft. Probable malfunctions are allowed ($< 10^{-3}$ per flight hour) as long as they have only a minor effect, and extremely improbable malfunctions ($< 10^{-9}$ per flight hour) need not be considered [17, p. 118]. This fact has limited the need to model off-nominal performance to a high degree of accuracy.

2.3 Aerodynamics of Aircraft with Failed Systems

Investigation into the off-nominal performance of an aircraft begins with the aerodynamics of the nominal aircraft. When modeling the aerodynamics of the nominal aircraft one must first decide what the mathematical form should be. A review of relevant research suggests that there are several methodologies to determine the forces and moments acting on an aircraft. The analytical methods include computational fluid dynamics (CFD), panel methods or strip theory, and USAF data compendium (DATCOM).

“CFD involves solving the partial differential equations governing the motion of air about the aircraft; panel methods or strip theory, which involves slicing the aircraft into sections that approximate two-dimensional airfoil sections, then summing up lift, drag, and pitch moments for each airfoil section to get the resultant forces and moments about the aircraft; and U. S. Air Force DATCOM, which is an extensive set of rules of thumb for aerodynamic dependencies, based on experience and airplane geometry” [41, p. 45].

For nominal, low-angle of attack scenarios, these methodologies work well to predict aircraft performance. However, the most insight can be gained by utilizing a non-dimensional physical model of the aircraft’s aerodynamic forces and moments, whose parameters are gathered from real-world flight test (or wind-tunnel data). This process is a balance between the complexity of the model and the meaningfulness of the individual parameters. An extremely complex model may lead to more accurate predictions in some flight regimes, but increases the number of parameters, making it difficult to understand the model inter-dependencies. An overly simplified model will only be valid for a single flight condition, and though the parameters are meaningful, they would not be able to predict performance outside that specific condition.

A typical aerodynamic model (extracted from the JSBSim flight manual [9] and also the one used as a baseline in this research) for an aircraft in a nominal state is shown in the equations below. The longitudinal force coefficients (lift, pitch and drag) are determined using Equations 6, 7, and 8:

$$C_N = C_{N_\alpha} \alpha + C_{N_{\delta_e}} \delta_e + C_{N_0} \quad (6)$$

$$C_m = C_{m_\alpha} \alpha + C_{m_q} \frac{q\bar{c}}{2V} + C_{m_{\delta_e}} \delta_e + C_{m_0} \quad (7)$$

$$C_D = C_{D_0} + KC_L^2 \quad (8)$$

where C_N is the normal force coefficient (normal force is equivalent to lift force at very small angles-of-attack), C_m is the pitch moment coefficient, and C_D is the drag

coefficient. Individual parameters include change in normal force coefficient with angle-of-attack (C_{N_α}), angle-of-attack (α), change in normal force coefficient with elevator deflection angle ($C_{N_{\delta_e}}$), elevator deflection angle (δ_e), normal force coefficient at zero angle-of-attack (C_{N_0}), change in pitch moment coefficient with angle-of-attack (C_{m_α}), change in pitch moment coefficient with pitch rate (C_{m_q}), pitch rate (q), mean chord length ($\bar{c}$), velocity (V), change in pitch moment coefficient with elevator deflection angle ($C_{m_{\delta_e}}$), pitch moment coefficient at zero control deflection (C_{m_0}), drag at zero lift (C_{D_0}), induced drag correction factor (K), and lift coefficient (C_L). The lateral coefficients (side force, roll, and yaw) are determined using Equations 9, 10, and 11:

$$C_Y = C_{Y_\beta}\beta + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r + C_{Y_0} \quad (9)$$

$$C_l = C_{l_\beta}\beta + C_{l_p}\frac{pb}{2V} + C_{l_r}\frac{rb}{2V} + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r + C_{l_0} \quad (10)$$

$$C_n = C_{n_\beta}\beta + C_{n_p}\frac{pb}{2V} + C_{n_r}\frac{rb}{2V} + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r + C_{n_0} \quad (11)$$

where C_Y is the side force coefficient, C_l is the rolling moment coefficient, and C_n is the yaw moment coefficient. The side force coefficient parameters include change in side force coefficient with sideslip angle (C_{Y_β}), sideslip angle (β), change in side force coefficient with aileron angle ($C_{Y_{\delta_a}}$), change in side force coefficient with rudder angle ($C_{Y_{\delta_r}}$), rudder deflection angle (δ_r), and side force coefficient at zero control deflection (C_{Y_0}). The rolling moment coefficient parameters include change in rolling moment coefficient with sideslip angle (C_{l_β}), change in rolling moment coefficient with roll rate (C_{l_p}), roll rate (p), wing span (b), change in rolling moment coefficient with yaw rate (C_{l_r}), yaw rate (r), change in rolling moment coefficient with aileron angle ($C_{l_{\delta_a}}$), change in rolling moment coefficient with rudder angle ($C_{l_{\delta_r}}$), and rolling moment coefficient at zero control deflection (C_{l_0}). The yaw moment coefficient parameters

include change in yaw moment coefficient with sideslip angle (C_{n_β}), change in yaw moment coefficient with roll rate (C_{n_p}), change in yaw moment coefficient with yaw rate (C_{n_r}), change in yaw moment coefficient with aileron angle ($C_{n_{\delta_a}}$), change in yaw moment coefficient with rudder angle ($C_{n_{\delta_r}}$), and yaw moment coefficient at zero control deflection (C_{n_0}). Further description of JSBSim’s aerodynamic coefficient build-up model is contained in Appendix A.

Very little information is published on adjustment and modification of these equations to model an aircraft in a damaged or off-nominal state. Most research into this area has involved aircraft structural or battle damage. Additionally, most battle damage research has been limited to answering questions like: Can the aircraft continue its mission? Can the aircraft land and return to combat? Will the aircraft be lost? Additionally, the infinite amount of scenarios that could result from battle damage (all of which are equally unlikely to occur) has tended to lead to model development *ex post facto* [23]. However, several researchers have investigated the effects of damage on aerodynamic parameters. Spearman [68] investigated damage effects on the aircraft’s static aerodynamic characteristics using wind-tunnel testing. Betzina and Brown [11] used a full scale McDonnell-Douglas A-4B subjected to actual gunfire to one wing to determine aerodynamic effects. Hayes [32] employed a wing section with removable panels in a wind-tunnel to simulate battle damage and its effect on the force and moment coefficients. These three researchers all utilized wind-tunnel data to determine if the aircraft was still flyable after suffering battle damage. Due to the safety issues of flight testing aircraft with battle damage, validation of these models is difficult. When looking at failed components rather than battle damage, most research has been limited to control design and optimization, as discussed earlier.

2.3.1 Parameter Estimation and Identification

Once a mathematical model is proposed for an aircraft system, a methodology must be developed to determine what values the coefficients (also known as parameters) in the equations will have. The mathematics and statistics behind parameter estimation and identification are well-developed. An early least-squares methodology for parameter estimation was presented by L. Shumacher in 1952 utilizing a B-25J aircraft [65]. Since the 1950s, the applications for parameter estimation have grown with hundreds of published articles in a variety of fields. An excellent summary of aircraft parameter estimation methods is contained within [41] and [72]. A brief overview of estimation theory and regression analysis is covered here to provide a basis for the estimation methodology discussed in Chapter III.

Parameter estimation is the process of estimating values of unknown system parameters based on noisy measurements. Since there is always a source of noise in the measurement, explicit values cannot be analytically determined with absolute certainty. Instead, the values of the parameters are represented using statistical terms. For example, a linear measured response, z , is a function of a vector of random (or constant) variables, θ , a noisy measurement, v , and unknown parameters, H , Equation 12 (nomenclature from [41]):

$$z = H\theta + v \tag{12}$$

In the nonlinear case, H is a function of θ , or $h(\theta)$ in Equation 13.

$$z = h(\theta) + v \tag{13}$$

In these cases, both θ and v can be represented using statistical terms depending on how much is known prior to the experiment. What is desired is the probability

density of a specific variable. The probability density of a variable is the likelihood that the variable will take on a specific value. The probability density is defined by the variable's mean (or average value over time) and its standard deviation (a measure of variation in the sample). In aircraft parameter estimation, the statistics (standard deviation and mean) of the noise are typically known *a priori*, or before the test. Three models are typically utilized depending on what is known about each variable. The Bayesian model assumes that θ is a collection of random variables with a probability density $p(\theta)$ and the noise, v , can be represented by the probability density $p(v)$. In aircraft parameter estimation, this is typically not the method of choice for it is difficult to make a statement about the probability density of the parameters. The Fisher model assumes that θ is a vector of unknown constant parameters and the noise, v , can be represented by the probability density $p(v)$. This model involves maximization of the likelihood function, known to statisticians as maximum likelihood estimation (MLE). MLE “involves minimizing the weighted least-squares difference between measured outputs and model outputs,” [41, p. 181]. Thirdly, the least-squares model, involves no probability statements for θ or v . The reasoning of the least-squares model is that the most accurate estimate comes from minimizing the weighted sum of the squared differences between the model and the measured outputs. Ultimately, depending on what is known about the noise and parameters beforehand will drive what model is chosen to estimate the parameters. Additionally, deciding what parameters are incorporated into θ will drive how accurate and predictive the model is. The goal is a balance between minimizing the number of parameters to ensure simplicity, while maximizing its predictive capability. This goal maximizes understanding of the relationship between the parameters. In an effort to reduce the model to its simplest form, regression analysis techniques were developed.

Regression analysis involves studying the relationships between the variables in a

parameter estimation process. An example of a regression relating the normal force coefficient, C_N , to angle-of-attack, α , and elevator deflection, δ_e , is shown below in Equation 14:

$$C_N = C_{N_\alpha}\alpha + C_{N_{\delta_e}}\delta_e + C_{N_0} + v_N \quad (14)$$

In this equation, the angle-of-attack and elevator deflection are known measured quantities at each test point. The normal force coefficient, C_N , in this proposed model is predicted to depend on only the parameters listed above. While there may be other parameters that affect the normal force coefficient (e.g. flap deflection), only the assumed influential variables are addressed with the addition of an unknown residual error, v_N . This model for the normal force coefficient is said to be a *linear-regression* because the output and parameters are linearly related. The goal for the experimenter is to determine the values of the parameters, C_{N_α} , $C_{N_{\delta_e}}$ and C_{N_0} , by adjusting the elevator deflection and angle-of-attack. Measurements of the normal force coefficient can be made indirectly through the use of several state measurements (e.g. weight, velocity, dynamic pressure, etc.). These state measurements are assumed to all contain noise, therefore a residual error is included in the linear regression. From this data, the parameters can be estimated to determine the best fit possible for the data, typically using a least-squares algorithm. Results can be either a validated model (assuming a good fit) or a new aerodynamic model (due to a poor fit). Additional detail on regression analysis techniques can be found in [20] and [50].

When estimating parameters from flight test data, the dependent variables are not always measured directly and subject to the errors in the state measurement. Additionally, the measurements of the control deflections include measurement error. This fact violates the assumption that the independent variables are measured without error, contributing to the bias of the least-squares solution. This problem can be

mitigated through meticulous handling of the data and isolating each equation using a scripted maneuver. These flight test maneuvers have been developed over the past 65 years, to determine very specific parameters of linear-regressions. The United States Air Force Test Pilot School (USAFTPS) has extensive experience with flight test maneuvers designed to extract parameters from a specific aircraft. Chapter III, contains specific descriptions of the test maneuvers and their desired outcome.

A review of relevant research into the field of system identification has revealed several software packages that are designed to estimate parameters based on wind-tunnel or flight test data. These packages include:

1. System Identification Programs for Aircraft (SIDPAC) is a collection of computer programs used to solve a variety of problems involving system identification. The development of SIDPAC occurred at NASA Langley Research Center over a period of 10 years, and uses published methodologies from Morelli and Klein's *Aircraft System Identification: Theory and Practice* [41]. The programs are written for use in the MATLAB workspace and can easily be adapted to suit special system identification needs. A good overview of each program is contained within [48].
2. MATLAB's System Identification Toolbox is a collection of MATLAB programs and Simulink blocks used to model linear and nonlinear systems and to identify parameters within the model using a variety of methods. A description of the software packages features and tools can be found at the MathWorks website [43].
3. Control Designer's Unified Interface (CONDUIT) and Comprehensive Identification from Frequency Responses (CIFER) are two widely used system identification and control design packages developed by Mark B. Tischler at the

University of Maryland [72].

The list above, while not exhaustive, represents common aircraft system identification packages in use today.

2.4 Flight Test and Simulation Integration

For the past 50 years, simulation has played an important role in aircraft design and flight testing. As an aircraft design tool, simulation has led to improvements in efficiency, safety, and drastic reductions in overall program costs. Because high-fidelity simulation can be very computationally intensive, it has grown in parallel to the advances in microchip industry, with the most dramatic advances happening over the past 20 years. As a result, simulation continues to play a larger role in initial aircraft design due to the reduction in computational cost and validation of the results with flight test.

Early simulation models evolved from the advent of the digital computer in the 1950s with the first USAF simulator built in 1958 to support the Bell X-2 aircraft [33]. Since then, commercial aircraft developers like Boeing have increasingly relied on modeling and simulation prior to flight test to drive costs down. Development of the 777 aircraft was the first major airline project that relied heavily on simulation during the whole design cycle. This led to a shortened flight test cycle, improved safety, and cost avoidances when compared to the 767 program. In the design of Boeing's 787 Dreamliner, Cray supercomputers spent over 800,000 hours modeling and simulating initial designs prior to flight testing. As a results, Boeing only had to physically build and test 11 wing designs versus the 77 wing designs it tested with its legacy 767 aircraft [60].

2.4.1 Types of Simulation

The type of simulator used in support of flight test typically falls into one of five categories. These categories are covered in NATO's Research and Technology Organization (RTO) report on *Simulation in Support of Flight Testing* [33] and include:

1. **Analytic** simulation is designed for use in a multitude of scenarios including performance characterization, operational limits, safety studies, test planning, and test prediction. These simulations are not considered "real-time" as there is no real-time control of an aircraft or system. A system is simply given a vector of control inputs over time and the states are calculated by the simulation software.
2. **Man-In-The-Loop (MITL)** simulators offer real-time support of flight dynamics testing. These simulators typically contain high-fidelity mock-ups of the aircraft's cockpit and visualizations and offer the operator a ground-based method to simulate flight of an aircraft.
3. **Hardware-In-The-Loop (HWIL)** simulators expand on MITL simulators by incorporating the actual aircraft's avionics and flight controls. These systems are useful in validating the reliability of the physical components when given actual pilot commands.
4. **Iron Bird** simulators consist of the entire mechanical and electrical makeup of the aircraft without the skin. While useful for a designer concerned with system integration, they have limited utility for exploration of flight dynamics.
5. **In-Flight Simulation** is the last and most useful type of simulation system. In-flight simulation allows a pilot to fly a simulated version of a new aircraft while computers ensure safety limits are not exceeded. This type of simulation

requires aircraft to be modified with a reprogrammable FBW flight control system to mimic the performance characteristics of any desired aircraft. These simulators are also useful in avionics testing, specifically radars, pitot-static systems, and navigation systems.

This thesis incorporates analytic, MITL and IFS methods to meet the research objectives. Analytic simulation is incorporated utilizing JSBSim, coupled with MATLAB’s Simulink. MITL simulation utilized was a proprietary model developed by the Calspan Corporation. Finally, in-flight simulation is used to validate the analytical and MITL results. Further discussion on the simulation methodology utilized in this research is contained in Chapter III.

2.5 Data Analysis

The last step in the system identification process is to verify and validate the model. In this step, the experimental data (wind tunnel or flight test) is compared with the data produced by the simulation model. When compared, assuming equal inputs, the error between measured and predicted performance should be zero mean and purely random in nature, “indicating that all the deterministic components in the measured output have been represented by the identified model” [41].

One of the problems with collecting real-world flight test or wind-tunnel data is the ubiquitous nature of noise. This noise can result from a sensor providing measurements or from external forces like wind or turbulence. When propagating a simulator model, the sources of error are controlled by the user with simulated perfect measurements and no exogenous inputs to the system. Therefore, a typical data analysis plan will attempt to filter or smooth this data to isolate the sensor noise, prior to comparison with the simulator data.

2.5.1 Filtering

A filtering method can be applied to the output signal from a measurement through the incorporation of a discrete transfer function. One concern with the implementation of a filter is that they contain their own dynamics and can alter the magnitude and phase of the incoming signal. This effect can be seen through the use of a frequency-domain plot (Bode plot). Since measurement noise is often at a much higher frequency than the data being studied, a *low-pass filter* can be incorporated to isolate only the low-frequency portion of the data and discard the noisy high-frequency data. This type of filter allows all frequencies up until its cutoff (Nyquist) frequency through for analysis while attenuating the frequencies above. An excellent description of low-pass filter design is contained within [55]. A filtering method can work well when investigating linear relationships with the aircraft model, but their application is limited when using nonlinear models. With linear models the input frequency matches the output frequency with an altered amplitude and phase angle. For nonlinear models, the amplitude and phase changes may be associated with a different input/output frequency relationship. For this reason, the filter method would not correctly isolate the measurement error from the data [41]. Another method to reduce the noise in the data is to utilize a smoothing algorithm post-test, which generally yields a better result over the filtering method.

2.5.2 Smoothing

Smoothing involves post-test alteration of the data to isolate errors from exogenous inputs (sensor or external noise). A review of relevant research shows that several methodologies have been developed. One method involves implementing a frequency-based filter in the time-domain. This process can be accomplished by designing a low-pass filter (discussed in the previous section) and then applying an inverse Fourier

transform to the filter. This procedure transforms the function from the frequency-domain to the time-domain. Another smoothing method is to apply a curve-fit to the data in the time domain. This is accomplished applying an assumption to the data. For example, if all the data points lie in a line but are noisy, then a linear relationship can be assumed, and a curve-fit would involve solving for the slope and y-axis intersect point. If the data points appeared to be in a parabolic shape, then a 2nd-order polynomial could be chosen to fit to the data. The coefficients of the polynomial fit are calculated via a least-squares process. Ultimately, it is the tester's choice what to choose for the fit of the data and an error should be calculated to determine how well the data fits the curve. This error can be presented as a root-mean-square (RMS) error, whose magnitude tells the reader how well the curve-fit represents the data (with a smaller value being a better fit). Another technique that has produced accurate results is using Fourier analysis to isolate the high frequency noise. One example of this practice is the Wiener filter [41], which minimizes the RMS error between the desired process and the estimated random process.

The smoothing and filtering of flight test data allows the isolation of the high-frequency noise and the revelation of the deterministic signal. These methods are commonly applied to provide a better comparison between the simulation output and real-world test data. Further details of the smoothing and filtering techniques utilized are covered in Chapter III.

2.5.3 Evaluation Criteria

The Department of Defense (DoD) has published several documents outlining the process of verifying and validating the output of a simulator model. For the USAF, this process, called validation, verification and accreditation (VV&A), is conducted by the test and evaluation (T&E) community. DoD document (DoDD) 5000.59 defines

verification as “the process of determining that a model implementation accurately represents the developer’s conceptual description and specifications,” and validation as “the process of determining the degree which a model is an accurate representation of the real world *from the perspective of the intended uses of the model*” [18]. Finally, accreditation is “the official certification that a model or simulation is acceptable for use for a specific application” [18]. The VV&A *Recommended Practices Guide* puts specific emphasis on validating the simulation based on its intended use;

“it is critical that the simulation be assessed in terms of how it will be used. Accurate knowledge of how the simulation will be used determines the degree of detail that must be represented for the simulation to provide useable results and the degree of correspondence with real-world phenomena that will be sufficient for you to use the simulation with confidence” [16].

This statement illustrates why most simulation models are limited in their testing of off-nominal performance. Those which do model off-nominal performance are typically limited to engine-out scenarios, icing, wind-shear events, and manual flight control reversions if and only if the probability of occurrence is significant (typically $> 10^{-9}$ per flight hour [17]).

When it comes to evaluation of nominal and off-nominal performance, the FAA has published quantitative validation requirements (also called objective tests) for both these categories in *Airplane Simulator Qualification AC 120-40B* [24] and more recently in *14 CFR 60 - Flight Simulation Training Device Initial and Continuing Qualification and Use* [25]. For example, for longitudinal trim in *14 CFR 60*, (cruise, landing and approach) the simulation compared to flight test data validation requirement is $\pm 1^\circ$ elevator deflection (δ_e), $\pm 1^\circ$ pitch angle (θ), and $\pm 5\%$ net thrust (δ_t). For the purposes of validating the simulation results versus the flight test data, this thesis uses the most current publication (as of 2012), *14 CFR 60* [25], as the primary

standard for validation. The specific metrics are discussed in the individual sections: nominal in Section 3.5.2 and off-nominal in Section 3.5.3.

Since many of the published metrics were not intended to be utilized for off-nominal aircraft states, Theil’s Inequality Coefficient (TIC) is provided as an additional measure of the similarities between the two time histories of data (simulation and flight test). TIC has been used in economics since the late 1950s to measure economic inequality among groups [71]. It has been recently been adapted in the aerospace industry as a robust control metric to analyze model uncertainty for both linear and nonlinear systems [37, 72]. In 2013, Dorbantu, Seiler and Balas [19] applied the TIC metric to a UAV simulation coupled with flight test and showed TIC to have “rigorous ties to robust control metrics, thus increasing confidence and validity in the technique.” This approach was utilized when comparing the off-nominal case studies (see Section 3.5.3).

TIC essentially measures the similarities between two time histories of data normalized from 0 to 1 [19]. If the value is near 0, then the data histories show strong similarities, indicating the model is very accurate. TIC values near 1, however, represent the largest deviation and very poor similarities. TIC values less than 0.25-0.30 have shown to result in accurate and validated models [37, 72]. TIC is represented by Equations 15 and 16:

$$TIC = \frac{\sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \tilde{x}_i)^T W (x_i - \tilde{x}_i)}}{\sqrt{\frac{1}{n} \sum_{i=1}^n x_i^T W x_i + \frac{1}{n} \sum_{i=1}^n \tilde{x}_i^T W \tilde{x}_i}} \quad (15)$$

$$W = \text{diag}[w_{x_1}^2, w_{x_2}^2, \dots, w_{x_m}^2] \quad (16)$$

where $\tilde{x}$ is the simulation time history, x is the flight test time history, n is the number of data points, m is the number of outputs compared. The weighting matrix, W , is a diagonal matrix of weighting factors for each time history. For the case of this

research an identity matrix was used for W because all measurements were assumed equally important.

2.6 Chapter Summary

This chapter covered the background information and a review of relevant research necessary to understand the methodology, results, and impact of this research. A review of research into aircraft modeling with failed subsystems showed that military standards for off-nominal aircraft states are described by the probability that they will occur and their effect on the controllability of the aircraft. This requirement has limited the need for high-fidelity off-nominal modeling. It was subsequently shown that for multistate design, adverse environmental conditions, and high angle-of-attack flight can all benefit from increased fidelity of off-nominal modeling. Background information and research into aerodynamic modeling was presented to show that the modeling equations are chosen by the user and can be altered to suit a specific scenario. A brief overview of system identification and parameter estimation practices was then presented to show how previous researchers determined values from the real-world data. A history of the integration of flight test with simulation was covered to validate the research methodology outlined in Chapter III. Finally, data analysis methodologies of smoothing and filtering were covered to show how previous researchers isolated the sources of error.

Taking advantage of previous research, the goal of this research is to analyze several “attainable” off-nominal states of the Learjet test aircraft and compare the performance using a high-fidelity flight dynamics model. The purpose is to attempt to quantify and validate the accuracy of the simulator versus true off-nominal performance in order to contribute the field of off-nominal aircraft modeling. The research methodology used to achieve these goals is contained in the next chapter.

III. Research Methodology

This chapter outlines the input/output structure of the simulation model, the limitations of the test aircraft, and the methodology used to achieve the primary objectives of this research. The primary goal of this research is to validate the model and the simulation software on its ability to predict off-nominal P&FQ of a known aircraft. Several nominal and off-nominal maneuvers are conducted in the simulator and in the test aircraft. Nominal flight data collected helps to refine the simulation model to match baseline performance of the aircraft. Once complete, off-nominal states are introduced and a performance maneuver conducted. Initial conditions for both simulator and aircraft are matched as accurately as possible prior to each test. This combination of modeling and flight test contributes to a methodology for validating an aircraft model for off-nominal cases.

3.1 Simulation Model State and Input Vectors

To illustrate how the simulation software is altered to include off-nominal states, an understanding of the basic inputs and outputs of the model is required. The baseline JSBSim simulation model accepts eight control inputs and outputs five state vectors. In this case, baseline indicates the C++ code is unmodified. The output state vectors are labeled: *state*, *state derivative*, *calculated*, *aerodynamic*, *control*, and *propulsion*. Each of these vectors contains useful properties for research though only the state ($\bar{x}$), state derivative ($\dot{\bar{x}}$), calculated output ($\bar{y}_{calc}$) and control input vectors are utilized. Equation 17 represents the twelve parameter state vector:

$$\bar{x} = [u, v, w, p, q, r, h, longitude, latitude, \phi, \theta, \psi]^T \quad (17)$$

with aircraft body-axis velocities (u , v , and w), roll, pitch, and yaw rates (p , q , and

r), altitude (h), latitude and longitude, and roll, pitch, and yaw angles (ϕ , θ , ψ). Equation 18 represents the state derivative vector:

$$\dot{\mathbf{x}} = [\dot{u}, \dot{v}, \dot{w}, \dot{p}, \dot{q}, \dot{r}]^T \quad (18)$$

with aircraft body-axis accelerations ($\dot{u}$, $\dot{v}$, and $\dot{w}$), and roll, pitch, and yaw angular accelerations ($\dot{p}$, $\dot{q}$, and $\dot{r}$). Equation 19 represents the calculated output vector:

$$\bar{\mathbf{y}}_{calc} = [N_z, \alpha, \dot{\alpha}, \beta, \dot{\beta}, V_{C1}, V_{C2}, V_T, V_G, M, V_{climb}]^T \quad (19)$$

with g-force (N_z), angle-of-attack (α), angle-of-attack rate ($\dot{\alpha}$), angle of sideslip (β), angle of sideslip rate ($\dot{\beta}$), calibrated airspeed (V_{C1} and V_{C2} , in fps and knots indicated airspeed (KIAS)), true airspeed (V_T), ground airspeed (V_G), Mach number (M), and climb rate (V_{climb}). The control input vector contains the normalized control surface positions. In addition to the output vectors listed, up to 288 native parameters can be outputted to a comma-separated value (CSV) file. These native parameters can be imported into MATLAB or Microsoft Excel for analysis. A full list of the native parameters is contained within [8, p. 141-144].

The simulation model used in this research is compiled as a MATLAB S-Function, which is a computer language description of a Simulink block [35]. The original JSBSim S-function utilizes eight controls to describe the inputs to the aircraft. The aircraft's control input signals are normalized about minimum and maximum deflection, ranging from 0 to 1. Equation 20 represents the original control input vector:

$$\bar{\mathbf{u}} = [\delta_t, \delta_a, \delta_e, \delta_r, mixture, setrunning, flaps, gear]^T \quad (20)$$

with throttle command (δ_t), aileron command (δ_a), elevator command (δ_e), rudder command (δ_r), engine mixture setting (not utilized with turbojet engines), set-

running (starts the engines), flap position, and gear position. Acceptable throttle commands range between 0 and 100%, while aileron, elevator and rudder commands accept -100% to 100% deflection.

The JSBSim S-function C++ code was modified to incorporate additional input channels required for this research. First, the original throttle channel (δ_t) was split into two throttle channels (δ_{tl} and δ_{tr}) to allow independent actuation of the engines. A series of autopilot set-points, switches, and proportional integral derivative (PID) gains were also integrated to execute a performance maneuver. The final S-function incorporates 43 input channels as shown in Appendix A, Figure A.2. A full description and tutorial of the C++ code modification is contained within Appendix B.

3.2 Aircraft Capabilities and Limitations

Within the aircraft, several desired modes are not attainable due to the aircraft's physical limitations as well as safety restrictions. When flying from the left seat, the pilot controls the baseline Learjet's fully reversible flight control system (Figure 2). When flying from the right seat, the pilot controls an irreversible (hydraulically actuated) flight control system that connects to the normal flight control cables. Digital input signals from the right seat pilot are fed from the rudder pedals and control stick to the Variable Stability System (VSS) computer. These signals are then fed to the hydraulic actuators, and can be used to simulate various flying qualities.

The primary flight controls consist of the ailerons, elevators and a rudder. The secondary flight controls are the flaps and the spoilers. Tertiary flight controls are aileron, elevator, and rudder trim tabs, and the aileron balance tabs. Additionally, an all-moving horizontal stabilizer is actuated using the Learjet's AUTO-TRIM system. This system was designed to prevent an overstress of the tail by relieving excessive hinge moments. The primary flight control surfaces are mechanically driven by direct



Figure 2. Learjet VSS Cockpit [53]

cable connections to the cockpit controls. As a result, ailerons always move inversely to one another (as one aileron moves up the other moves down), this effect provides a symmetric rolling moment (see Figure 3). The elevators are also connected by a mechanical linkage, which ensures symmetric deflection when commanding a change in pitch (see Figure 4). There were no restrictions to the rudder system (see Figure 5). Figures 3, 4, and 5 show the reversible Learjet 25 flight control system. As a result, aileron and elevator failures were limited to symmetric deflections.

The Learjet's VSS software interface can electrically control the inputs to the throttle, and primary flight control surfaces. Secondary and tertiary flight controls are not control by the VSS interface, but can be utilized to replicate off-nominal conditions. The flap positions are always symmetric at 8° , 20° , or 40° . The spoiler system can be either fully extended or fully retracted but cannot be utilized when the flaps are extended. The aileron balance tab system is used to decrease control column force required but is not pilot controllable.

While some flight controls are limited to symmetric deflections, the VSS interface

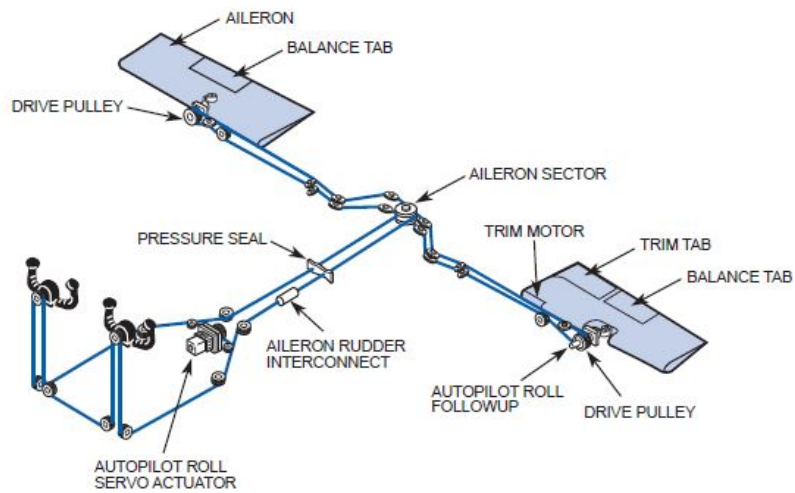


Figure 3. Learjet Aileron System [12]

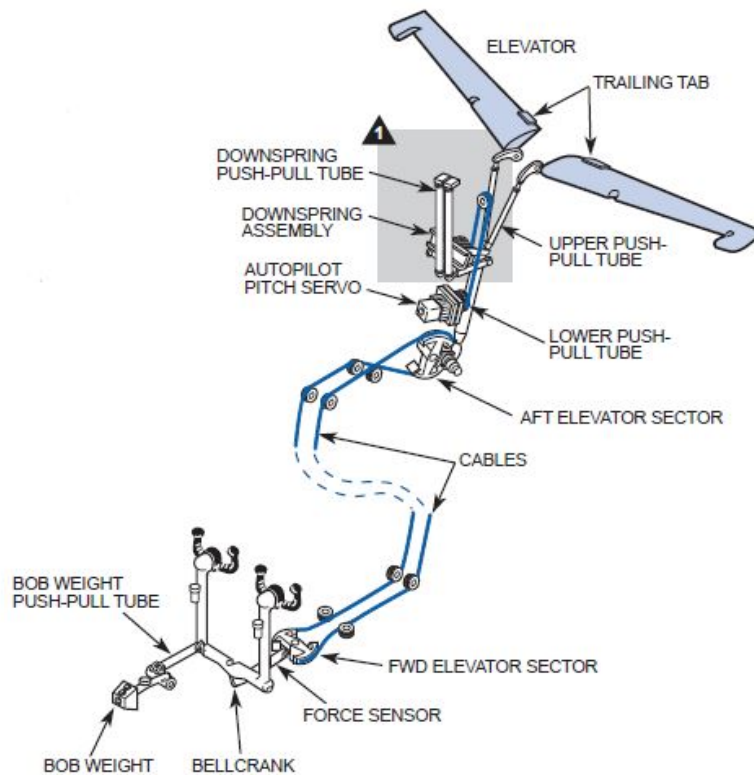


Figure 4. Learjet Elevator System [12]

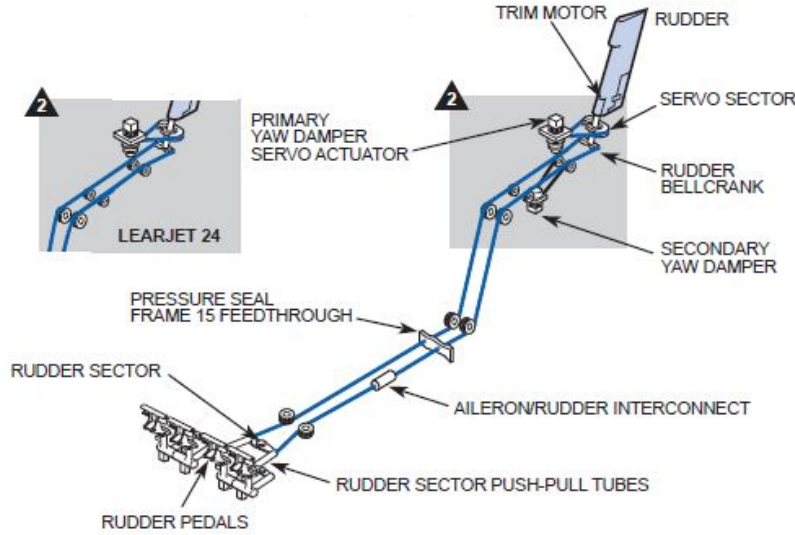


Figure 5. Learjet Rudder System [12]

has several unique capabilities that can be utilized to mimic failure modes. Control surface failures can be simulated by zeroing out the pilot's commands to the surface actuators inside of the VSS computer. Partial failures can be simulated by the reduction of gains on the input signal, i.e. reduced roll authority due to a single aileron failure can be simulated by reducing the pilot gains. This is accomplished through the VSS computer interface, not depicted in Figure 3. An engine failure can be simulated by moving one engine to in-flight idle [45]. Additional off-nominal VSS capabilities included control biases and runway trim events [40]. These restrictions and capabilities ultimately drove the design of the experiment outlined in Section 3.4.

3.3 Post-flight Calculations

In order to match initial conditions of the simulation model and each flight test maneuver, several variables need to be calculated. These variables (center of gravity, moment and product of inertia, etc.) are based on variations in aircraft configuration and fuel distribution at the time of the test. Additionally, a correction factor

is required to the simulation model to compensate for the effects of non-standard temperature.

3.3.1 Variations in Aircraft Center of Gravity

Since the force generated by each control surface generates a moment about the aircraft's center of gravity, the location is of particular importance to accurate aerodynamic modeling. While shifts in the center of gravity (c.g.) are not large, variations can occur due to changes in the weight distribution in the aircraft. The aircraft's center of gravity is dependent on the distribution of mass within the aircraft (number and location of passengers) and the amount and how the fuel is distributed in the various tanks. The reference datum for the aircraft's center of gravity is extracted from the Learjet 25 Type Certificate Data Sheet (TCDS) [26]. The x-axis originates 100 inches in front of the nose of the aircraft through the center of the fuselage. The positive y-axis points toward the right wing, and the positive z-axis points toward downwards.

Center of gravity coordinates are represented by x_{cg} , y_{cg} , and z_{cg} . The x_{cg} position (in inches) is calculated based on fuel and passenger distribution using Equations 21, 22 and 23:

$$W = Z_{FW} + W_{fuel} \quad (21)$$

$$M = 1000 \cdot M_{ZF} + 385.3 \cdot W_{wing} + 386.44 \cdot W_{tip} + 439.5 \cdot W_{fuse} \quad (22)$$

$$x_{cg} = \frac{M}{W} \quad (23)$$

where W is the weight of the aircraft and is a sum of the zero fuel weight (Z_{FW}) and the weight of the fuel onboard (W_{fuel}). The mass distribution, M , is calculated using the zero-fuel mass (M_{ZF}), the fuel weight in the wing (W_{wing}), the fuel weight

in the tip tanks (W_{tip}) and the fuel weight in the fuselage (W_{fuse}). The y_{cg} and z_{cg} are unaffected by fuel distribution and are 0 inches and 23.63 inches, respectively. These formulas were determined by the Calspan Corporation from empirical data and only applies to the specific Learjet aircraft used for this research [45].

3.3.2 Mass Moment and Product of Inertia

The aircraft's moments and products of inertia have a significant impact how aerodynamic forces affect the translation and rotation of the aircraft during flight. These variables are not static variables but change with respect to the aircraft weight distribution and configuration. As a result, variations of the aircraft moments of inertia (I_{xx} , I_{yy} , I_{zz} , and product of inertia I_{xz}) were calculated prior to each test maneuver and updated for each simulation run. The major aircraft structural components were approximated using basic geometric shapes with estimated densities to form I_{cg} . For individual components (avionics, instrumentation, furnishings, passengers), weight and balance records were consulted and components were treated as point masses. Fuel distribution is the largest contributor to variations of moment and product of inertia. Once the fuel state is collected, the aircraft's moment and product of inertia are calculated using the parallel-axis theorem in Equations 24, 25, 26, and 27.

$$I_{xx} = \sum_{i=1}^n (I_{CGx_i} + m_i X_i^2) \quad (24)$$

$$I_{yy} = \sum_{i=1}^n (I_{CGy_i} + m_i Y_i^2) \quad (25)$$

$$I_{zz} = \sum_{i=1}^n (I_{CGz_i} + m_i Z_i^2) \quad (26)$$

$$I_{xz} = \sum_{i=1}^n X_i Z_i m_i \quad (27)$$

where X_i represents the distance from the center of gravity along the x-axis, n is the number of components, and m_i represents the mass of component i . I_{CGx_i} , I_{CGy_i} , and I_{CGz_i} represent the zero-fuel weight moments of inertia. Products of inertia, I_{xy} and I_{yz} , were assumed zero due to aircraft symmetry. The values of center of gravity and moment of inertia were matched when running comparisons between simulation and flight test.

3.3.3 Non-standard Temperature Effects

The simulation model runs at standard day temperature and pressure conditions at 15,000 ft mean sea level (MSL), while test day conditions vary based on a variety of environmental conditions. A non-standard temperature affects the density of the surrounding air, which in turn affects the engine performance and control effectiveness. To adjust the test day conditions to the comparable density altitude used in the simulation model, the test day temperature (T_{test}) was divided by the standard sea level temperature (T_{SL}) as in Equation 28. The correction method is extracted from [22].

$$\theta_T = \frac{T_{test}}{T_{SL}} \quad (28)$$

where θ_T is the temperature ratio. The pressure ratio (δ_P) is then calculated using the test pressure altitude ($H_c = 15,000$ ft PA) using Equation 29.

$$\delta_P = (1 - 6,87559e6 * H_c)^{5.25590} \quad (29)$$

The density ratio σ_D is then calculated using Equation 30.

$$\sigma_D = \frac{\delta_P}{\theta_T} \quad (30)$$

Finally, the equivalent density can be calculated using Equation 31 below.

$$\sigma = 0.0023769 * \sigma_D \quad (31)$$

Using this equivalent density (σ) the simulation software can be executed at the altitude in which the test day density equates to.

3.4 Experiment Design

In previous sections, limitations and methods to match initial conditions of the simulation and flight test are discussed. This section covers what quantities are measured, how the quantities are measured, and how the maneuvers are conducted to excite the system.

As discussed in Section 3.1, there are no limitations on data acquisition or failure modes integration using the simulation software. The aircraft, however, is subject to several practical constraints. These constraints contain sensor resolution, sampling rate, input/output amplitude limitations, time available for test maneuvers, and aircraft physical limitations (including rate saturation). The Learjet test aircraft contained an advanced instrumentation and data acquisition system (DAS) that captures aircraft parameters at 200 Hertz. The directly measured quantities were velocity (V), angle-of-attack (α), angle-of-sideslip (β), body-axis Euler angles (ϕ, θ, ψ), angular velocities (p, q, r), translational accelerations ($\dot{u}, \dot{v}, \dot{w}$), and angular accelerations ($\dot{p}, \dot{q}, \dot{r}$). Additional measurements were aileron, elevator, horizontal stabilizer, and rudder positions through potentiometers, altitude through an altimeter, and temperature via a thermometer. Thrust for each engine is estimated using the aircraft's exhaust pressure ratio (EPR), as depicted in Equation 32:

$$T = \Delta P * A_e = (P_e - P_i) * A_e \quad (32)$$

where T is the thrust in pounds-force (lbf), P_e is exhaust pressure (psi), P_i is inlet pressure (psi) and A_e is nozzle exit area (in^2). Since the thrust signal is based upon measured engine pressures, a large amount of high-frequency noise exists. Filtering of the thrust signal is described in Chapter IV.

One of the problems with parameter estimation of modern aircraft is that many of these aircraft already incorporate some sort of feedback control mechanism based on some inherent instability. For most fighter-type aircraft, a stability augmentation system (SAS) is incorporated to improve maneuverability. For transport type aircraft, the use of yaw dampers is common to reduce the tendency for Dutch roll. When attempting to match a model to an aircraft, specific maneuvers are designed to excite the natural modes of the bare airframe and not the augmented (closed-loop) airframe. As a result, limitations on the gains of the specific feedback controller need to be imposed to accurately estimate the response of the aircraft. The Learjet test aircraft uses a dual yaw damper system to control transient yaw motion and to increase yaw stability. This yaw damper system can be disabled through switches on the autopilot control panel [12]. To maintain bare airframe dynamics the yaw damper is disabled for the test maneuvers.

The maneuvers for this research did not require a specific altitude or airspeed. Therefore, test maneuvers were intended to be executed in the heart of the Learjet's flight envelope, 250 knots and 15,000 feet. This airspeed and altitude provided the best trade-off between maneuverability and fuel consumption for the tests. Additionally, stability and control derivatives were available for this test condition based on previous parameter identification missions collected by the Calspan Corporation.

3.4.1 Simulation Control Integration

This research requires execution of a pre-planned performance maneuver with a high-degree of stability in order to compare to flight test data. As a result, several autopilot controllers are incorporated into the simulation model. The autopilots are used to control various parameters pertaining to the flight of the aircraft. For example, the bank angle hold autopilot takes a desired bank set-point and subtracts it from the measured bank angle. This summation is the error signal that feeds into the bank angle controller. A response is calculated, based on the gains of the controller and routed to the ailerons to command a roll. When the error signal is zero (desired bank angle equals actual bank angle), the command to the ailerons is zero and the aircraft is at the desired bank set-point.

One of the most common controller-types used do to their wide range of operating conditions and applications are PID controllers. The basic idea of the PID controller is to take an error signal (as described above) and sum a proportional, integral and derivative response to that error signal. The proportional response determines the ratio of error to output response. For example, an error signal of magnitude 10 and a proportional gain (K_p) of 2 would create a proportional response of 20. In general, the proportional gain speeds up the control system response, at the cost of oscillations. The integral response portion of the controller sums the error term over time and its effect is to gradually drive the steady-state error to zero, with the integral gain represented by K_i . The derivative response is proportional to the rate of change of the error value. A larger derivative gain (K_d) will cause the system to react stronger to changes in the error term [36]. Further description of PID control design can be found in [55, p. 592] and [70, p. 601].

One popular method of tuning the gains of PID controllers is the Ziegler-Nichols method. This method, determines a critical gain (K_c) and period of oscillation (P_c)

by setting the integral and derivative gains to zero and determining when the response starts to oscillate. Once the critical gain and period of oscillation are determined, K_d is set to 60% of K_c , K_i is set to 50% of P_c , and K_d is set to $\frac{1}{8}P_c$ [36]. Another method to tune the autopilot gains it is set up a cost function designed to minimize a desired path deviation, as in Equation 33:

$$\min J = \int_{t_1}^{t_f} (r(t) - r_{des}(t))^2 dt \quad (33)$$

where t_1 is an arbitrary time after control implementation, t_f is the final time, $r(t)$ is the measured parameter, and $r_{des}(t)$ is the desired location of the parameter. Utilizing the PID gains (K_p , K_i , and K_d) as the design variables, an optimization routine (like MATLAB's FMINCON) can be utilized to find gains to minimize the deviation from the desired path. For example, an autopilot switch is turned on at 10 seconds into the simulation with a desired bank angle of 30° ($r_{des}(t)$) starting from an initial bank angle of 0° ($r(t)$). It is desired that the aircraft reach the desired bank angle and stabilize two seconds after control implementation, therefore t_1 is set to 12 seconds. An optimization routine can then iterate on the values of the PID gains to minimize the deviation from the desired bank angle. Table 1 contains a list of the autopilots used in this research and their associated gains.

Within the simulation software, JSBSim, PID control is implemented in a separate autopilot file tied to the aircraft file using an XML script, more detail can be found in [9, p. 68-69]. The autopilot XML configuration file is contained in Appendix E.

3.4.2 Planned Test Maneuvers

To support the objectives of this research in a clear, concise and methodical way, a test point matrix is provided. Each objective is divided into several individual test points, which, if completed, are designed to answer the question the objectives

Table 1. Table of PID Autopilots

Control Parameter	Control Surface	K_p	K_i	K_d
Airspeed (V_c)	Throttle (δ_t)	4.5	0.003	8.175
Rate of Climb (V_{climb})	Elevator (δ_e)	0.01	0.0055	0.005
Roll Angle (ϕ)	Aileron (δ_a)	6.654	0.09142	4.11
Roll Angle (ϕ)	Rudder (δ_r)	0.07	0.02	0.001
Roll Angle (ϕ)	Left/Right Throttle (δ_{tl} or δ_{tr})	1.5	0.04	0.001
Sideslip Angle (β)	Rudder (δ_r)	2.534	2.269	0.602
Sideslip Angle (β)	Throttle (δ_t)	0.15	0.005	0.001

pose. Task, configuration information, and maneuver parameters are included. Flight test objectives are designed to support the research objectives by 1) measuring the nominal performance of the aircraft, and 2) by measuring off-nominal performance of the aircraft. The flight test objectives were thus divided into two phases (listed below). These objectives match those listed in Table 2:

1. Nominal flight test
2. Off-nominal flight test

To ensure that transient effects (turbulence, bad trims, etc.) which may affect the output data, are minimized, maneuvers are designed to be repeated. At least two repetitions are required but four or five are desired.

3.5 Methodology

The subsequent sections describes the tasks outlined by Table 2, how each is conducted in simulation and flight test, and what evaluation criteria is used to compare the solutions.

Table 2. Test Point Matrix

Obj.	Task	Failure(s)	Velocity (KIAS)	Altitude (ft)
1	Nominal Trim	None	200-300	15,000
2	Off-Nominal Trim	Single Engine	200-300	15,000
1	Short Period Dynamics	None	250	15,000
1	Phugoid Response	None	250	15,000
1	Roll Response	None	250	15,000
1	Dutch Roll Dynamics	None	250	15,000
1	Spiral Stability	None	250	15,000
1	Performance Maneuver	None	250	15,000-16,000
2	Performance Maneuver	Rudder	250	15,000-16,000
2	Performance Maneuver	Ailerons	250	15,000-16,000
2	Performance Maneuver	Single Engine	250	15,000-16,000
2	Performance Maneuver	Single Engine and Rudder	250	15,000-16,000
2	Performance Maneuver	Single Engine and Ailerons	250	15,000-16,000
2	Performance Maneuver	Ailerons and Rudder	250	15,000-16,000
2	Performance Maneuver	Elevator	250	15,000-16,000

3.5.1 Trim Method

The trimmed state, otherwise known as the aircraft’s steady-state level flight condition, is significant as the baseline for design and analysis of an aircraft. In this analysis, the trim conditions are used to validate the nominal prediction capability of the simulation model and as an initial state for each maneuver conducted.

Simulation. Several trimming strategies have been proven utilizing JSBSim, and validated with wind-tunnel data. In 2011, Goppert, Hwang and De Marco [31] showed a comparison between interval (the baseline JSBSim trim algorithm) and a Nelder-Mead simplex method [51]. The Nelder-Mead method was proven superior in

all cases, thus was the method chosen in this research to seek the trimmed solution.

The solution to the trim problem is not trivial. Most textbooks simplify the solution to the trim problem by utilizing linear models for longitudinal or lateral equations of motion. In the case of the JSBSim S-function used in this research, all nonlinear equations are retained within the S-Function and cannot be solved analytically. As a result, a numerical approach is required to minimize a cost function for a trimmed aircraft condition. Equation 34 describes the optimization routine utilized in the simulation to find a state input vector that minimizes the translational and rotational accelerations. Ideally, this cost function should vanish to zero at trim equilibrium conditions.

$$\min J = \dot{u}^2 + \dot{v}^2 + \dot{w}^2 + \dot{p}^2 + \dot{q}^2 + \dot{r}^2 \quad (34)$$

subject to system dynamics:

$$\dot{x} = f(\bar{x}, \bar{u}) \quad (35)$$

initial conditions:

$$t_0 \text{ and } x_0 = [u_0, v_0, w_0, p_0, q_0, r_0, h_0, long_0, lat_0, \phi_0, \theta_0, \psi_0]^T$$

and final conditions:

$$t_f$$

For the trim conditions, the desire is to minimize the cost function such that the system dynamics are (a stationary point):

$$0 = f(\bar{x}_{eq}, \bar{u}_{eq}) \quad (36)$$

For the trim problem with the Learjet, the design vector was limited to 5 dimen-

sions: δ_t , δ_a , δ_e , δ_r , and θ_0 .

For the case of the Learjet, stability and control derivatives are extracted at a known flight condition, 250 knots (421.95 feet/sec) and 15,000 ft MSL, therefore the minimization algorithm (MATLAB's FMINCON) is setup to seek the control and initial states that minimize the cost function at this flight condition. The trim problem seeks the minimum of a final cost (rather than a running cost) at final time and only the initial control input and states are in the design space. The control inputs are fixed at the start time and only the states are allowed to vary. The states are then compared at the final time and a cost is computed to determine how much deviation from the desired airspeed and altitude was induced based on the choice of the design vector. This is equivalent to the pilot trimming control pressures to zero and setting a "best guess" for throttle position to ensure minimal airspeed change. This same approach can be conducted with the trim problem. A small initial pitch angle (θ_0) is set to minimize altitude loss. For the control inputs, throttle is set to mid-range (50%), a small negative deflection of the elevator is selected to provide a initial pitch up of the aircraft (-15%), and aileron and rudder are set to zero deflection to prevent roll and yaw motion.

To reduce computational burden another successful way to trim a simulation model is described by Equation 37:

$$J = \sum_{i=1}^n \frac{(V_i - V_{des})^2}{V_{des}} + \frac{(H_i - H_{des})^2}{H_{des}} \quad (37)$$

where i is the index, n is the number of samples, V_i is the aircraft's velocity, V_{des} is the desired velocity, H_i is the aircraft's altitude, H_{des} is the desired altitude. This method works well for the nominal condition and sampling (i) only at the middle and final time. This is similar to how an aircraft pilot would trim an aircraft by observing the change in airspeed and altitude over a 10-15 second period. For the nominal

trim case, aileron and rudder deflections can be set to zero deflection (assuming aerodynamic and thrust symmetry). Therefore, the algorithm needs to only iterate on initial conditions for initial pitch (θ_0), elevator deflection (δ_e), and throttle percent (δ_t).

For the off-nominal case (single engine failed) a significant yaw moment exists causing sideslip and roll into the failed engine. To counteract the yaw and roll moment, a non-zero aileron and rudder deflection is required, thus the algorithm needs to iterate on aileron (δ_a) and rudder (δ_r) deflection. To minimize the heading drift, a third cost function (Equation 38) is used:

$$J = \sum_{i=1}^n \frac{(V_i - V_{des})^2}{V_{des}} + \frac{(H_i - H_{des})^2}{H_{des}} + \frac{(\psi_i - \psi_{des})^2}{\psi_{des}} \quad (38)$$

where ψ_i is heading, and ψ_{des} is desired heading. The desired heading should not be set at north for the optimization will iterate between 360 and 0 degrees, which is exactly the same. While Equation 34 represents the true trimmed solution, Equations 37 and 38 can help reduce computational burden.

Flight. Prior to each flight test maneuver, and during the off-nominal cases a “trim-shot” is conducted to determine the inputs required to hold the aircraft at steady, level, unaccelerated flight. A “trim-shot” is a flight test maneuver in which the pilot sets a pitch and throttle setting to hold an altitude, airspeed, and heading while trimming off excess control stick forces in the roll, pitch and yaw axes. Once the pilot feels that the control stick forces are zeroized, he or she simple lets go of the controls for a predetermined amount of time. At the end of the time-frame (typically 10-15 seconds) deviations from altitude and airspeed are recorded. If deviations are large, the the maneuver is repeated. In flight, the Learjet’s Data Acquisition System (DAS) captures aircraft parameters for comparison with simulation.

Evaluation Criteria. The evaluation criteria for determining the validity of the model in the trimmed cruise condition is sourced from the FAA’s *14 CFR 60 - Flight Simulation Training Device Initial and Continuing Qualification and Use* [25] and is shown in Table 3.

Table 3. Longitudinal Trim Metric [25, p. 55]

Parameter	Tolerance
Pitch Control (δ_e)	$\pm 1^\circ$
Pitch (θ)	$\pm 1^\circ$
Net Thrust	$\pm 5\%$

Of particular interest in multi-engine platforms is the aircraft’s performance with one or multiple engines inoperative. Much of the modeling and simulation done with failed engines has focused on the critical phases of flight (e.g. one engine inoperative abort, takeoff, missed approach, and landing). The FAA’s *14 CFR 60* contains metrics for evaluating the accuracy of these scenarios including the allowable one engine operational trim characteristics, Table 4.

Table 4. Engine Inoperative Trim [25, p. 58]

Parameter	Tolerance
Rudder Angle (δ_r)	$\pm 1^\circ$
Sideslip Angle (β)	$\pm 2^\circ$

3.5.2 Nominal Open-Loop Flying Qualities

Several OLFQ of the Learjet 25 are collected to determine the nominal predictive ability of the simulation software. These include short period dynamics, roll response, phugoid dynamics, Dutch roll dynamics, and spiral stability. Simulation and flight

test methodology for each are discussed in the subsequent sections, including evaluation criteria for each parameter.

3.5.2.1 Short Period Dynamics

The short period represents one of the two longitudinal modes of a traditional aircraft. A description of an aircraft’s short period is captured well by Thomas R. Yechout’s *Introduction to Aircraft Flight Mechanics*.

“The short period mode is characterized by complex conjugate roots with a moderate to relatively high damping ratio and relatively high natural frequency and damping frequency (short period). It is easily demonstrated by first trimming the aircraft and then disturbing in from trim with a forward-aft-neutral pitch stick input (commonly called a doublet). The resulting response back to trim can be either first-order (exponential decay) or second-order (oscillatory).” [75, p. 358].

Simulation. A pitch doublet is simply a two-sided square pulse to a elevator control input used to excite the short period mode. Figure 6 shows an example of an input amplitude/duration for a square doublet. In this case, the input is elevator deflection (δ_e). From the aircraft’s trimmed condition, and pitch doublet is executed using a time history of control inputs and the aircraft’s time history of states recorded.

Flight. In flight the Learjet’s VSS computer incorporated a PTI functionality designed for the collection of OLFQ. The use of a computer to produce square wave inputs reduces the error generated by erratic pilot input and makes comparison between simulation and flight test more effective. To excite the predicted short period natural frequency, the amplitude and frequency of pitch doublet is shown in Table 5.

Evaluation Criteria. The FAA’s *14 CFR 60* contains metrics for evaluating the pitch response and short period dynamics, Table 6.

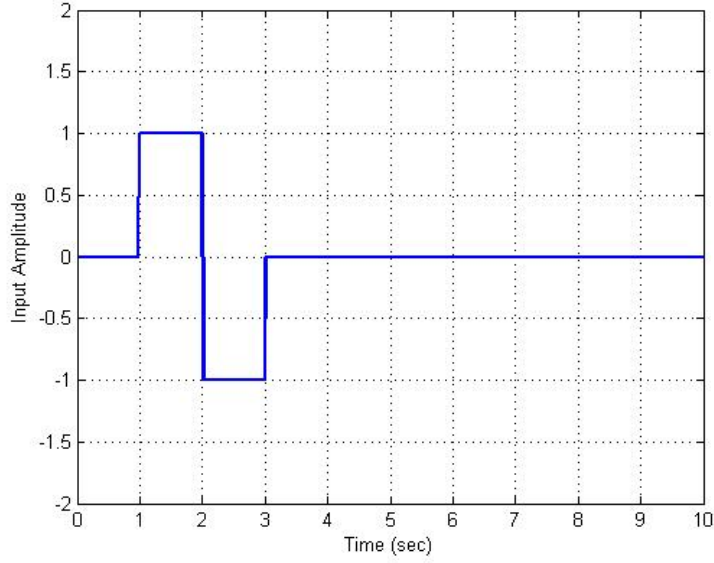


Figure 6. Doublet Input Example

Table 5. Short Period PTI

Control Surface	Amplitude	Frequency
Elevator Doublet (δ_e)	$\pm 1^\circ$	3.7 rad/sec

In addition to the FAA's metrics, short period natural frequency and damping ratio are calculated for both flight test and simulation and then compared. To determine the natural frequency and damping ratio from a time history without *a priori* knowledge of the modal parameters, one can measure the free response of the system subject to a displacement, in this case a doublet. For a second-order oscillatory response, such as the short period, the short period natural frequency (ω_{nsp}) and damping ratio (ζ_{sp}) can be found using the **logarithmic decrement** method:

$$\Delta = \ln(x_1) - \ln(x_2) = \ln\left(\frac{x_1}{x_2}\right) \quad (39)$$

where $x_1 = x(t_1)$ is a local maxima at t_1 and $x_2 = x(t_2)$ is the next local maxima at t_2 . From here, the damped natural frequency is dependent only on the time between

Table 6. Pitch Response and Short Period Dynamics [25, p. 57]

Parameter	Tolerance
Pitch (θ)	$\pm 1.5^\circ$
Pitch Rate (Q)	$\pm 2^\circ/sec$

the two peaks, Equation 40.

$$\omega_d = \frac{2\pi}{t_2 - t_1} \quad (40)$$

Using Equations 39 and 40, the damping ratio can be found by Equation 41 and natural frequency by Equation 42. Further information on this method can be found in [56].

$$\zeta = \frac{\Delta/2\pi}{\sqrt{1 + (\Delta/2\pi)^2}} \quad (41)$$

$$\omega_n = \frac{\omega_d}{\sqrt{1 - \zeta^2}} \quad (42)$$

3.5.2.2 Phugoid Dynamics

The phugoid represents the second longitudinal mode of a traditional aircraft.

“The phugoid mode is characterized by complex conjugate roots with a relatively low damping ratio and natural/damped frequency (long period). It is demonstrated by trimming the aircraft in level flight, then inputting aft stick for approximately 2-3 s, bleeding off some airspeed, then then returning the stick to the neutral (trimmed) position. The resulting response is usually oscillatory with significant variations in pitch attitude and airspeed while angle of attack remains fairly constant” [75, p. 359].

Simulation. In simulation the phugoid mode is excited by first trimming the model using methods previously discussed then reinitializing the simulation with an airspeed slightly offset from the trimmed value (≈ 20 KIAS). The period, natural frequency, and damping ratio can then be calculated using the method outlined in Section 3.5.2.1.

Flight. In flight the phugoid mode is excited much like the method in simulation. First, a “trim-shot” is conducted then a slight pitch-up to decrease airspeed followed by a pitch-down once the aircraft reaches a decreased airspeed. Once the aircraft reaches level flight, the controls are released and aircraft is free to seek a equilibrium condition. From the time history, the period, natural frequency, and damping ratio can then be calculated using the method outlined in Section 3.5.2.1.

Evaluation Criteria. The FAA’s *14 CFR 60* contains metrics for evaluating the phugoid dynamics, Table 7.

Table 7. Phugoid Dynamics [25, p. 57]

Parameter	Tolerance
Period	$\pm 10\%$
Damping Ratio (ζ_p)	$\pm .02$

In addition to the FAA’s metrics, the phugoid natural frequency and damping ratio are calculated for both flight test and simulation and then compared.

3.5.2.3 Roll Response

The roll response of the aircraft is significant because it captures one of the three lateral-direction modes, the roll mode.

“The roll mode of an aircraft has a real root and a first-order (non-oscillatory) response that involves almost a pure rolling motion about the x stability axis... The roll mode can be excited by a disturbance or an aileron input. If a step aileron input δ_a is made to the aircraft, there is an exponential rise in the roll rate ($\approx \dot{\phi}$) until a steady-state roll rate is achieved” [75, p. 372].

Simulation. In simulation, the roll mode is excited by first trimming the model using methods previously discussed then the initiating a small step input to the ailerons (δ_a). An example input is shown in Figure 7. Since the roll mode is a first-order (non-oscillatory) mode, the rise time to reach a steady-state roll rate is of interest in calculating the roll mode time constant (τ_r). The roll mode time constant is the time it takes for the aircraft to achieve 63.2% of the steady-state roll rate [75, p. 377].

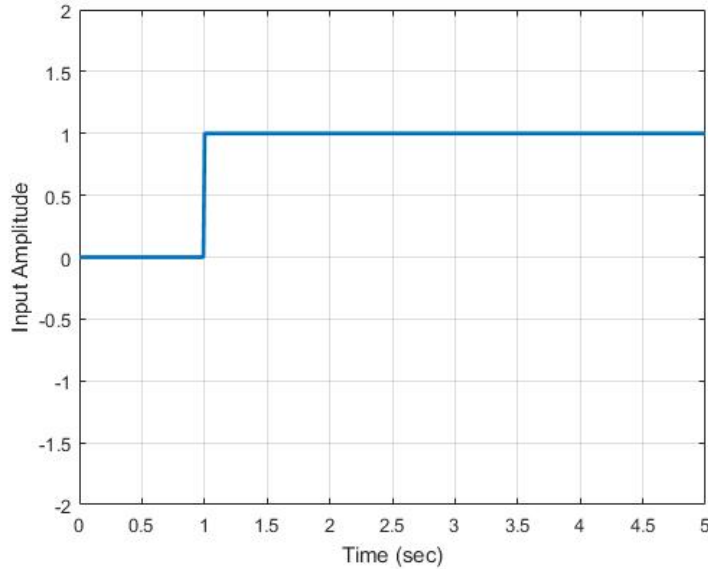


Figure 7. Step Input Example

Flight. In flight, the roll mode is excited much like the method in simulation. First, a “trim-shot” is conducted then a small PTI is executed through the VSS

interface. The PTI input parameters are shown in Table 8.

Table 8. Roll Response PTI

Control Surface	Amplitude	Duration
Ailerons (δ_a)	1° step	4 sec

From the time histories of roll angle (ϕ) and roll rate (P), the roll mode time constant (τ_r) can be calculated.

Evaluation Criteria. The FAA’s *14 CFR 60* contains metrics for evaluating the roll (aileron) response of an aircraft simulation, Table 9.

Table 9. Roll (Aileron) Response [25, p. 58]

Parameter	Tolerance
Deviation	$\pm 10\%$
Roll Rate (p)	$\pm 2^\circ/sec$

While not a requirement for the FAA, the roll mode time constant (τ_r) will be calculated and compared to the simulation result.

3.5.2.4 Dutch Roll Dynamics

The second lateral-directional mode of motion is the Dutch roll.

“The Dutch roll mode is a second-order response (complex conjugate roots) usually characterized by concurrent oscillations in the three lateral-directional motion variables β , ϕ , ψ . . . The Dutch roll usually begins with a sideslip perturbation followed by oscillations in roll and yaw” [75, p. 372].

Simulation. In simulation, the Dutch roll is excited using a procedure similar to that of the short period. Instead of using the elevator as the input channel, the Dutch roll mode is excited using a doublet input to the rudder (δ_r). From the aircraft's trimmed condition, a time history of rudder input matching that of the flight test PTI is inputted into the simulation and the resulting sideslip (β), roll angle (ϕ), and yaw rate (R) is recorded for analysis.

Flight. In flight, the Dutch roll mode is excited much like the method in simulation. First, a "trim-shot" is conducted then a small PTI is executed through the VSS interface. The PTI input parameters are shown in Table 10.

Table 10. Dutch Roll PTI

Control Surface	Amplitude	Frequency
Rudder Doublet (δ_r)	$\pm 1^\circ$	1.96 rad/sec

Determination of the Dutch roll natural frequency (ω_{ndr}) and damping ratio (ζ_{dr}) is calculated using the same method described in Section 3.5.2.1.

Evaluation Criteria. The FAA's *14 CFR 60* contains metrics for evaluating the yaw (rudder) response and Dutch roll (yaw damper OFF), Table 11.

Table 11. Yaw (Rudder) Response [25, p. 58] and Dutch Roll (Yaw Damper OFF) [25, p. 59]

Parameter	Tolerance
Yaw Rate (R)	$\pm 2^\circ/\text{sec}$ or 10%
Period	± 0.5 sec or $\pm 10\%$ of period
Damping Ratio	± 0.02

While not a requirement for the FAA then Dutch roll natural frequency (ω_{ndr}) will be calculated and compared to the simulation result.

3.5.2.5 Spiral Stability

The third lateral-directional mode of motion is the spiral mode.

“The spiral mode is a first-order response (real root) that involves a relatively slow roll and yawing motion of the aircraft. It can be stable or unstable. The spiral is usually initiated by a displacement the roll angle and appears as a descending turn with increasing roll angle if unstable. If the spiral is stable, the aircraft simply returns to wings level after a roll angle displacement” [75, p. 373].

Simulation. In simulation the spiral mode is excited by trimming the aircraft to straight, level, and unaccelerated flight (SLUF) then initializing the model at 20 degrees of bank (typically both left and right bank are tested separately). The aircraft’s roll angle after 20 seconds is recorded and compared to the flight test result.

Flight. Much like simulation, the flight test spiral mode excitation is executed from a “trim-shot” then by setting 20 degrees of bank and releasing the controls. The roll angle (ϕ) time history is recorded for 20 seconds and the final bank angle is noted.

Evaluation Criteria. The FAA’s *14 CFR 60* contains metrics for evaluating the spiral stability, Table 12.

Table 12. Spiral Stability [25, p. 58]

Parameter	Tolerance
Bank angle	Correct trend and $\pm 2^\circ$ or $\pm 10\%$ bank angle in 20 seconds

Correct trend will be noted as STABLE if the bank angle decreases to less than 20 degrees or UNSTABLE if increasing greater than 20 degrees after 20 seconds.

3.5.3 Baseline Performance Maneuver

For the exploration of off-nominal performance, a baseline performance maneuver is designed: 1) to simulate a real-world mission requirement, 2) to stress the simulation model in multiple axes, and 3) to provide a baseline for comparison between simulation and flight test. Since a multistate approach to the design of a new aircraft would most likely apply to a long-endurance platform (though not limited to), the ability to orbit in a single location and to maintain that orbit is chosen as the mission requirement. To stress the simulation in multiple axes, a coordinated turn within the safety envelope is desired to stress the lateral-directional axis. Maintaining a desired climb rate stresses the elevator control. Maintaining a constant airspeed stresses the throttle control. As a result, the performance maneuver attempted is a constant airspeed, constant climb rate coordinated turn beginning from equilibrium (trimmed) state. The desired parameters are shown in Table 13, with a 3-D view shown in Figure 8.

Table 13. Baseline Performance Maneuver

Parameter	Desired Value
Rate of Climb (V_{climb})	500, 1000 and 2000 fpm
Bank (ϕ)	30°
Sideslip (β)	0°
Velocity (V_c)	250 KIAS

Three climb rates were attempted to stress the simulation's predictive ability with regards to specific excess power (P_s) and maneuver feasibility, where P_s is given by Equation 43:

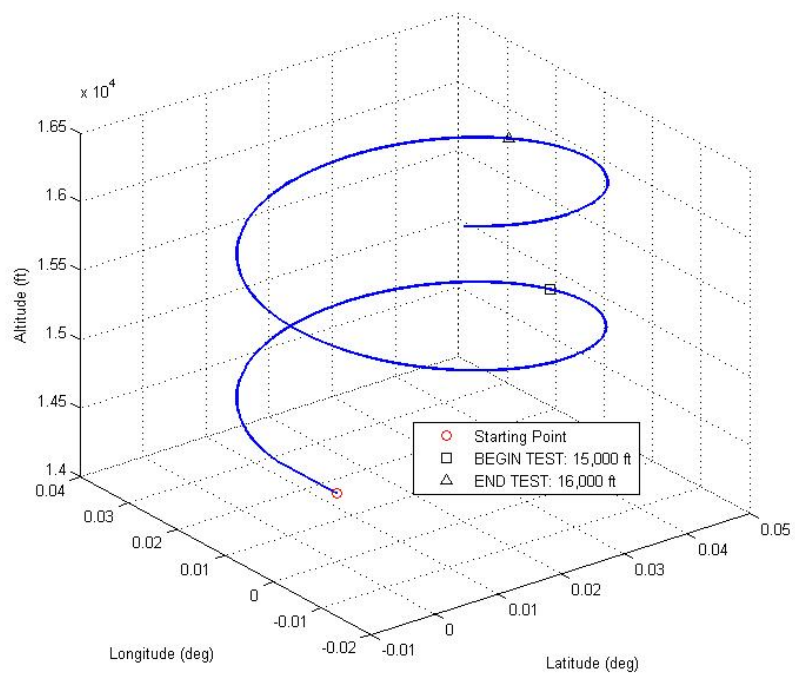


Figure 8. Baseline Performance Maneuver

$$P_s = \frac{dH}{dt} + \frac{V}{g} \frac{dV}{dt} \quad (43)$$

where $\frac{dH}{dt}$ is change in altitude over time, V is velocity, g is gravity, and $\frac{dV}{dt}$ is the change in velocity over time. For a constant airspeed climb, the zero specific excess power occurs when the aircraft can no longer maintain the desired climb rate.

The baseline performance maneuver is conducted using the following aircraft configurations. In this section “Failure” represents the performance threshold described by the multistate design process, discussed in Section 2.1. Performance threshold is essentially the way elements of a larger system can fail or what varying levels of failure is the process modeled to. For this research, the performance threshold for ailerons and rudder are: frozen and zero deflection or fully operational. For elevator, the performance threshold is: frozen at trimmed airspeed or fully operational. For engine, the performance threshold is: idle power (simulating failure) or fully operational. Using the VSS interface, the flight controls are digitally restricted or in the case of an engine, the throttle is placed at idle power.

1. Nominal
2. Rudder Failure
3. Aileron Failure
4. Engine Failure
5. Engine and Aileron Failure
6. Engine and Rudder Failure
7. Aileron and Rudder Failure
8. Elevator Failure

Simulation. Within JSBSim a series of PID controllers are utilized to execute the baseline performance maneuver. The method for determining the appropriate gains is outlined in Section 3.4.1. Using the desired parameters for the baseline performance maneuver as the set-points for the error signals. The following autopilot switch configuration is used to fly the maneuver based on the configuration, Table 14.

Table 14. Table of Autopilot Switch Settings

Control Parameter	V_c	V_{climb}	V_{climb}	ϕ	ϕ	ϕ	β	β
Control Surface	δ_t	δ_e	δ_t	δ_a	δ_r	δ_{tr}/δ_{tl}	δ_r	δ_{tr}/δ_{tl}
Nominal	ON	ON	OFF	ON	OFF	OFF	ON	OFF
Rudder Failure	ON	ON	OFF	ON	OFF	OFF	OFF	ON
Aileron Failure	ON	ON	OFF	OFF	ON	OFF	ON	OFF
Engine Failure	ON	ON	OFF	ON	OFF	OFF	ON	OFF
Engine and Aileron Failure	ON	ON	OFF	OFF	ON	OFF	ON	OFF
Engine and Rudder Failure	ON	ON	OFF	ON	OFF	OFF	OFF	ON
Aileron and Rudder Failure	ON	ON	OFF	OFF	OFF	ON	OFF	OFF
Elevator Failure	ON	OFF	ON	ON	OFF	OFF	ON	OFF

The simulation is initialized from a trimmed equilibrium condition at an altitude below the desired databand (typically $\approx 14,000$ ft MSL). After several seconds have elapsed, the autopilot switches are switched to their appropriate position described in Table 14. The simulation is initialized at a lower altitude to allow for the transient response of the control law to settle out. This is illustrated by Figure 9, which shows the normalized positions of the control surfaces and throttle.

From Figure 9 one can see the large oscillations in all four control effectors (δ_e , δ_a , δ_r , and δ_t). This is a product of the PID gains chosen to control each parameter. It is important to note that gain optimization with rate saturation or smoothness of

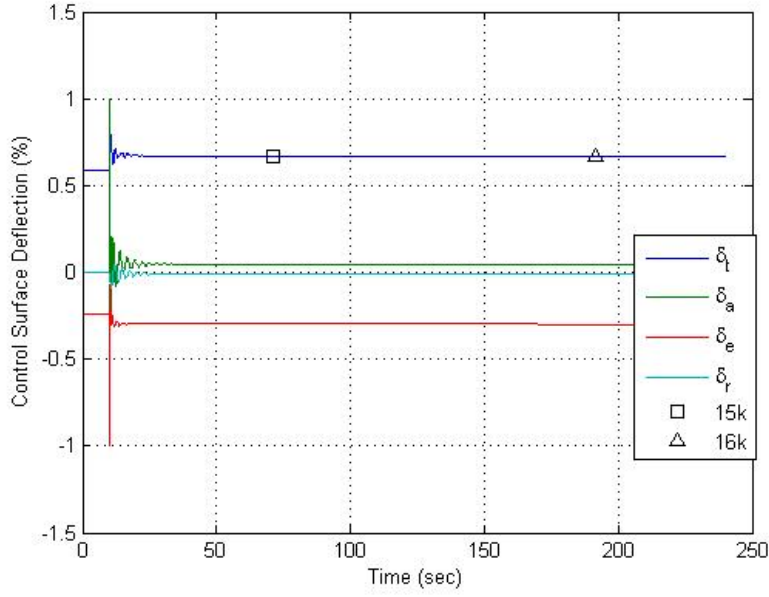


Figure 9. Performance Maneuver Control History Isolation

maneuver was not considered, only that the controller could drive the error to near zero in a reasonable amount of time. One can see from Figure 9 that the control effectors are stable at roughly 25 seconds into the simulation. A square ($\square$) is shown to highlight when the aircraft crosses 15,000 ft MSL and a triangle ($\triangle$) is shown at the top of the desired databand (16,000 ft MSL). To compare with flight test, only the steady-state deflections are captured, as shown by the time history between the $\square$ and $\triangle$.

Flight. Within the Learjet 25's VSS system, various configuration files are created to isolate the control surfaces and simulation failure modes. This is accomplished digitally by setting the command gain to 0, shown by Figure 10. Since the controls in the left seat (safety pilot) are fully reversible, verification of the configuration files can be accomplished on the ground, adding to the margin of safety. Additionally, the safety pilot maintains direct control of the aircraft in the case of a safety trip or VSS system failure.

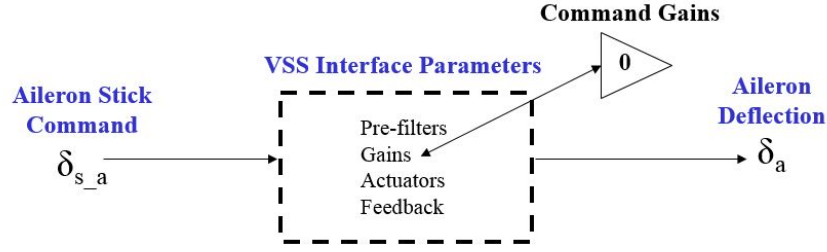


Figure 10. VSS Control Channel Isolation

Due to time and funding constraints, the autopilot controllers are not integrated into the test aircraft. The result of this is the simulation time history of parameters is much smoother than the hand-flown maneuver. To increase the stability of the hand-flown maneuvers, the maneuver is initiated (like the simulation) at an altitude less than the databand. The time for the maneuver parameters to become stable depended on individual pilot technique and the desired climb rate. To ensure the maneuvers are high quality, tolerances are set for each parameter (Table 15). Deviations outside the tolerances requires a termination of maneuver and reset.

Table 15. Performance Maneuver Tolerances

Parameter	Tolerance
Airspeed (V_c)	± 10 KIAS
Bank (ϕ)	$\pm 3^\circ$
Sideslip (β)	$\pm 2^\circ$
Rate of Climb (V_{climb})	± 150 fpm (± 250 for 1000 and 2000 fpm climb rates)

Each configuration was accomplished several times and overall maneuver quality assessed. The maneuver with the closest tracking of the desired performance parameters is selected for comparison to the simulation result. Since multiple parameters are being tracked, a MATLAB script is used to assess overall quality of each maneuver.

An example of this script’s output is shown in Figure 11.

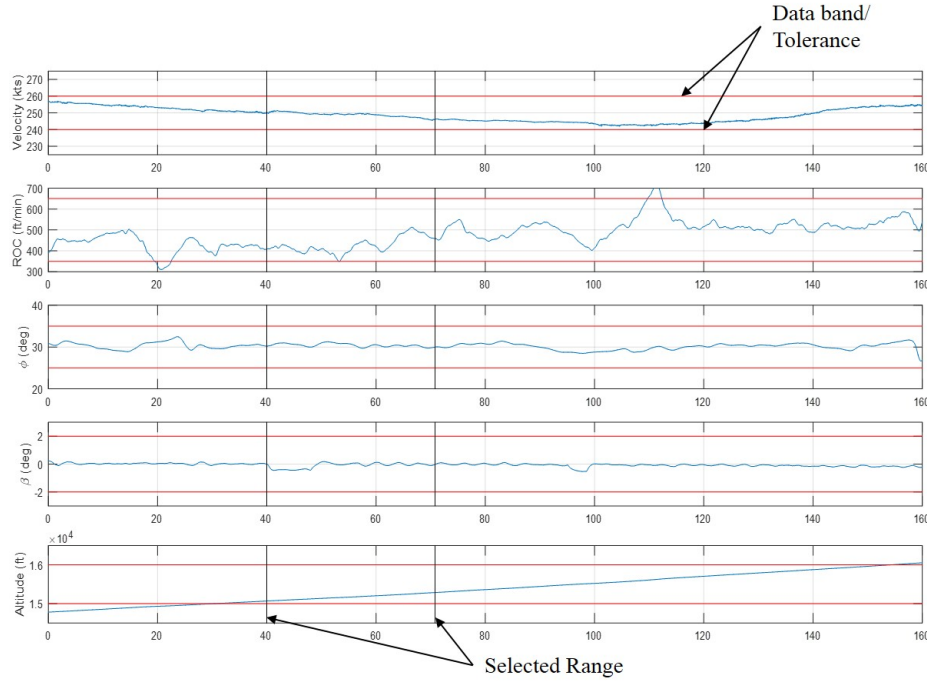


Figure 11. Maneuver Quality Assessment (Example)

The x-axis of Figure 11 represents time while the y-axes represent the various parameters evaluated. Red horizontal bars represent the tolerances described in Table 15 and black vertical bars represent a selected “high-quality” section of the maneuver. This method is used to evaluate maneuver quality for all the configurations studied.

Evaluation Criteria. Investigation of off-nominal flight control performance falls into the subjective test requirements outlined in the FAA’s *14 CFR 60*, “all system functions will be assessed for normal and, when appropriate, alternate operations. Normal, abnormal, and emergency procedures associated with a flight phase will be assessed during the evaluation of maneuvers or events within that flight phase” [25, p. 101]. The CFR does not specify what parameters should be used to assess a simulator for off-nominal performance. As a result, nominal evaluation parameters are used as outlined by Table 16.

Table 16. FAA 14 CFR 60 Criteria Summary [25]

Roll Response		Pitch Response		Yaw Response		Engine Thrust	
P	$\pm 2^\circ/\text{sec}$	Q	$\pm 2^\circ/\text{sec}$	R	$\pm 2^\circ/\text{sec}$	Net Thrust	$\pm 5\%$
		δ_e	$\pm 1^\circ$	δ_r	$\pm 1^\circ$		

3.6 Chapter Summary

In this chapter, definitions of the state vectors and control vector were provided to aid understanding of the results in Chapter IV. Next, several limitations of the aircraft were discussed to give the reader a sense of why specific maneuvers were chosen as the test points. A discussion of the method for matching the initial conditions of each test was provided. A summary of the simulation control integration and test maneuvers were discussed. Finally, the research methodology was discussed to include the test points conducted, the method of simulation and flight, and the evaluation criteria used. Chapter IV presents the results of this methodology.

IV. Results

Initial development of the simulation model was conducted in mid-2015. The simulation model was updated in March 2016 to include updated stability and control derivatives collected from flight test during the summer of 2015. Real-world flight test was conducted from August 29 - September 8th, 2016 at Edward AFB, California, as part of a USAF Test Pilot School test management project (TMP). Subsequent simulations were conducted, after updating with the nominal flight test data in September 2016. Data was collected, analyzed and conclusions were drawn in the months following the flight tests. This chapter presents the results collected from simulation and flight-test and the conclusions drawn from the comparison.

During the time-span of this research, there was a parallel effort by the Calspan Corporation to develop a MITL simulator using the Learjet 25D as the model. The intent of this simulator was for aircrew training and for software verification. This model contained Learjet parameters extracted at the same test conditions (250 KIAS and 15,000 ft MSL) as the test maneuvers were centered around. This model (called the “Hotbench Lite (HBL)”), while its code is proprietary in nature and not discussed in the previous sections, allows an additional point of comparison between simulation and flight test. The three sources of data become 1) JSBSim, 2) HBL, and 3) flight test data.

4.1 Data Presentation

Each figure displayed in Chapter IV contains header information that describes the test configuration, pressure altitude, fuel weight (or range), test date, temperature (in relation to International Standard Atmosphere (ISA), and evaluation criteria (displayed as an error-bar). The single error-bar was chosen for figure clarity but

applies to the whole time history or data-set. The single error-bar indicates that the entire time history or data-set lies within the evaluation criteria. If multiple error-bars are shown, red error-bars show where JSBSim did not meet evaluation criteria. Within the figures, flight test data will always be displayed in **black**, JSBSim data in **blue**, and HBL in **red**.

4.2 Trim

During flight test, 22 “trim-shots” were conducted (11 in a nominal condition and 11 in a simulated engine-out condition). Gross weight, outside air temperature, altitude, velocity, pitch attitude (θ), elevator deflection (δ_e), and thrust (δ_t) was recorded and compared with the simulation results. For the off-nominal case, rudder deflection (δ_r), angle of sideslip (β), and bank angle (ϕ) was also compared. The Learjet 25D controlled pitch using an all-moving horizontal stabilizer and elevator. The trim was set using a combination of elevator position and stabilizer position once the aircraft elevator hinge limits were reached. The stabilizer could be set to AUTO or MANUAL trim mode. In the AUTO mode the stabilizer position was automatically adjusted as the elevator approached its hinge moment limit. In the MANUAL mode all stabilizer adjustments were controlled by the left seat (safety pilot) as the hinge moment approached its limit. The initial intent was to fix the horizontal stabilizer at a single angle then compare the resulting elevator deflection. Due to hinge moment limits, it was discovered that a fixed stabilizer angle was only possible for a ± 10 KIAS airspeed range. Therefore, a composite elevator-stabilizer deflection was required to compare to the simulator result. Within the simulation the stabilizer was fixed at -1.9° while the flight test stabilizer varied from -1.5° to -2.5° . Using their control derivatives, a relative weight of their contributions to lift and pitching moment were used to determine an effective elevator-stabilizer position (δ_{es}), given by Equation 44:

$$\delta_{es} = 0.63 * \delta_s + 0.37 * \delta_e \quad (44)$$

where δ_s is the stabilizer deflection and δ_e is the elevator deflection. This composition rating replaces elevator deflection in the remaining figures in Chapter IV.

4.2.1 Nominal trim

The variation in control and thrust required to hold a trimmed condition is shown in Figures 12 and 13 for the gross weight range described in the header.

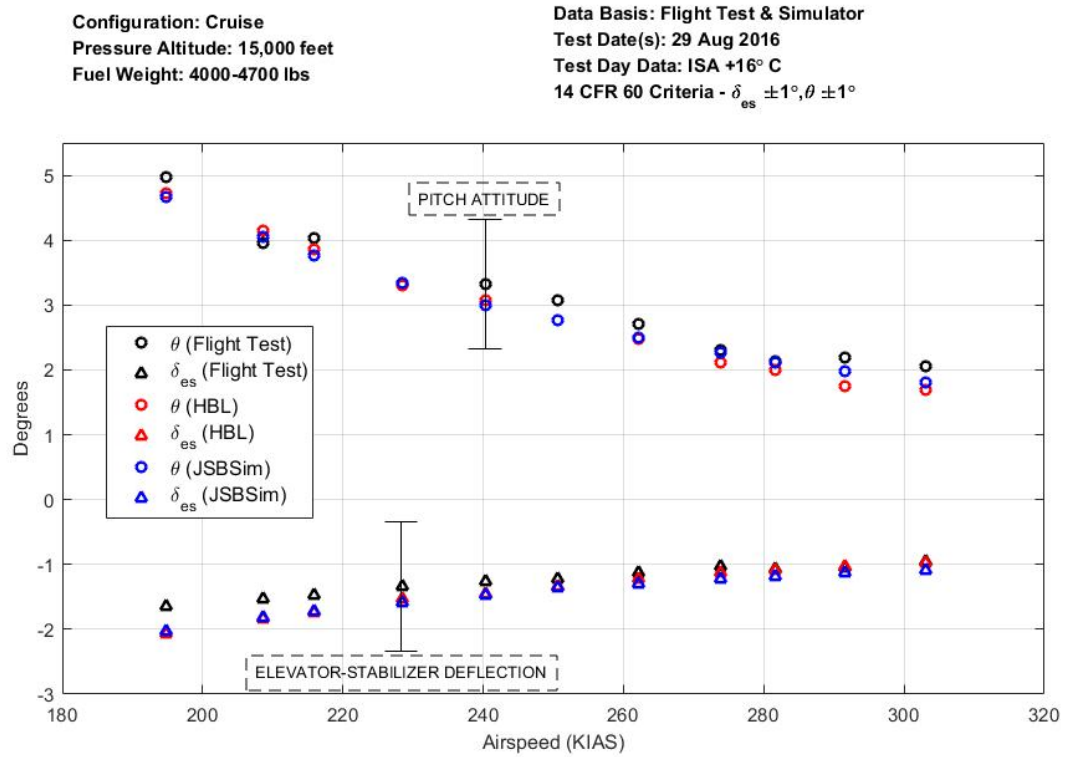


Figure 12. Nominal Trim: Pitch and Elevator-Stabilizer Deflection

Figure 12 shows the pitch attitude (shown by $\circ$) and effective elevator-stabilizer position (shown by $\triangle$) required to hold a trimmed condition. One can see that both HBL and JSBSim are well inside evaluation criteria, shown by the single black

error-bar on a single data point.

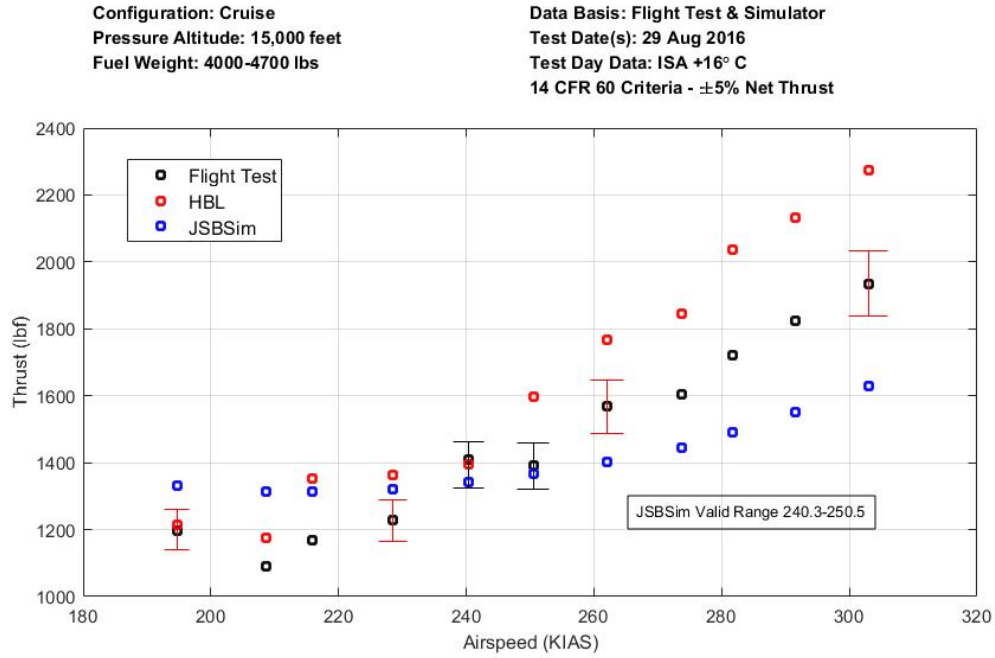


Figure 13. Nominal Trim: Net Thrust

Figure 13 shows the net thrust required to hold a trimmed condition. Black error-bars are displayed as black if within criteria for JSBSim. Red error-bars are shown when the simulation was outside evaluation criteria. JSBSim was valid from 240.3 to 250.5 KIAS. The fact that the thrust predictions are off near 200 KIAS and 300 KIAS points to the fact that the drag model was extracted at a single condition, 250 KIAS. It makes sense that JSBSim under-predicts drag (i.e. thrust, since they are related) at the higher speeds and over-predicts drag at the lower speeds. As a result, the tolerance bounds of the performance maneuver, discussed in later sections, are ± 10 KIAS, which is the window where the thrust predictions are accurate.

Using TIC and comparing the 11 data points as a vector within Equation 15, JSBSim provides a better match to the flight test result (shown by the lower TIC value for all the parameters, Table 17).

Table 17. Nominal Trim TIC

Parameter	JSBSim	HBL
Pitch Attitude (θ)	0.074	0.078
Elevator-Stabilizer (δ_{es})	0.035	0.064
Net Thrust	0.064	0.068

4.2.2 Off-Nominal Trim

During flight test, 11 engine-out “trim-shots” were conducted between 200 and 300 KIAS at 15,000 ft MSL. The maneuver was conducted in flight by first setting pitch attitude and thrust such that altitude and airspeed was held constant. Then rudder was used to zeroize the angle of sideslip (β) while ailerons were used to induce a bank (ϕ) to hold a steady heading. In addition the the nominal trim parameters, bank angle (ϕ) and rudder deflection (δ_r) was recorded and compared against the simulator solution. Variation in control and thrust required to hold a trimmed, steady-heading condition in an engine-out scenario is shown in Figures 14, 15, and 17.

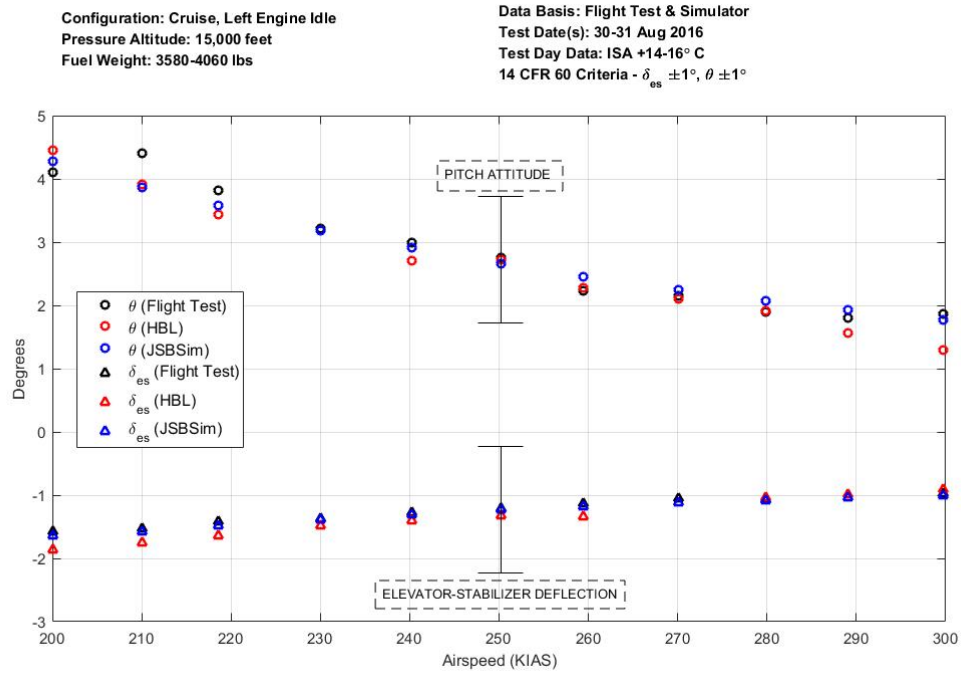


Figure 14. Off-Nominal Trim: Pitch and Elevator-Stabilizer Deflection

Figure 14 shows the pitch attitude (shown by $\bigcirc$) and effective elevator-stabilizer position (shown by $\triangle$) required to hold a trimmed condition. One can see that both HBL and JSBSim are well inside evaluation criteria, shown by the single black error-bar on a single data point.

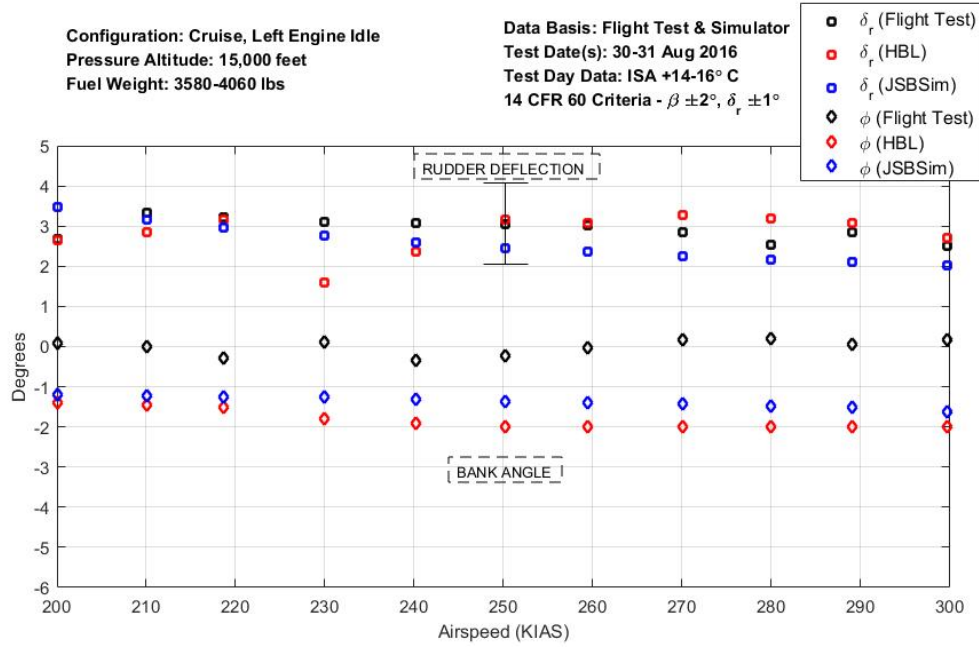


Figure 15. Off-Nominal Trim: Rudder Deflection and Bank Angle

Figure 15 shows the rudder deflection (shown by $\square$) and bank angle required (shown by $\diamond$) required to hold a trimmed condition. The maneuver was flown with rudder as required to zeroize sideslip angle (β). Therefore, sideslip is zero for the three cases, with no deviations outside the evaluation criteria of $\pm 2^\circ$. One can see that JSBSim is inside evaluation criteria for the full range of airspeeds, shown by the single black error-bar on a single data point. One can see that the flight test data for bank angle is centered about 0 degrees. Careful analysis of the aircraft mechanics (or equations) suggest a small bank angle should be required to hold a steady heading in a single engine configuration. An analysis of a nominal trim data described in Section 4.2.1, showed a small bias existed in aileron deflection and bank angle when in a symmetric thrust configuration. This small bias in aileron deflection (δ_a) was 1.65° and for bank angle (ϕ) was 0.81° . No bias was found in rudder deflection (δ_r). These biases were attributed to aerodynamic asymmetries with the flight test aircraft, which

were not modeled within the simulations. For this reason, the biases were removed from the flight test data for comparison to simulation.

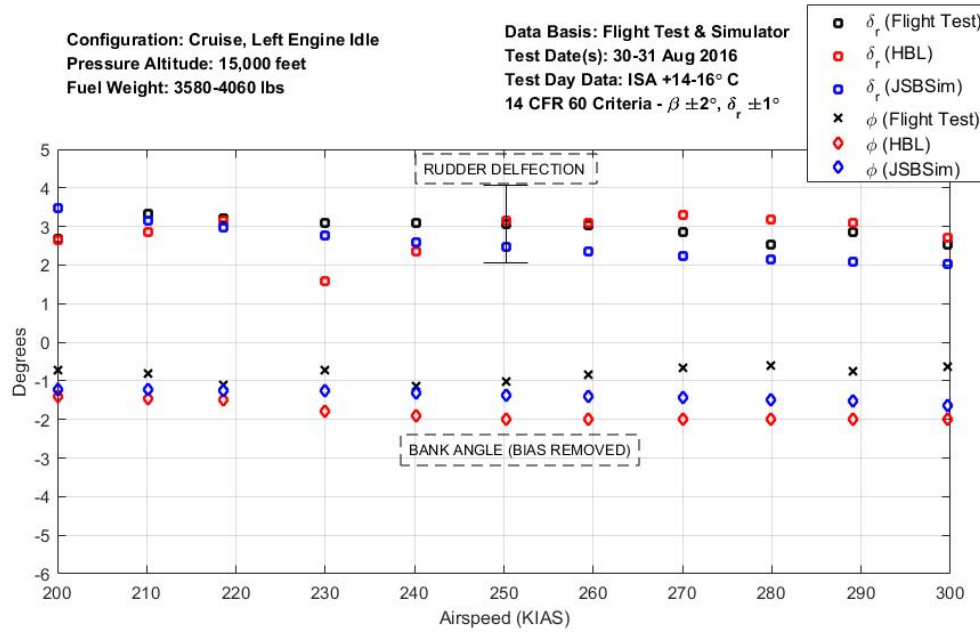


Figure 16. Off-Nominal Trim: Rudder Deflection and Bank Angle (bias removed)

Figure 16 updates Figure 15 with a removal of the roll angle bias of 0.81° , **X** represents the updated flight test data. As a result, both JSBSim and HBL fall within evaluation criteria for rudder deflection and by the nature of the maneuver, they meet sideslip criteria.

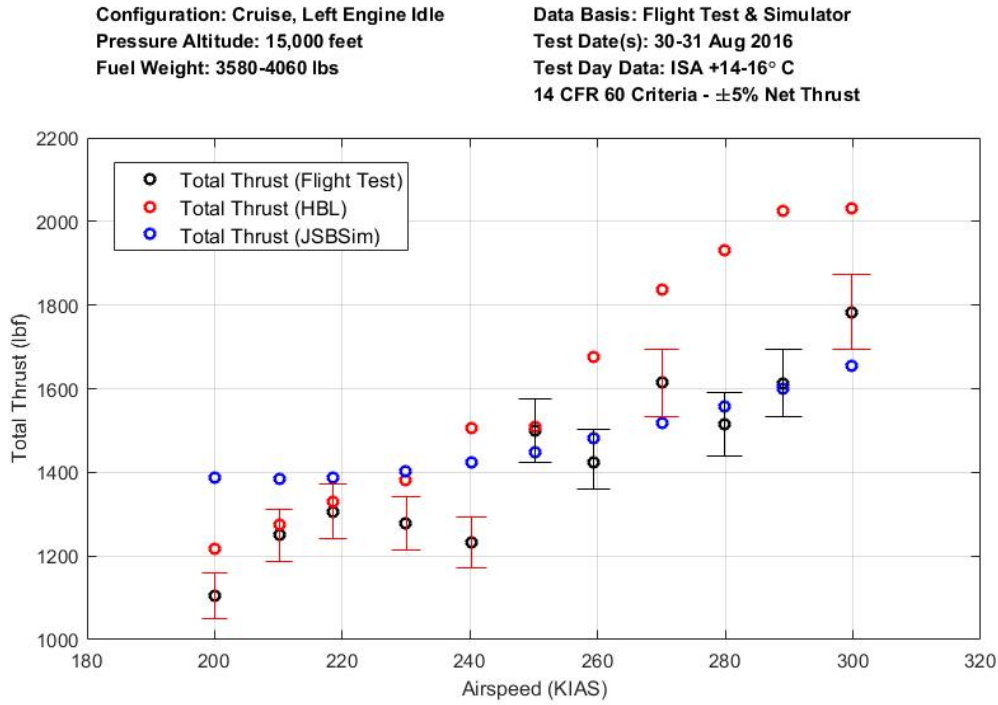


Figure 17. Off-Nominal Trim: Net Thrust

Figure 17 shows the net thrust required to hold a trimmed condition. Net thrust, in this case, is the sum of left engine IDLE thrust and right engine thrust. Black error-bars are displayed as the bounds of the valid range for JSBSim, red when outside evaluation criteria. One can see much more variability in thrust predictions for flight test. Variations were attributed to throttle technique and the fact that very small changes ± 50 lbs of thrust produced minimal change the aircraft's states. As it is, thrust predicted around 250 KIAS is accurate $\pm \approx 10$ KIAS. This agrees with the fact that the drag model was extracted at a single condition (250 KIAS), thus is only valid for a very small window centered around 250 KIAS. The nominal trim information agrees with this, and the tolerances on the performance maneuver 250 ± 10 KIAS puts data within the valid thrust predicted window.

Using TIC and comparing the 11 data points as a vector within Equation 15, JSBSim provides a better match to the flight test result (shown by the lower TIC value) for all the parameters, Table 18. The value in parenthesis for roll angle represents data with the bias removed. The large value of TIC for bank angle highlights a limitation of the metric when a bias is present in the data.

Table 18. Off-Nominal Trim TIC

Parameter	JSBSim	HBL
Pitch Attitude (θ)	0.036	0.051
Elevator-Stabilizer (δ_{es})	0.017	0.058
Bank Angle (ϕ)	0.893 (0.279)	0.914 (0.395)
Rudder (δ_r)	0.098	0.101
Net Thrust	0.045	0.077

Parentheses indicate TIC with bias removed

4.3 Nominal Longitudinal and Lateral-Directional Modes

The nominal longitudinal and lateral-directional mode parameters were calculated using methods outlined in Chapter III and compared to flight test data. Additionally, time histories of aircraft response to input are shown in the subsequent sections and evaluated against FAA metrics for validation.

4.3.1 Short Period Dynamics

The short period dynamics of the test aircraft were analyzed through the use of a PTI. The pitch response is shown by Figure 18 and the pitch rate response is shown by Figure 19.

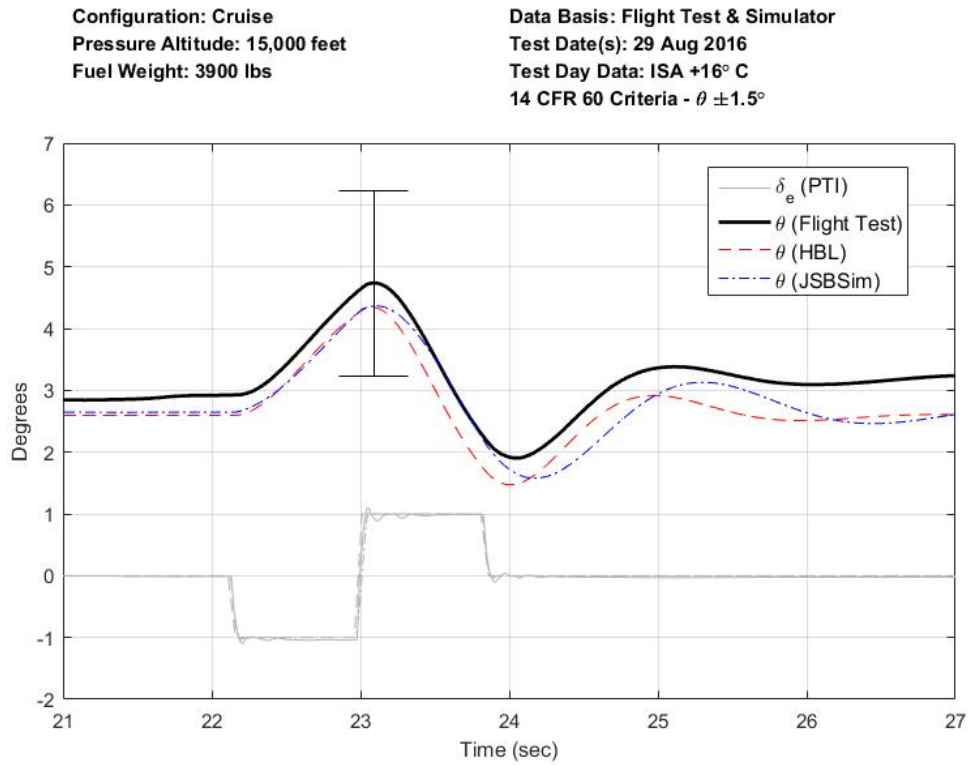


Figure 18. Short Period Dynamics: Pitch Response

Figure 18 shows the pitch response to the pitch doublet PTI. PTI overlays for flight test and simulation are shown by the gray lines. It is clear that the pitch response for both simulations lie within the evaluation criteria.

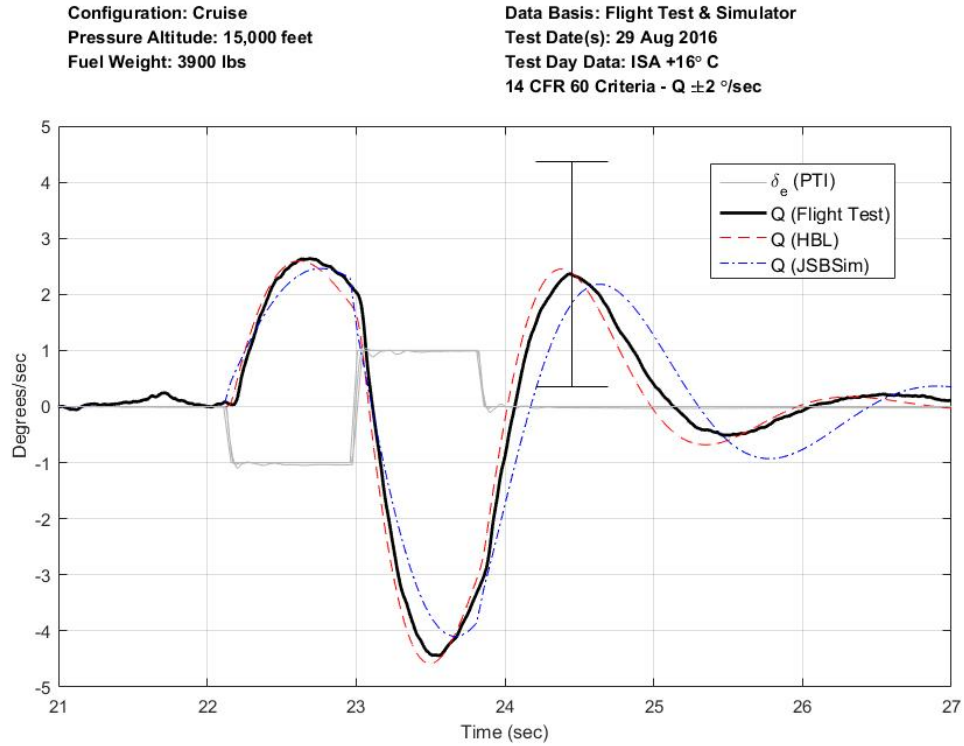


Figure 19. Short Period Dynamics: Pitch Rate Response

Figure 19 shows pitch rate response to the pitch doublet PTI. It is clear that the pitch response for both simulations lie within the evaluation criteria. Modal short period parameters are displayed in Table 19.

Table 19. Short Period Modal Parameters

Parameter	Flight Test	JSBSim	HBL
Natural Frequency (ω_{sp})	3.15 rad/sec	2.90 rad/sec	3.40 rad/sec
Damping Ratio (ζ_{sp})	0.36	0.33	0.39

A TIC comparison for the simulations is shown in Table 20. The simulations were assessed to have essentially equal performance with respect to short period dynamics.

Table 20. Short Period Dynamics TIC

Parameter	JSBSim	HBL
Pitch Attitude (θ)	0.088	0.093
Pitch Rate (Q)	0.193	0.191

4.3.2 Phugoid Response

The phugoid response was evaluated against FAA *14 CFR 60* criteria as outline in Chapter III. Table 21 summarizes the results.

Table 21. Phugoid Modal Parameters

Parameter	Flight Test	JSBSim	HBL
Damping Ratio (ζ_{ph})	0.07	0.06	0.05
Period (T_{ph})	71 sec	66 sec	73 sec

Table 21 shows that both JSBSim and HBL fall within evaluation criteria for damping ratio (± 0.02) and for period ($\pm 10\%$). A time history of phugoid response was not required for evaluation criteria, therefore TIC was not calculated.

4.3.3 Roll Response

The roll response of the test aircraft were analyzed through the use of a step input PTI. The roll response is shown in Figure 20.

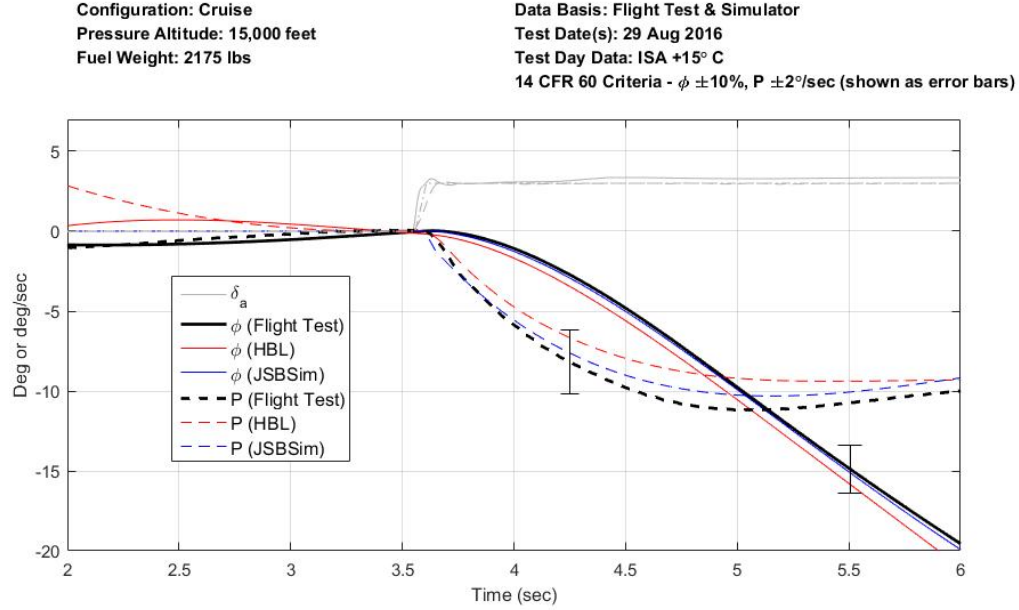


Figure 20. Roll Response

Figure 20 shows the aircraft's bank angle and roll rate response to a aileron step input. The dashed lines show the roll rate response (P) while the solid lines represent the change in bank angle (ϕ). Both simulations lie within evaluation criteria for roll rate and roll angle. Table 22 shows the calculated roll mode time constants.

Table 22. Roll Response Time Constants

Parameter	Flight Test	JSBSim	HBL
Roll Mode Time Constant (τ_R)	0.42 sec	0.41 sec	0.51 sec

Table 23 summarized the TIC metrics, with JSBSim providing the overall better match.

Table 23. Roll Response TIC

Parameter	JSBSim	HBL
Bank/Roll Angle (ϕ)	0.028	0.067
Roll Rate (P)	0.041	0.076

4.3.4 Dutch Roll Dynamics

The Dutch roll dynamics of the test aircraft were analyzed through the use of a doublet input PTI. The yaw rate response is shown in Figure 21.

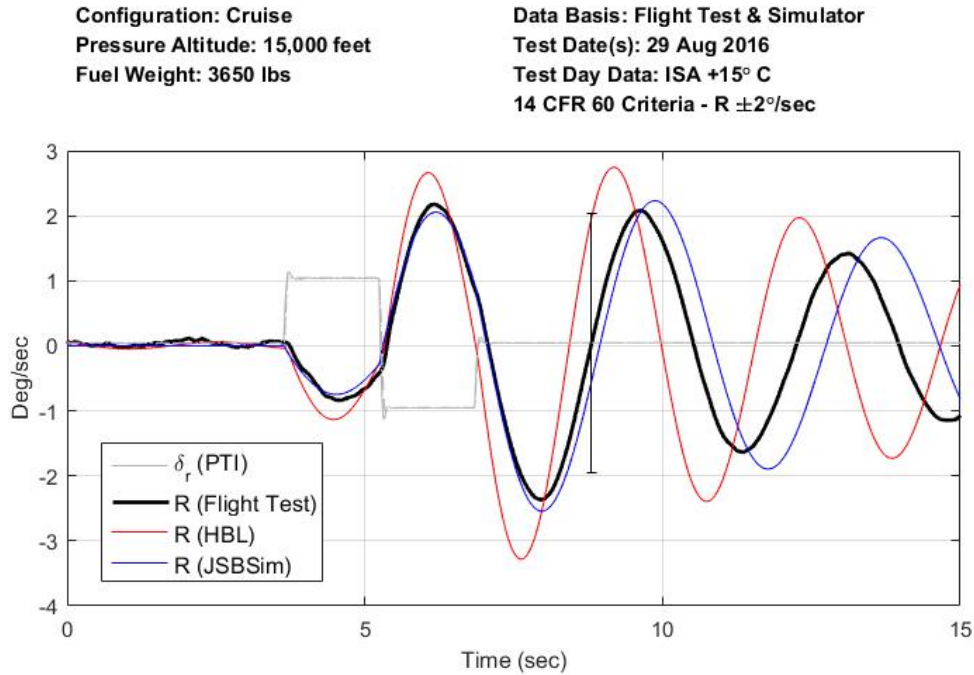
**Figure 21. Dutch Roll Dynamics: Yaw Rate**

Figure 21 shows the yaw rate response to a doublet input. This is one example of how generous the FAA criteria is for simulation validation, since Dutch roll is typically the most difficult mode to model correctly. A response could be completely out of phase with flight test and still meet evaluation criteria. Both simulations lie within

evaluation criteria. HBL lies on the edge of evaluation criteria for the majority of the time history thus was assessed to be meet requirements. Dutch roll modal parameters were calculated using methods outlined in Chapter III and are displayed in Table 24.

Table 24. Dutch Roll Modal Parameters

Parameter	Flight Test	JSBSim	HBL
Natural Frequency (ω_{DR})	1.80 rad/sec	1.72 rad/sec	2.00 rad/sec
Damping Ratio (ζ_{DR})	0.06	0.05	0.05
Period (T_{dr})	3.43 sec	3.8 sec	3.13 sec

A comparison of time histories using the TIC metrics shows that JSBSim provides a better match than HBL, Table 25. The phase error of each time history is the main driver at increasing the TIC values for this case. This shows another weakness in the TIC metric, since modal parameters were calculated to be very close to the truth data. Using a normalized time for each time history may be useful in forcing a phase match.

Table 25. Dutch Roll Dynamics TIC

Parameter	JSBSim	HBL
Sideslip (β)	0.375	0.466
Yaw Rate (R)	0.249	0.492

4.3.5 Spiral Stability

Spiral stability was evaluated using a stabilized banked turn followed by a release of the controls. Multiple tests were conducted in flight test and compared with the simulation result. Flight test agreed with simulation and showed a symmetric

response with both left and right hand turns. For conciseness, only the left bank turn response is shown, Figure 22.

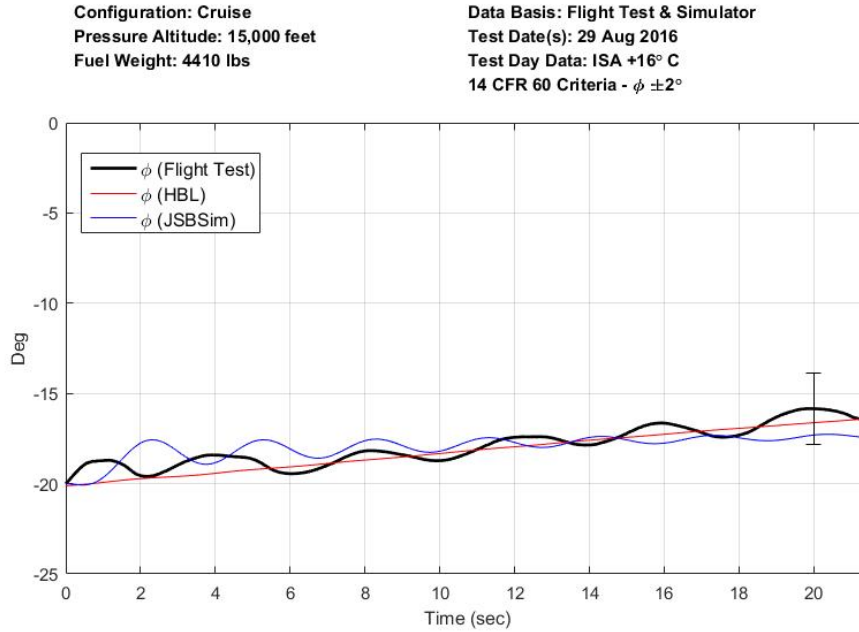


Figure 22. Spiral Stability

Figure 22 shows the spiral stability result of flight test and the two simulations. Both simulations lie within evaluation criteria and agree with the flight test result that the spiral mode of the aircraft is STABLE. Of note, the HBL simulation shows a linear response to a release of control while both flight test and JSBSim capture an oscillation in roll angle. This oscillation is an effect of the lightly damped Dutch roll mode. The HBL simulation does not capture this effect, while JSBSim does. The effect drives phase errors between JSBSim and flight test which increases the calculated TIC values. While JSBSim appears, by inspection, to provide a clearly better model, TIC tells us that JSBSim is only marginally better (Table 26).

Table 26. Spiral Stability TIC

Parameter	JSBSim	HBL
Bank Angle (ϕ)	0.20	0.23

4.3.6 Open-Loop Flying Qualities (OLFQ) Summary

The results of the nominal OLFQ are shown in Table 27.

Table 27. Lateral-Directional and Longitudinal Mode Solution Comparison

Parameter	Variable	Flight Test	JSBSim	Error	HBL	Error
Short Period Natural Frequency (rad/sec)	ω_{sp}	3.15	2.90	-8%	3.40	+8%
Short Period Damping Ratio	ζ_{sp}	0.36	0.33	-0.03	0.39	+0.03
Phugoid Damping Ratio	ζ_{ph}	0.07	0.06	-0.01	0.05	-0.02
Phugoid Period (sec)	T_{ph}	73	66	-10%	71	-3%
Roll Mode Time Constant (sec)	τ_R	0.42	0.41	-0.01 sec	0.51	+0.09 sec
Dutch Roll Natural Frequency (rad/sec)	ω_{DR}	1.80	1.72	-4%	2.00	+11%
Dutch Roll Damping Ratio	ζ_{DR}	0.06	0.05	-0.01	0.05	-0.01
Dutch Roll Period (sec)	T_{dr}	3.43 sec	3.8 sec	+0.37 sec	3.13 sec	-0.30 sec
Spiral Mode	N/A	Stable	Stable	N/A	Stable	N/A

Table 27 indicates that, with the exception of the phugoid period, JSBSim performed as well or better than the proprietary HBL simulation. Using the TIC metric, JSBSim outperformed the HBL simulation in every time history analyzed (essentially equal with regards to short period). It was also noted in this section that both simulations met FAA criteria for each of the modes studied. The analysis of the OLFQ indicates that the development of a high-fidelity, proprietary, simulation model to predict nominal OLFQ is not required and can be predicted with a low-cost, open-source option, like JSBSim. Additionally, limitations were presented with respect to the TIC metric with regards to how bias and phase errors can drive up the value. The collection of nominal trim, off-nominal trim, and nominal OLFQ was designed to

support objective 2, outline in Chapter I, which was to “Calibrate the simulation to match nominal performance data collected during flight test. Compare the accuracy of the simulator during nominal aircraft performance.”

4.4 Performance Maneuver Case Studies

The baseline performance maneuver was used to evaluate predictive capability of the simulations in off-nominal configurations. Overall, eight configurations were studied. These configurations include nominal, rudder failed, ailerons failed, single engine failed, single engine and ailerons failed, single engine and rudder failed, ailerons and rudder failed, and elevator failed. For the single engine cases, using a consistent right hand turn, the results differed significantly depending on what engine was simulated failed. As a result, each turn direction was analyzed separately within the single engine sections, which essentially increases the total number of configurations to 11. Three climb rates (500, 1000, and 2000 fpm) were attempted with each configuration for a total of 33 case studies.

4.4.1 Data Presentation/Analysis

For each configuration at a single climb rate, three figures are presented: 1) control deflections, 2) body-axis rates, and 3) engine thrust (total or individual). For conciseness, only the data pertaining to the 500 fpm climb rate is presented in this chapter. A summary of results for 1000 fpm and 2000 fpm climb rates is contained at the end of each subsection, with additional supporting figures contained within Appendix F.

The first figure contains time histories of elevator-stabilizer deflection (δ_{es}), aileron deflection (δ_a), and rudder deflection (δ_r). Evaluation criteria is highlighted in the header of each figure. Of note, there is no evaluation criteria for aileron deflection

but it still displayed for completeness. Additionally, the aileron deflection bias (1.65°) has been removed from the flight test data.

The second figure contains time histories of the aircraft's body-axis rates (P, Q, and R). Since the rates are outputted in the body-axis, the values for pitch rate (Q) and yaw rate (R) are non-zero for the stabilized climbing spiral performance maneuver. The body-axis rates will only be shown in the nominal configuration in Chapter IV, the other figures will be presented in Appendix F.

The third figure contains thrust predictions for the two simulations compared with the calculated thrust from flight test, derived from EPR. A significant amount of high-frequency noise was present in the data using the pressure sensors in the engine to derive thrust from flight test. As a result, post-flight filtering of the thrust measurement using a low-pass finite impulse response (FIR) filter was accomplished using MATLAB's FILTER.m function (the passband and stopband frequencies were 4 and 7 rad/sec, respectively). In addition to the thrust filtering for flight test data, the evaluation criteria was relaxed to include the velocity tolerances for the performance maneuver. Since the velocity tolerance was 250 ± 10 KIAS, the region of acceptable thrust predictions was widened to include the lower and upper bound. Using the data gathered from the nominal trim section, the change in thrust measured from 260 and 240 KIAS was determined and added to the $\pm 5\%$ Net Thrust bounds described by the FAA metric. The resulting evaluation criteria for thrust becomes (Net Thrust $\pm 5\%$) ± 112 lbf.

4.4.2 Nominal

The performance maneuver was conducted using all available control surfaces (nominal condition) as a baseline for comparison between the other case studies. Results of the nominal case are shown in Figures 23, 24, and 25.

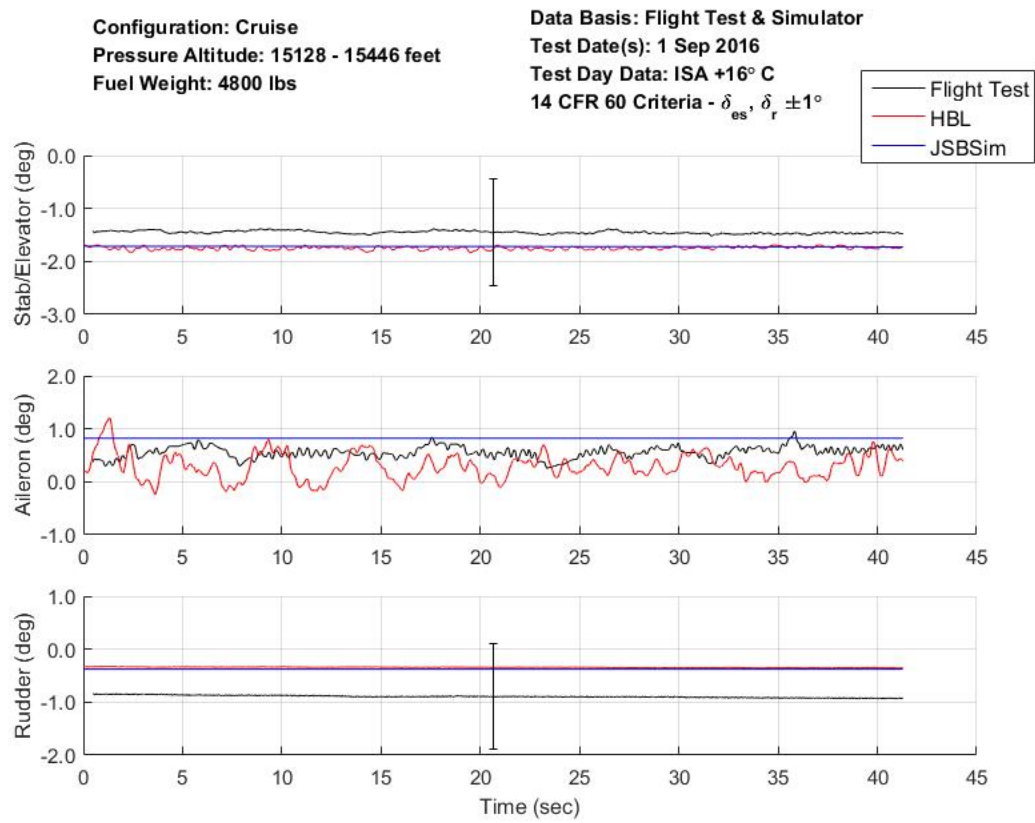


Figure 23. Nominal Configuration (500 fpm): Control Deflections

Figure 23 shows the control deflections required to maintain the stabilized climbing turn. Both simulations accurately predict control surface deflections.

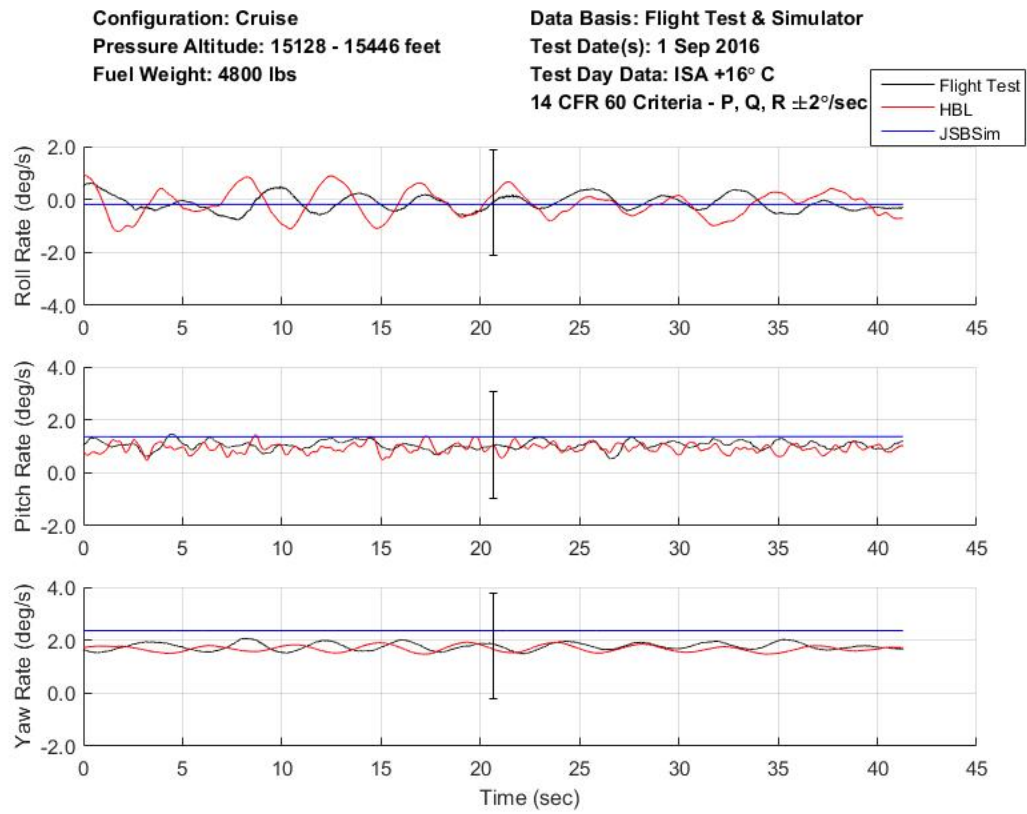


Figure 24. Nominal Configuration (500 fpm): Body-Axis Rates

Figure 24 show the roll, pitch, and yaw rates for the stabilized climbing spiral. Both simulations accurately predict the body-axis rates.

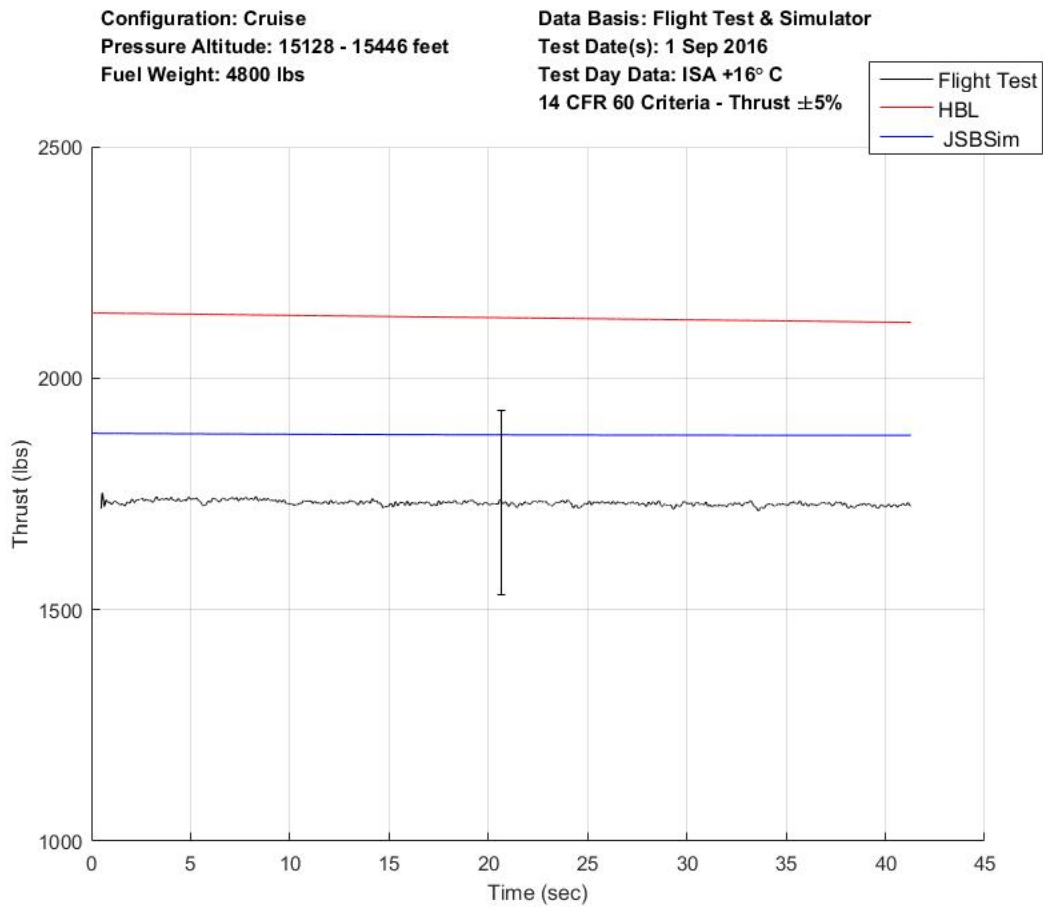


Figure 25. Nominal Configuration (500 fpm): Thrust

Figure 25 shows the total net thrust predictions of the two simulations compared to flight test. JSBSim falls within evaluation criteria while the HBL over-predicts thrust required.

Overall, JSBSim and HBL accurately predicted performance maneuver parameters with the exception of thrust for the HBL model. This maneuver was repeated at 1000 fpm and 2000 fpm climb rates. Supporting information for those climb rates is located in Appendix F: Figures F.1, F.2, F.3, F.4, F.5, and F.6. Data for the other two climb rates was consistent with the 500 fpm climb rate.

In the nominal configurations, TIC was calculated for meaningful parameters, the

control deflections and thrust and compared. Table 28 shows that JSBSim better predicted parameters.

Table 28. Nominal Configuration TIC (500 fpm)

Parameter	JSBSim	HBL
Elevator-Stabilizer Deflection (δ_{es})	0.085	0.094
Aileron Deflection (δ_a)	0.205	0.403
Rudder Deflection (δ_r)	0.407	0.446
Net Thrust	0.040	0.100

4.4.3 Rudder Failure

The rudder was simulated failed by setting the rudder command gain to zero with the VSS computer. The angle of sideslip (β) was controlled using differential throttle inputs. Data from the performance maneuver is shown in Figures 26, 28, and 27. Of note, since the throttle channel does not contain symmetric inputs in this case, both left (δ_{tl}) and right throttle (δ_{tr}) inputs are shown in Figure 28.

Post-flight analysis of the flight test data for 500 fpm and 2000 fpm climb rates showed a small rudder bias present, Figure F.7 and F.11. This rudder bias was in the direction of the turn which reduced the amount of split thrust required to maintain zero sideslip. The 1000 fpm climb rate showed no bias and therefore is analyzed in this section.

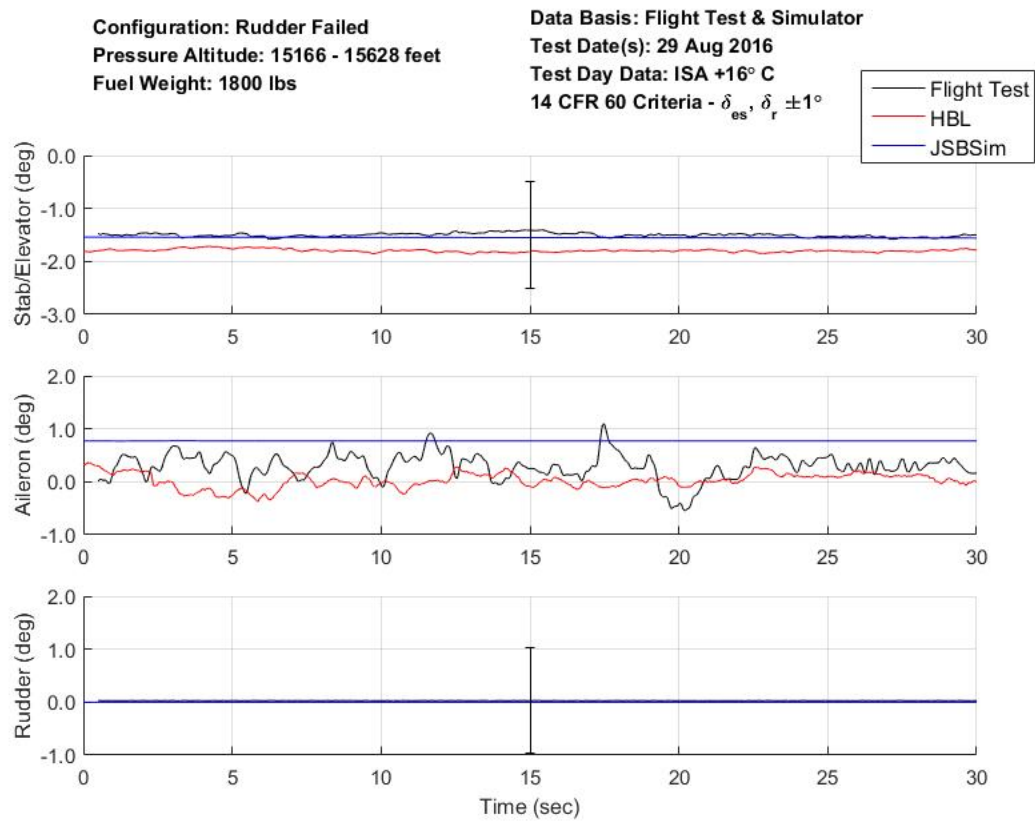


Figure 26. Rudder Failed Configuration (1000 fpm): Control Deflections

Figure 26 shows the control deflections required to maintain the stabilized climbing turn. Both simulations accurately predict control surface deflections.

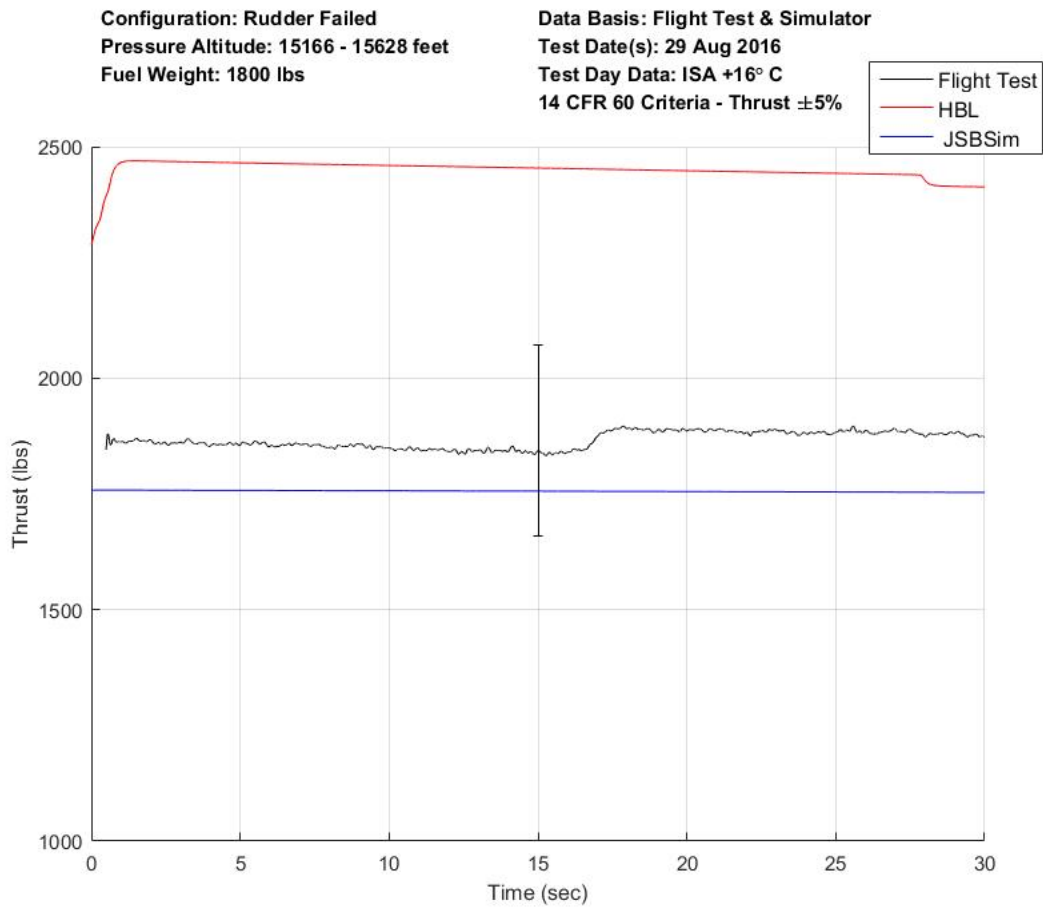


Figure 27. Rudder Failed Configuration (1000 fpm): Thrust

Figure 27 shows the total net thrust predictions of the two simulations compared to flight test. JSBSim falls within evaluation criteria while the HBL over-predicts thrust required.

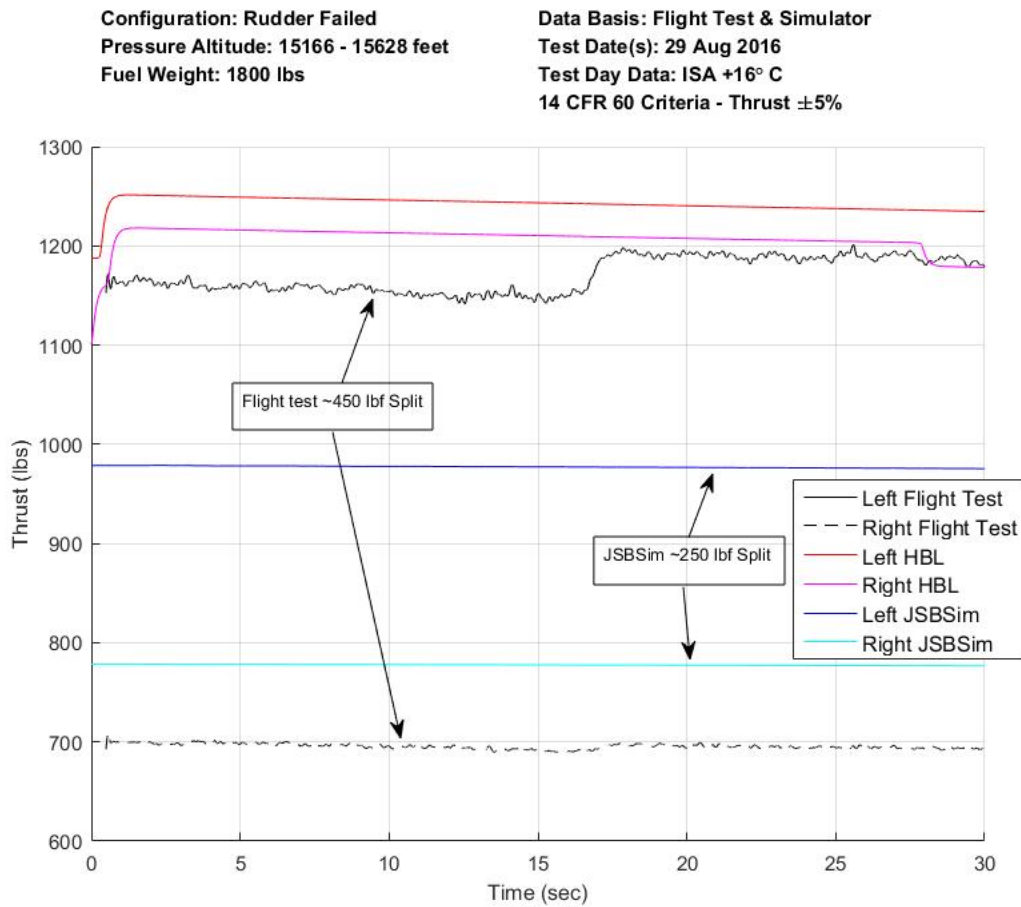


Figure 28. Rudder Failed Configuration (1000 fpm): Individual Engine Thrust

Figure 28 shows the split thrust predictions of the two simulations compared to flight test. Flight test data shows a 450 lbf difference required to maintain zero sideslip while JSBSim shows a 200 lbf difference. In this case JSBSim under-predicts the split throttle required to zeroize the sideslip.

An analysis of the angle of sideslip time history was conducted to determine if the pilots were “over-driving” the desired sideslip with their throttle inputs. Figure 29 shows the measured angle of sideslip for the three data sources for 1000 fpm, in the rudder failed configuration.

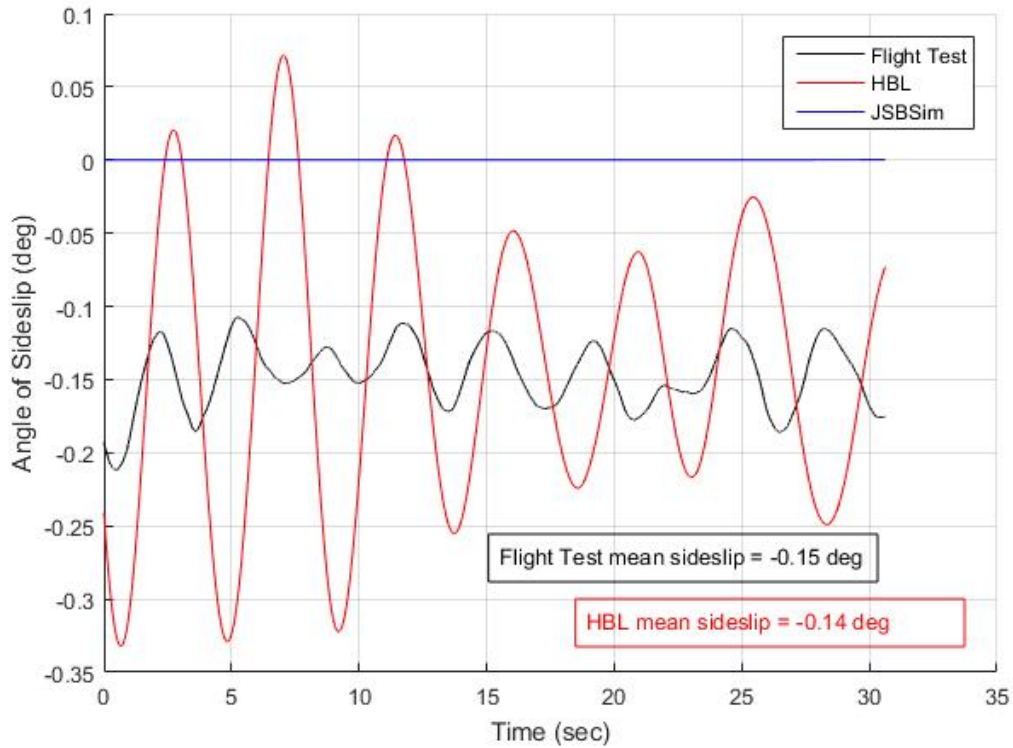


Figure 29. Rudder Failed Configuration (1000 fpm): Angle of Sideslip

Figure 29 shows a mean sideslip of -0.15° in flight test and -0.14° in HBL, with perfectly tracking sideslip using JSBSim. This indicates that the pilot was, in fact, over-driving the desired sideslip, but since their ability to read the instrument was $\approx \pm 2^\circ$, this deviation went unnoticed. The sideslip oscillations are due to the low damping Dutch roll mode. To determine whether this was a significant amount of deflection, JSBSim was run with commanded sideslip angles of -0.15° , -0.5° , and -1.0° . The resulting split thrust deltas were 335 lbf for -0.15° , 469 lbf for -0.5° , and 731 lbf for -1.0° . This shows how very little sideslip results from a split thrust situation. Unfortunately, this fact was discovered post-flight, therefore, an analysis of sideslip effects due to varying split thrust was not conducted in flight test to compare to the simulation result. A common test technique, called “ V_{MCA} determination” would help to answer this question (further discussed in Chapter V).

In the rudder failed configuration, TIC was calculated for meaningful parameters, the control deflections and individual thrust and compared. Table 29 shows that JSBSim better predicted parameters.

Table 29. Rudder Failed Configuration TIC (1000 fpm)

Parameter	JSBSim	HBL
Elevator-Stabilizer Deflection (δ_{es})	0.0197	0.0907
Aileron Deflection (δ_a)	0.4683	0.7629
Left Engine Thrust	0.0226	0.0313
Right Engine Thrust	0.1955	0.2693

Additional analysis was conducted using JSBSim at 500 fpm and 2000 fpm climb rates with the rudder failed at a non-zero position. The non-zero position was 0.4935° , the same as the bias present in flight test data. With this fixed rudder position, JSBSim showed that a ≈ 40 lbf difference in throttle position was necessary to zeroize the sideslip. This result agrees with the 500 fpm and 2000 fpm result shown in Figures F.10 and F.14, in Appendix F.

4.4.4 Aileron Failure

The ailerons were simulated failed by setting the aileron command gain to zero within the VSS computer. The roll angle (ϕ) was controlled using rudder inputs. Resulting angle of sideslip (β) was ignored from the performance maneuver tolerances. Data from the performance maneuver is shown in Figures 30, 31, and 32.

Upon inspection of the flight test data, a small aileron bias was found to be present after the failure mode was implemented.

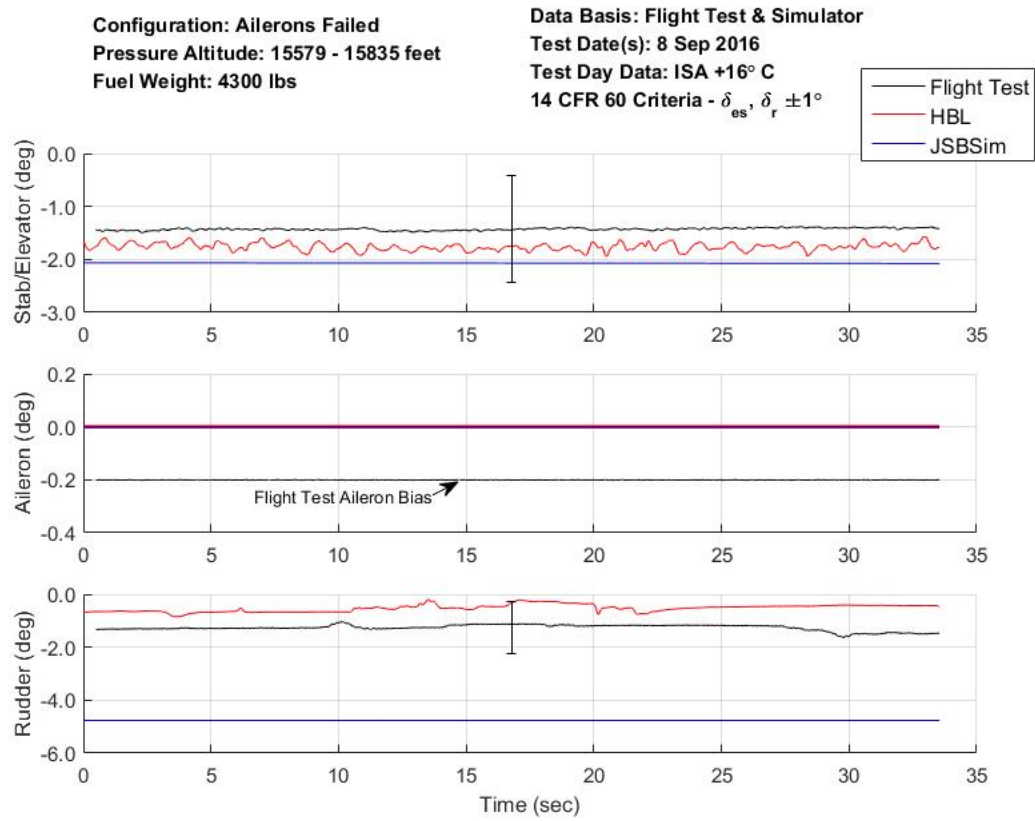


Figure 30. Ailerons Failed Configuration (500 fpm): Control Deflections (Bias Present)

Figure 30 shows the small aileron bias of -0.2° present in the flight test data. JSBSim and HBL both have a 0° aileron deflection. JSBSim in this case has a much larger rudder deflection predicted to achieve 30° of bank. To make the comparison more accurate, the bias present in the flight test data was inserted into JSBSim, in addition to the 1.65° aerodynamic bias. Figure 31 shows the effect of these biases.

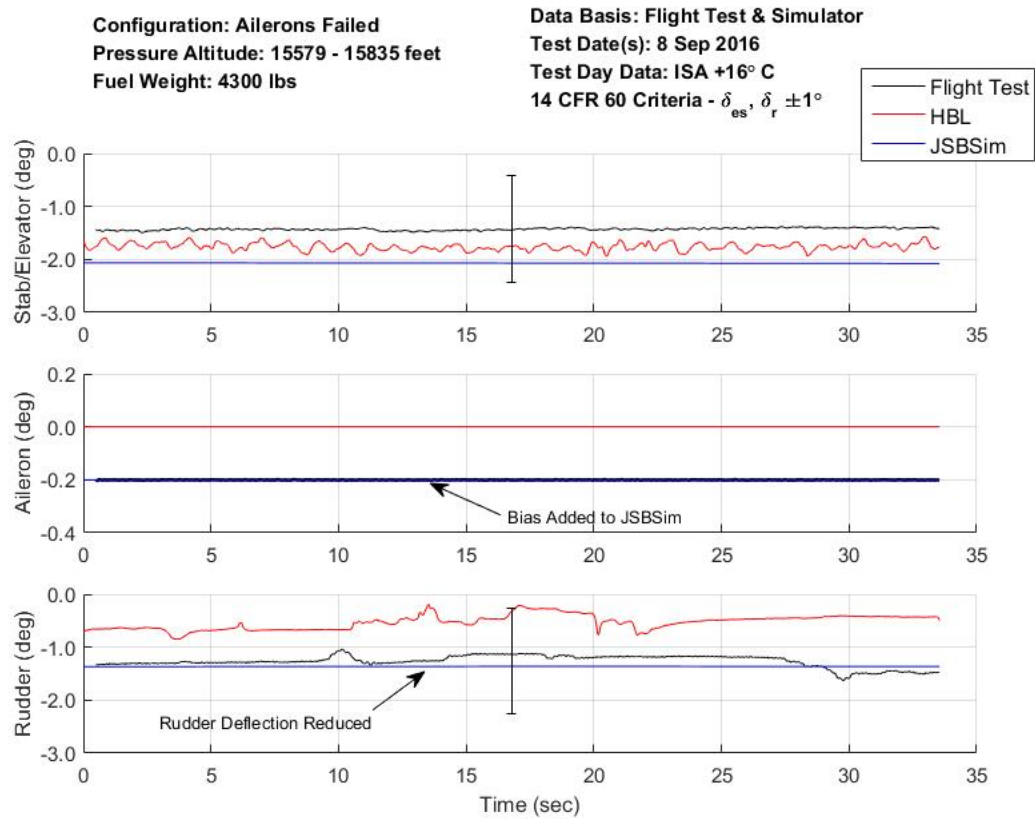


Figure 31. Ailerons Failed Configuration (500 fpm): Control Deflections (Bias Removed)

Figure 31 inserts the -0.2° aileron deflection into JSBSim and shows that the bias reduces the rudder deflection required to achieved 30° of bank. One can see that JSBSim meets evaluation criteria with this bias added. The HBL had no functionality to add a non-zero bias to a specific control surface therefore the comparison to flight test and JSBSim is invalid in this case.

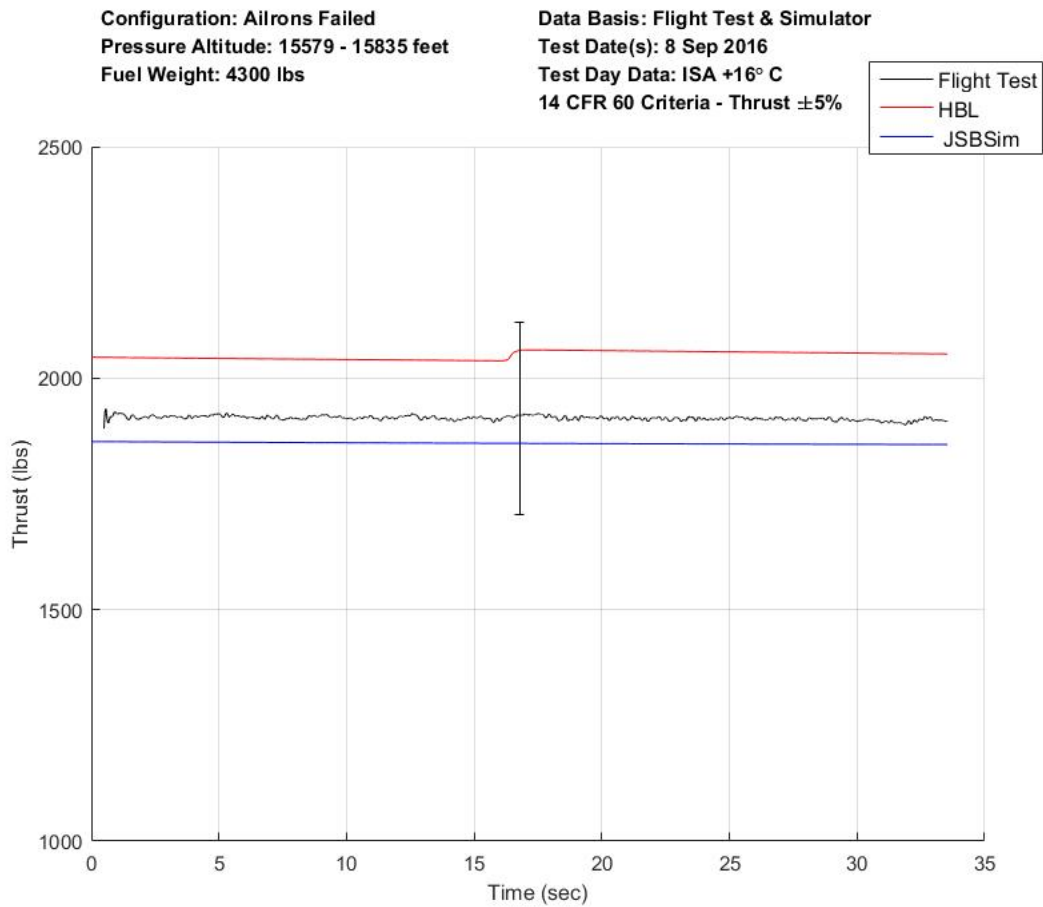


Figure 32. Ailerons Failed Configuration (500 fpm): Thrust

Figure 32 shows that JSBSim meets evaluation criteria for thrust. Additionally, results from 1000 and 2000 fpm climb rates are shown in Appendix F, Figures F.15, F.16, F.17, F.18, F.19, F.20, and F.21. Analysis of the 1000 and 2000 fpm climb rates showed similar results as 500 fpm, with an aileron bias of $\approx -0.20^\circ$. The bias has been removed in Figures F.16 and F.19.

A comparison using TIC would have little meaning in this case since HBL was unable to include a bias. For this reason, TIC was not calculated for this case. Overall, JSBSim provided a good match to flight test data. Matching initial conditions was

difficult in this case as rudder effectiveness is extremely sensitive to the x c.g. position (longitudinal) the stability derivative $C_{n\beta}$. Very small changes in the x c.g. position (aft) resulted in directional instabilities, and thus smaller rudder deflections required to roll the aircraft.

4.4.5 Engine Failure

The performance maneuver was conducted with right engine at IDLE (simulated failed) in both left and right hand directions. Two different results occurred since the engines are not mounted on the aircraft center-line. Both JSBSim and the HBL predicted that the aircraft would not have enough specific excess power (P_s) to complete the performance maneuver at 2000 fpm. This result was validated by the flight test, therefore only 500 fpm and 1000 fpm climb rates are discussed.

4.4.5.1 Right Engine Idle - Right-Hand Turn

Results from the performance maneuver attempted with the right engine at idle thrust are shown in Figures 33 and 34.

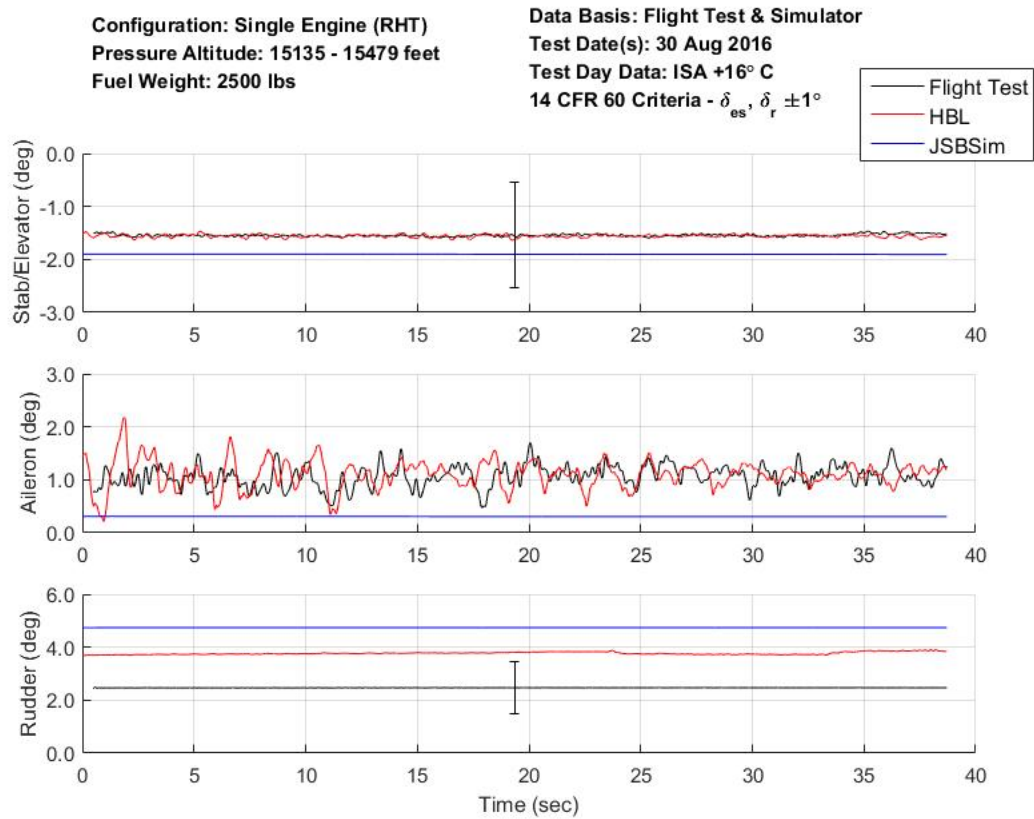


Figure 33. Single Engine Right-Hand Turn into Idle Engine (500 fpm): Control Deflections

Figure 33 shows the control deflections required maintain the performance maneuver parameters in a right-hand turn, single engine configuration (into the idle engine). In this case, the left engine is used to control airspeed, aileron is used to control roll angle, and rudder used to control sideslip. One can also see that JSBSim over-predicts rudder deflection required and under-predicts aileron deflection required. Rudder deflection required is outside evaluation criteria for both simulations in this case. The reason that the simulations have a difficult time matching flight test data is due to the coupled nature of the control surfaces. For example, both rudder and aileron produce

roll moments and yaw moments. Additionally, in a single engine configuration, the split thrust produces both a yaw and roll moment. It is difficult to isolate a single variable as the cause of the simulations falling outside evaluation criteria. As a result of the coupled nature of the three controls (aileron, rudder, and engine), it is possible that multiple solutions exist to achieve a stabilized climbing spiral, as required by the performance maneuver.

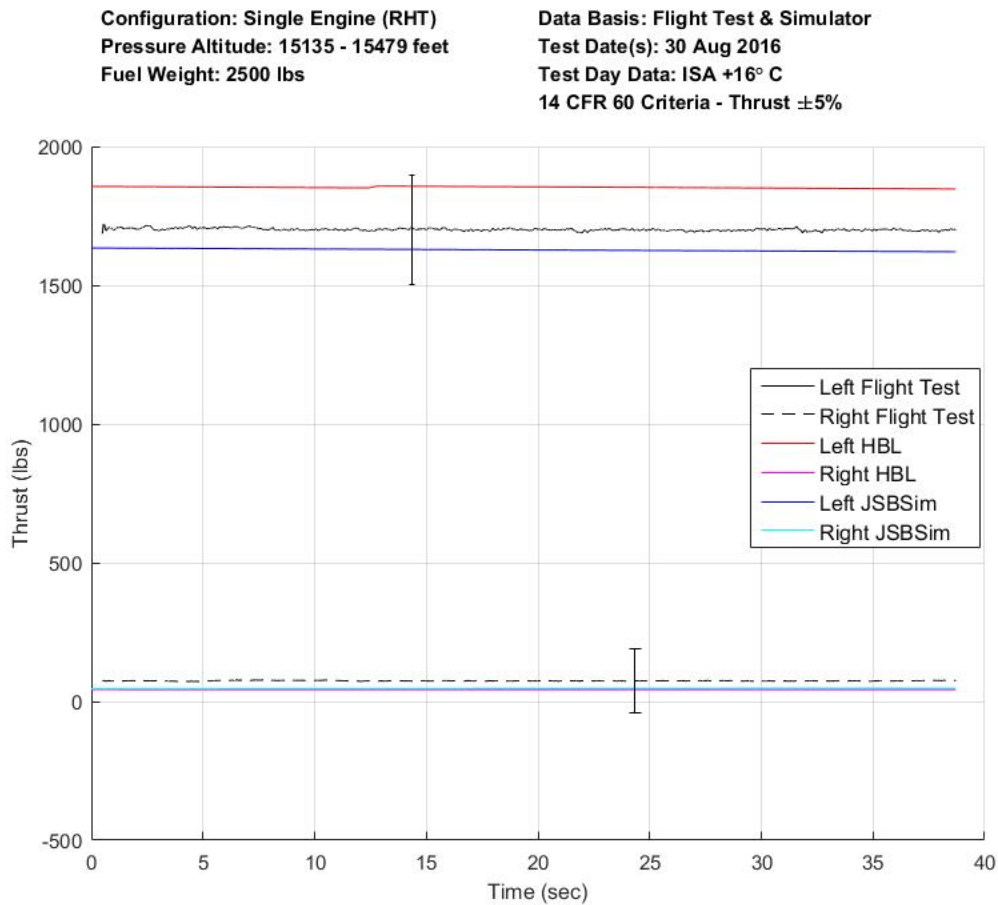


Figure 34. Single Engine Right-Hand Turn into Idle Engine (500 fpm): Split Thrust

Figure 34 shows the individual engine thrust predictions in the single engine configuration. Both simulations fall within evaluation criteria for thrust.

4.4.5.2 Right Engine Idle - Left-Hand Turn

Results from the performance maneuver attempted with the right engine at idle thrust are shown in Figures 35 and 36.

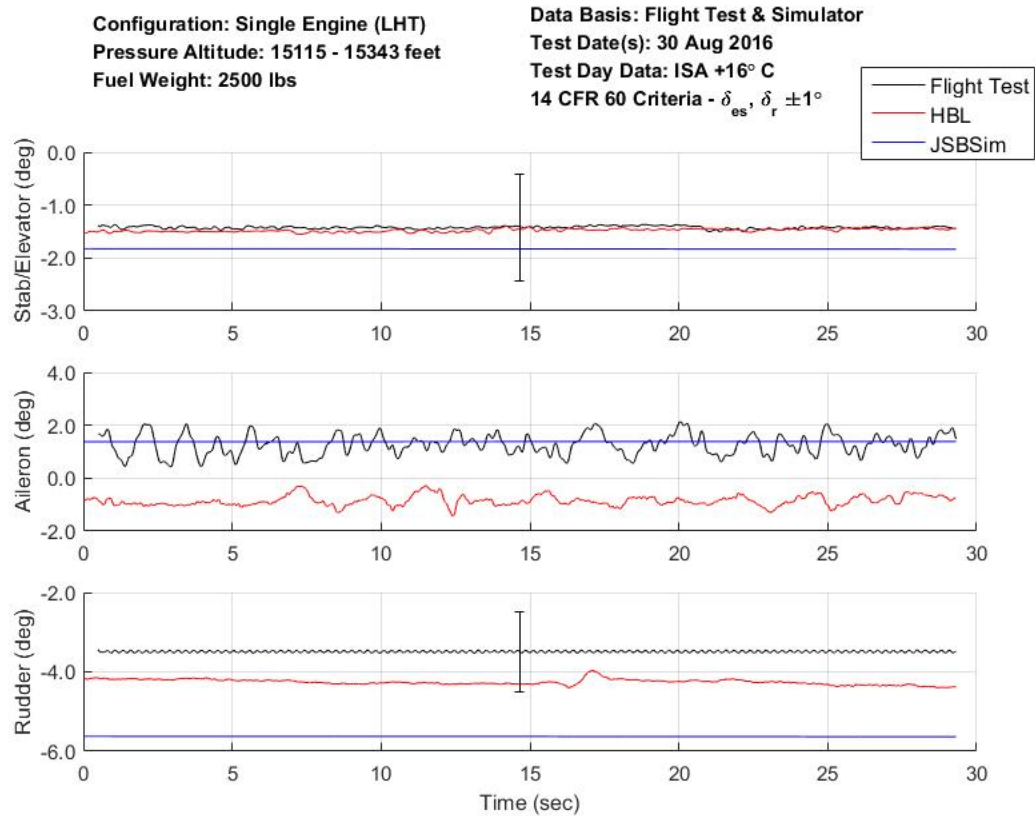


Figure 35. Single Engine Left-Hand Turn away from Idle Engine (500 fpm): Control Deflections

Figure 35 shows the control deflections required maintain the performance maneuver parameters in a left-hand turn, single engine configuration (away from the idle engine). Results are consistent with the right-hand turn case (500 fpm), which shows that JSBSim over-estimates rudder deflection required to zeroize sideslip.

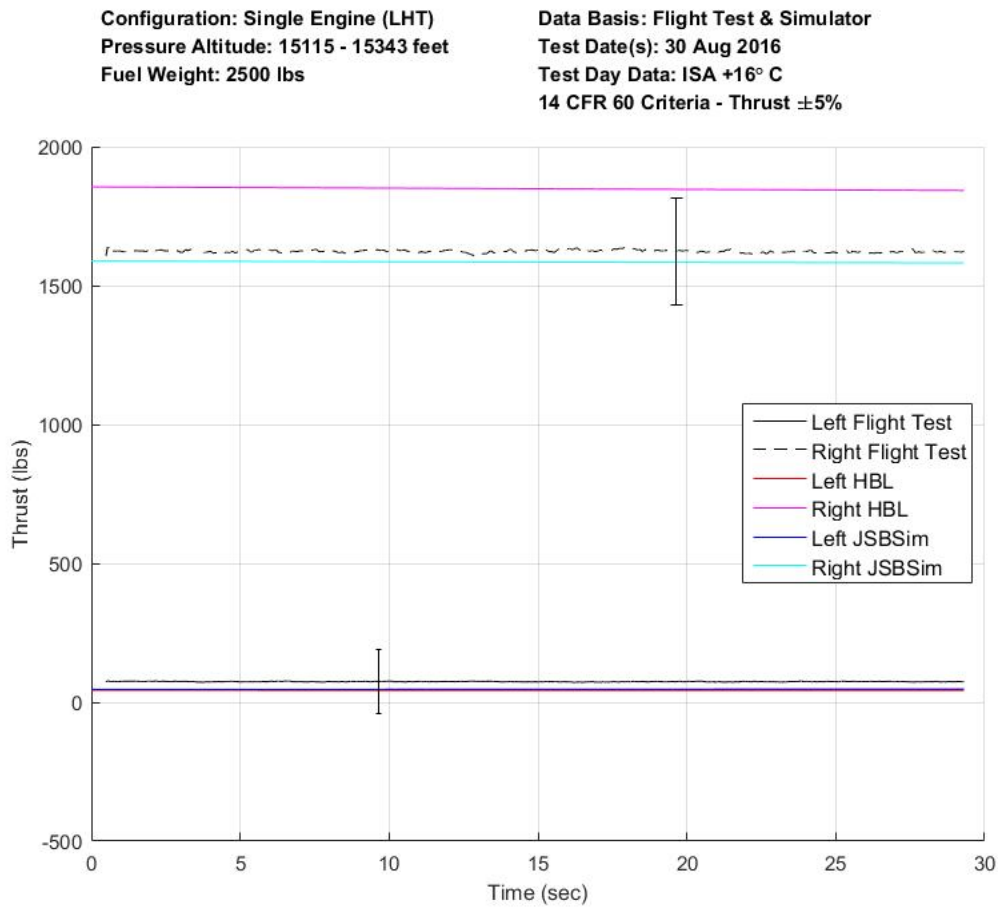


Figure 36. Single Engine Left-Hand Turn away from Idle Engine (500 fpm): Split Thrust

Figure 36 shows the individual engine thrust predictions in the single engine configuration. Both simulations fall within evaluation criteria for thrust.

The results of the 1000 fpm climb rate are contained within Appendix F, Figures F.22, F.23, F.24, F.25, F.26, F.27, F.28, and F.29. The additional climb rates show consistent results with the 500 fpm climb rate.

Overall, the errors associated with the prediction of the single engine performance maneuver is driven by the highly coupled nature of the flight controls and engines. Each of which, produce moments with either complement or negate each other. In a

general sense, both simulators do predict maneuver feasibility and accurately predict thrust required to maintain the climb, but fall outside evaluation criteria for control surface deflections. A TIC evaluation was completed for both turn directions and is summarized in Table 30. The HBL in these cases tended to better predict overall performance, but the numbers are not conclusive.

Table 30. Single Engine TIC (500 fpm)

Parameter	JSBSim (RHT)	HBL (RHT)	JSBSim (LHT)	HBL (LHT)
Elevator-Stabilizer Deflection (δ_{es})	0.1043	0.0123	0.1248	0.0223
Aileron Deflection (δ_a)	0.5654	0.1486	0.1484	0.9859
Rudder Deflection (δ_r)	0.3150	0.2089	0.2346	0.0995
Left Engine Thrust	0.0222	0.0427	0.2262	0.2680
Right Engine Thrust	0.2230	0.2702	0.0122	0.0649

4.4.6 Engine and Aileron Failure

The performance maneuver was conducted with the right engine at IDLE (simulated failed) and the aileron commands zeroized. Roll angle was controlled using rudder input while angle of sideslip was ignored. Like the other single engine cases, 2000 fpm climb rate was not possible, due to lack of P_s , but was accurately predicted by the simulations.

4.4.6.1 Right Engine Idle - Right-Hand Turn

Upon inspection of the 500 fpm climb rate, a non-zero aileron deflection bias was present within the flight test data of 0.35° . An aileron deflection bias of 2.47° was found within the 1000 fpm right-hand turn (RHT) data. This bias was added into JSBSim and the simulations re-run to attempt to match initial conditions as closely as possible. The HBL had no ability to add a non-zero deflection but it still shown

in the figures for completeness. This bias still meets the intent of a the research since it is possible that the ailerons could be stuck with a non-zero deflection. Figures 37, 38, and F.30 show the results of the 500 fpm climb.

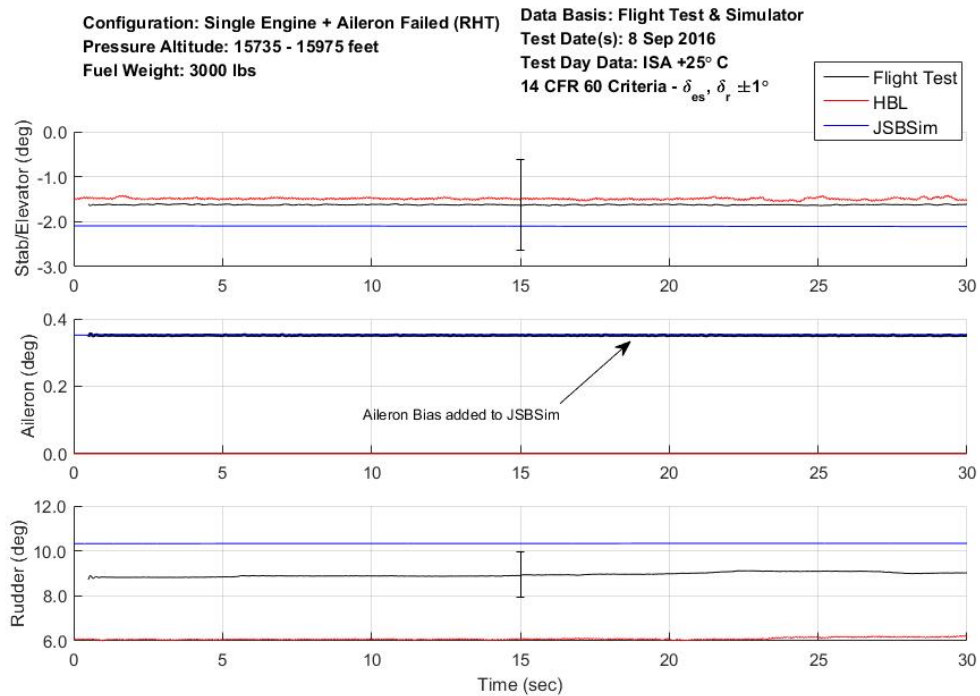


Figure 37. Single Engine and Aileron Failed (RHT) at 500 fpm: Deflections

Figure 37 shows the surface deflections required to maintain the performance maneuver. A bias was added to JSBSim which increased the rudder deflection required to roll the aircraft. JSBSim exceeds evaluation criteria for rudder deflection in this case.

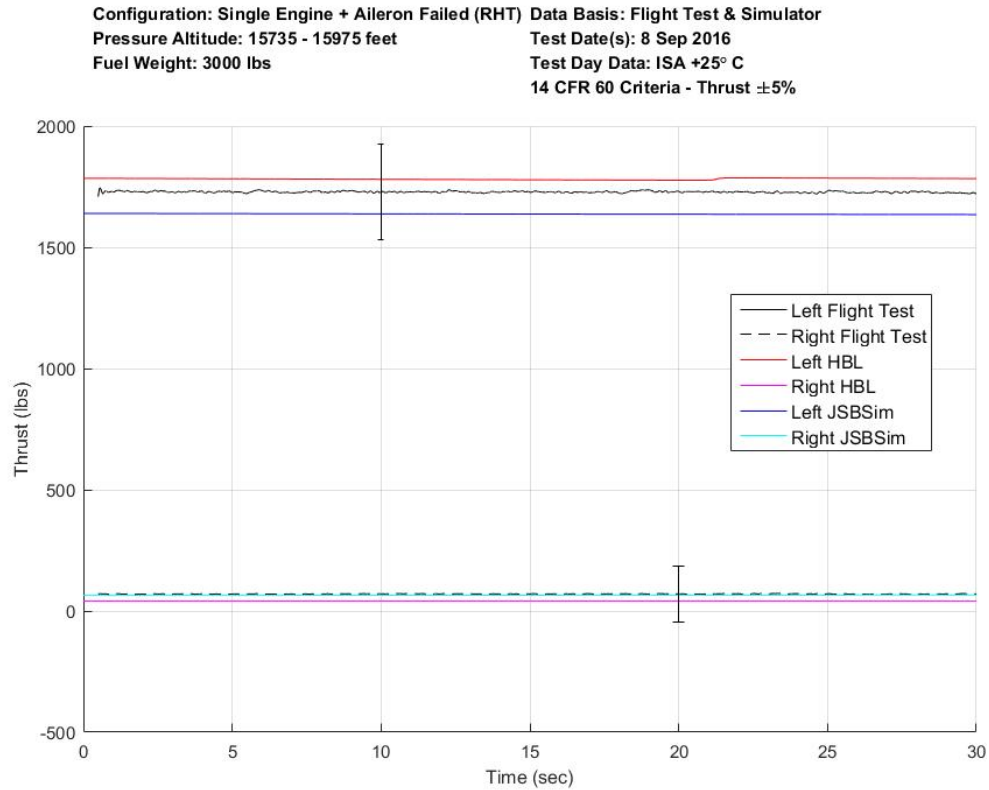


Figure 38. Single Engine and Aileron Failed (RHT) at 500 fpm: Split Thrust

Figure 38 shows the individual engine thrust predictions in the single engine configuration. Both simulations fall within evaluation criteria for thrust. At the 1000 fpm climb rate, the rudder deflection is slightly less due to a reduction in weight and shift in center-of-gravity but mostly due to the increase aileron bias in the flight test data; this can be seen in Figure F.32. The 1000 fpm climb rate right-hand turn data is displayed in Appendix F, Figures F.32, F.33, and F.34.

4.4.6.2 Right Engine Idle - Left-Hand Turn

With the left-hand turn maneuver, the bias present in the flight test data was 0.63° for both 500 fpm and 1000 fpm climb rates. These biases were matched using

JSBSim. Figure 39, F.37, and F.31 show the results.

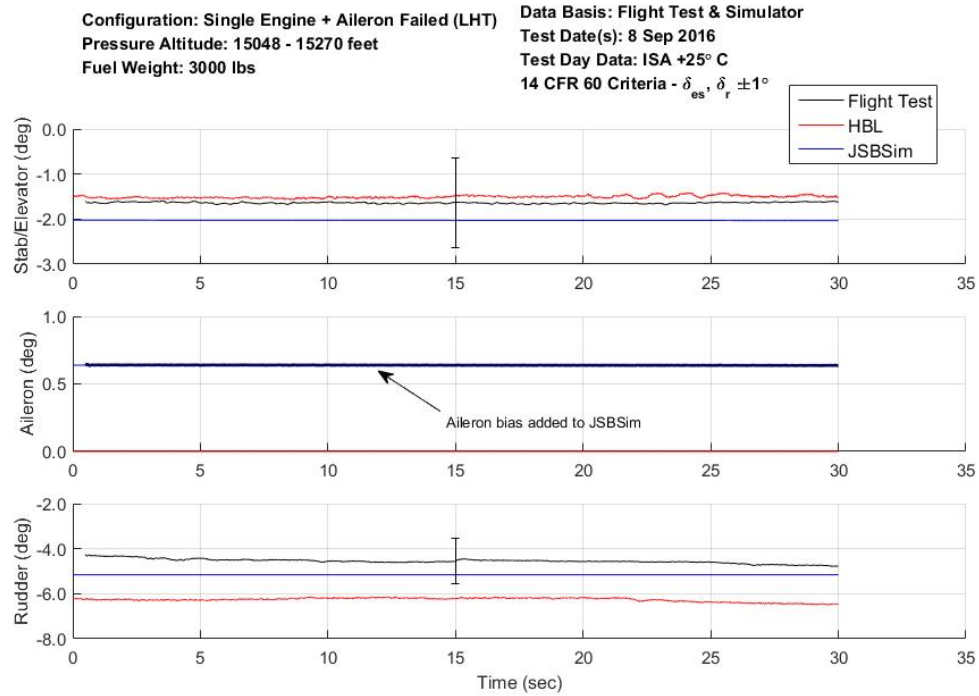


Figure 39. Single Engine and Aileron Failed (LHT) at 500 fpm: Control Deflections

Figure 39 shows the surface deflections required to maintain the performance maneuver. A bias was added to JSBSim which decreased the rudder deflection required to roll the aircraft. JSBSim met evaluation criteria for rudder deflection for both 500 and 1000 fpm climb rates (Figure F.35).

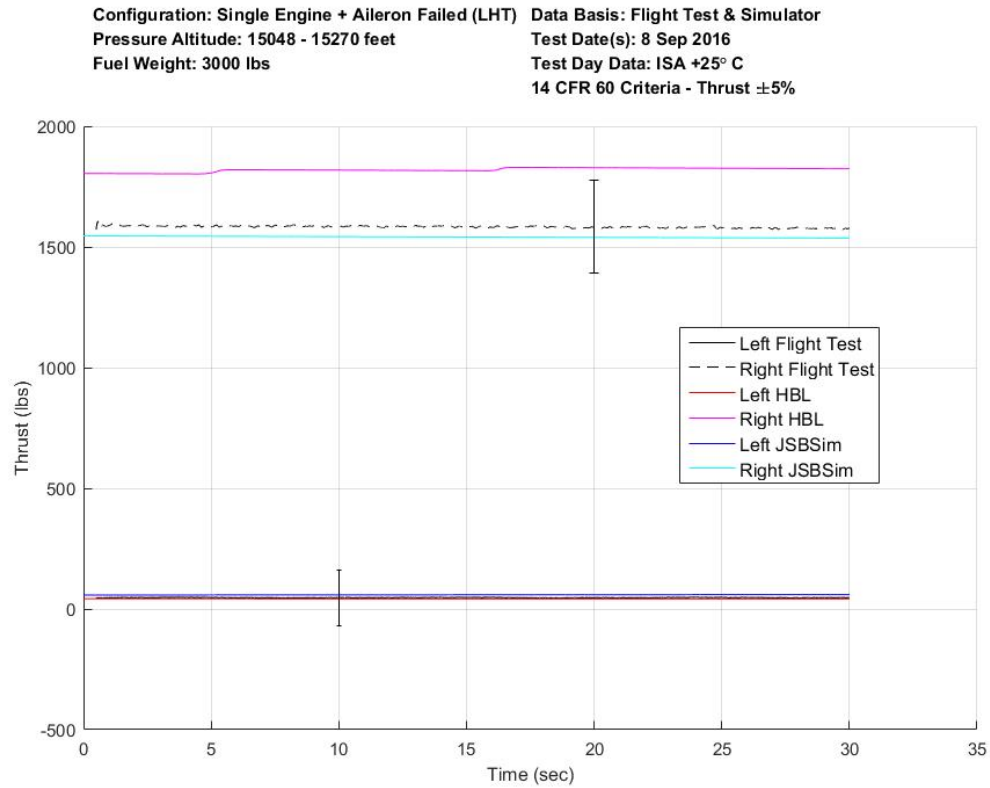


Figure 40. Single Engine and Aileron Failed (LHT) at 500 fpm: Split Thrust

Figure F.37 shows the individual engine thrust predictions in the single engine configuration. Both simulations fall within evaluation criteria for thrust.

The 1000 fpm climb rate right-hand turn data is displayed in Appendix F, Figures F.32, F.33, and F.34.

Overall, for 75% of the cases studied with a single engine and the ailerons failed, JSBSim met evaluation criteria for control surface deflections. The only exceedance from criteria occurred during the the right-hand turn 500 fpm climb rate for rudder deflection. JSBSim predicted a rudder deflection outside evaluation criteria by $\approx 0.39^\circ$, which was considered a very small deviation. TIC was not calculated for this configuration due to the inability of inserting an aileron bias into the HBL simulation.

4.4.7 Engine and Rudder Failure

The performance maneuver was conducted with the right engine at IDLE (simulated failed) and the rudder commands zeroized. Roll angle was controlled using aileron input while angle of sideslip was ignored. Like the other single engine cases, 2000 fpm climb rate was not possible but was accurately predicted by the simulations.

4.4.7.1 Right Engine Idle - Right-Hand Turn

Analysis of the flight test data showed a bias of -0.33° in the rudder for 500 fpm and -0.49° for 1000 fpm. The effect of this rudder bias was minimal but was still included as a bias in the JSBSim results. Figures 41, 42, and F.38 (Appendix F) show the results of the 500 fpm climb case.

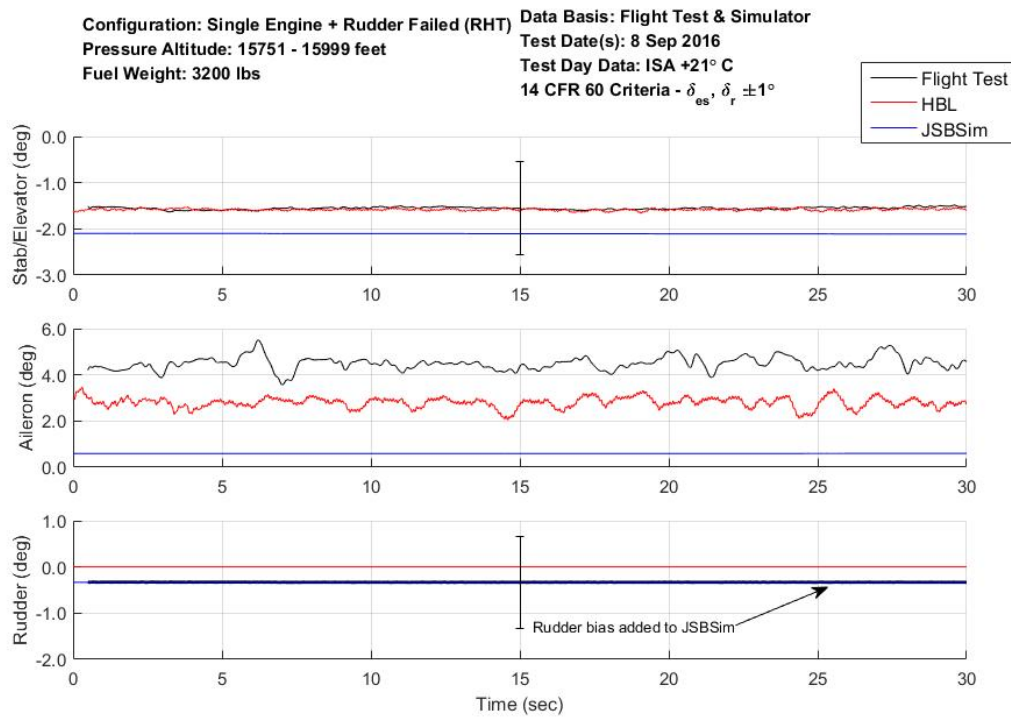


Figure 41. Single Engine and Rudder Failed (RHT) at 500 fpm: Control Deflections

Figure 41 shows the surface deflections required to maintain the performance maneuver. In this case, JSBSim predicts that the effect of the single engine producing thrust on the left side of the aircraft while rolling right would produce a larger rolling moment, thus less of an aileron deflection needed. The results of the 1000 fpm climb match this result, Figure F.40.

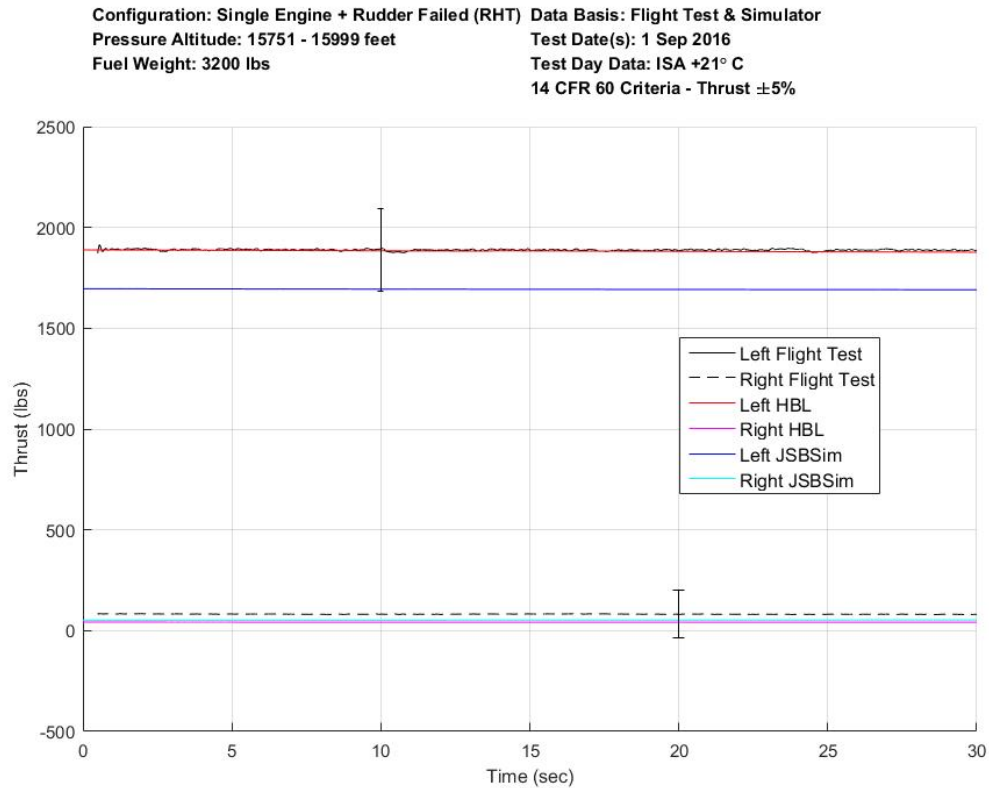


Figure 42. Single Engine and Rudder Failed (RHT) at 500 fpm: Split Thrust

Figure 42 shows the individual engine thrust predictions in the single engine configuration. Both simulations fall within evaluation criteria for thrust. The additional climb rate data is contained in Appendix F, Figures F.40, F.41, and F.42.

4.4.7.2 Right Engine Idle - Left-Hand Turn

Analysis of the flight test data showed a bias of -0.5° in the rudder for both 500 fpm and 1000 fpm climb rates. The rudder bias was included as in the JSBSim results. Figures 43, 44, and F.39 show the results of the 500 fpm climb case.

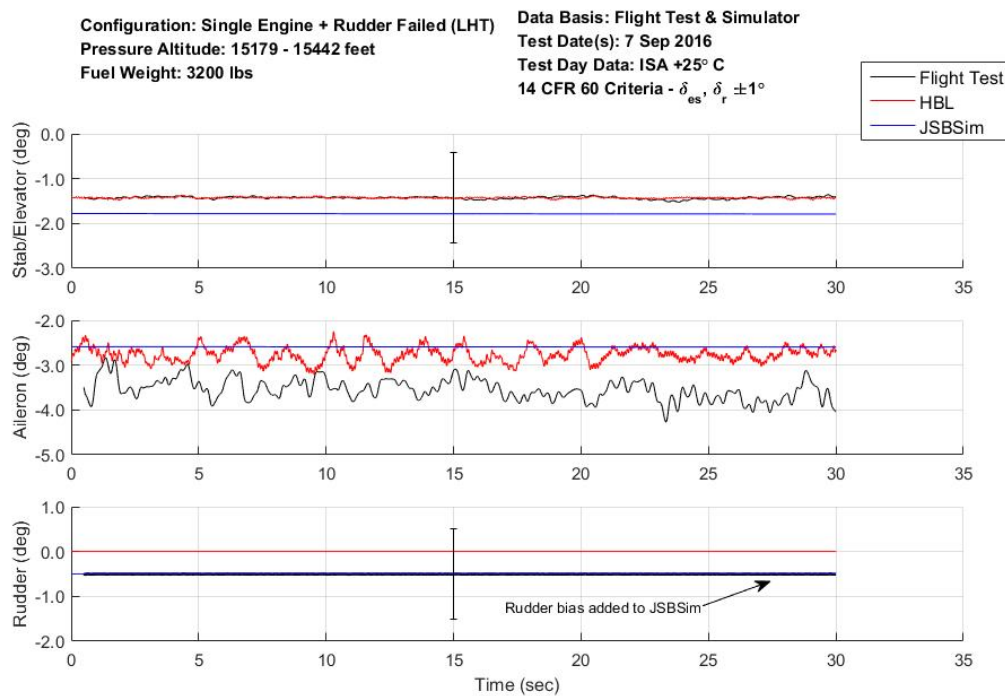


Figure 43. Single Engine and Rudder Failed (LHT) at 500 fpm: Control Deflections

Figure 43 shows the surface deflections required to maintain the performance maneuver. When rolling into the functioning engine, JSBSim matches flight test data better than the right-hand turn case. However, flight test data shows a fairly symmetric response in aileron deflection between right and left hand turns, indicating that the effect of the single engine has only a very small contribution to a rolling moment. JSBSim over estimates the contribution in both cases. The results of the 1000 fpm climb match this result, Figure F.43.

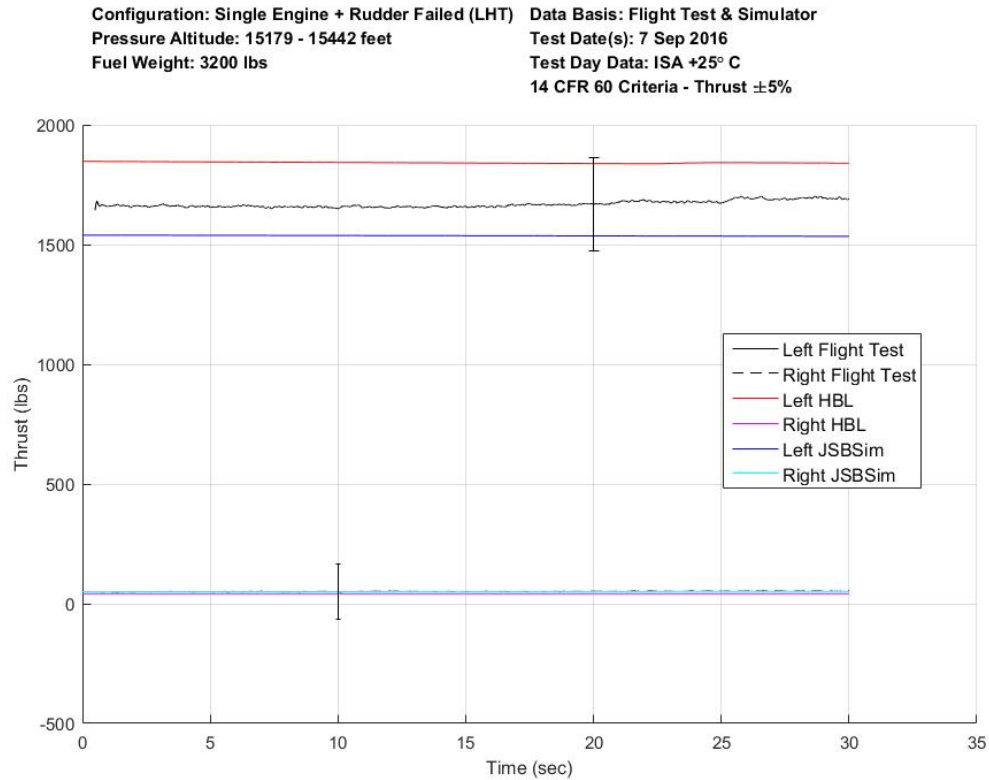


Figure 44. Single Engine and Rudder Failed (LHT) at 500 fpm: Split Thrust

Figure 44 shows the individual engine thrust predictions in the single engine configuration. Both simulations fall within evaluation criteria for thrust. The additional climb rate data is contained in Appendix F, Figures F.43, F.44, and F.45.

Overall, in the single engine and rudder failed case, JSBSim was found to overestimate the effect of a single engine on the roll performance of the aircraft. In reality, the rolling moment was very small from the flight test data as observed by the measured aileron deflection in both turn directions.

4.4.8 Aileron and Rudder Failure

The performance maneuver was conducted with both the rudder and ailerons zeroized. Roll angle was controlled using differential throttle. Figures 45, F.46, and

46 show the results of the 500 fpm climb case.

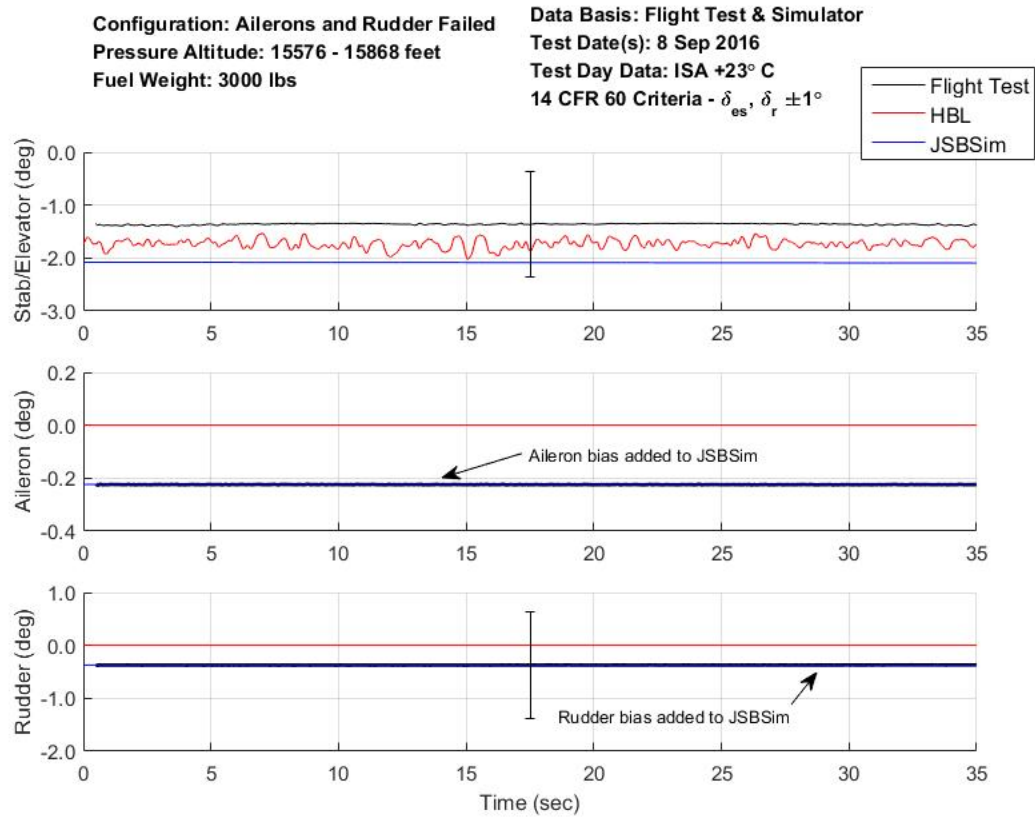


Figure 45. Ailerons and Rudder Failed at 500 fpm: Control Deflections

Figure 45 shows control deflections and highlights a bias in both aileron and rudder deflection within the flight test data. These biases were -0.37° for rudder and -0.22° for ailerons. For 1000 fpm, the biases were essentially the same, while for 2000 fpm climb the aileron bias was 1.22° , which significantly lessened the engine split required to hold 30° of bank. These biases were matched using JSBSim in each case.

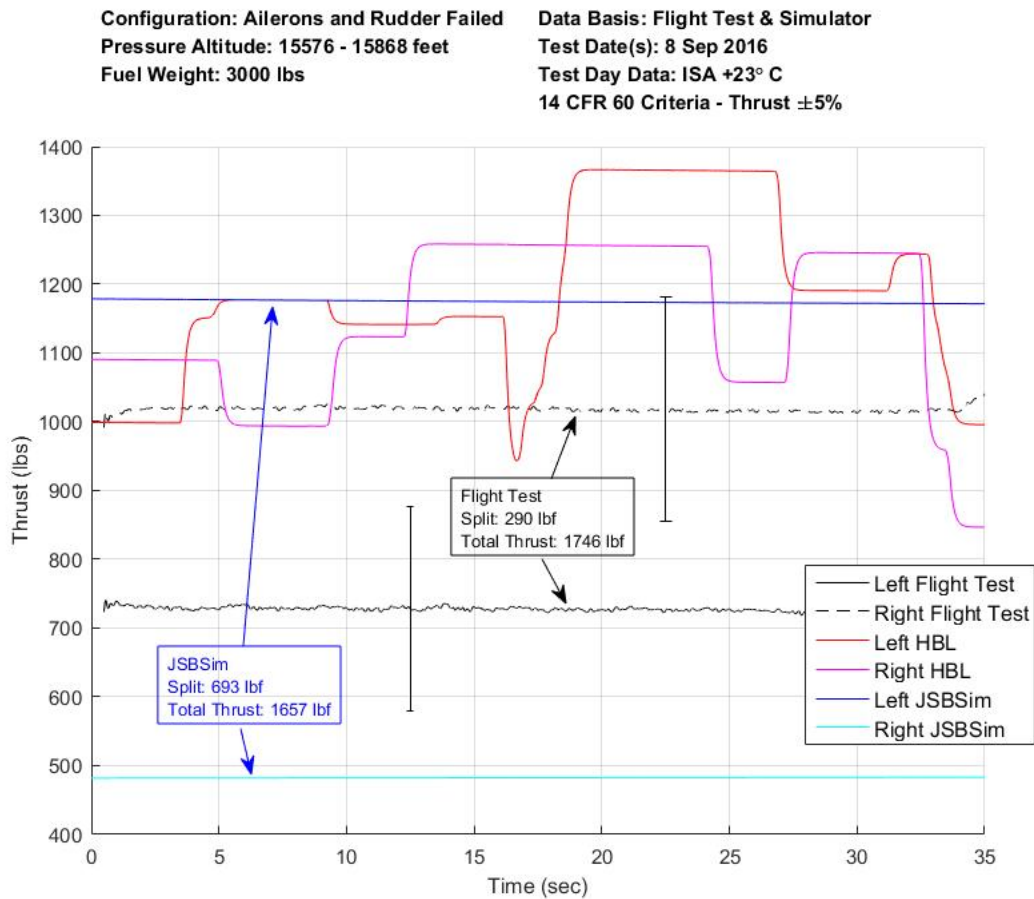


Figure 46. Ailerons and Rudder Failed at 500 fpm: Individual Engine Thrust

Figure 46 shows the split thrust required to hold the desired bank angle. For both 500 fpm and 1000 fpm (Figure F.49), JSBSim predicts a larger engine split required to roll the aircraft, but is consistent at ≈ 670 lbf from 500 to 1000 fpm since the biases are roughly the same. Overall thrust met evaluation criteria. The large bias present in aileron deflection for the 2000 fpm case lessened the required split with flight test predicting a split of 173 lbf while JSBSim predicted 332 lbf. This over-prediction in all three cases further suggests that JSBSim has a difficult time estimating the forces and moments generated by an asymmetric thrust situation.

Supporting data for 1000 and 2000 fpm climb rates are contained within Appendix

F, Figures F.47, F.48, F.49, F.50 ,F.51, and F.52.

4.4.9 Elevator Failure

The elevator was failed using the VSS interface with the horizontal-stabilizer at a fixed position. Control of aircraft pitch attitude and airspeed was attempted through the use of the throttles. Both analytical (JSBSim) and MITL (HBL) simulations predicted the baseline performance maneuver would not be possible given the relative location of the thrust line on the Learjet 25D. Since the engines are mounted above the c.g. an increase in thrust produced a pitch-down moment, while a decrease caused the aircraft to pitch-up. A variety of techniques were attempted (i.e. using rudder to control pitch from a small bank angle and using very slow input throttle increase) without success. A limited evaluation was conducted in-flight by attempting to capture a small pitch attitude at a constant airspeed. This limited evaluation was unsuccessful. The simulations were both judged to accurately predict that the maneuver was not feasible with an elevator failed.

4.5 Chapter Summary

In this chapter, the results were divided into three sections: trim performance which included nominal and off-nominal configurations, OLFQ, and off-nominal performance maneuver configurations. Data presentation, evaluation criteria, and the TIC metric was discussed. Due to various biases present in the flight test data, TIC was only calculated when a comparison between JSBSim and the HBL was appropriate. Chapter V presents a summary of the overall results, conclusions, and recommendations for future work.

V. Conclusions and Recommendations

As was discussed in Chapter I there is a desire to validate a simulation solution for a known aircraft in off-nominal configurations. This validation would contribute to a multistate approach to the design of robust aircraft, capable of overcoming any number of failures while still completing the intended mission. A literature review was conducted showing that a methodology for verifying and validating a simulation in off-nominal aircraft configurations was lacking. Because of this research shortfall, the objectives of this research were to:

1. Compile and condense previously collected data from a system ID process on a test aircraft and build a high-fidelity simulation model of that aircraft.
2. Calibrate the simulation to match nominal performance data collected during flight test. Compare the accuracy of the simulator during nominal aircraft performance.
3. Induce off-nominal aircraft states and perform flight test maneuvers, collect aerodynamic and performance data, and compare with the performance predicted by the simulation software.
4. Determine the degree of accuracy with which the software predicted performance.
5. Determine if changes to the baseline model can or need to be made to increase off-nominal accuracy.

The first objective was completed using an open-source flight dynamics model (FDM) (JSBSim) coupled with data collected on previous parameter ID missions by the Calspan Corporation and the USAF Test Pilot School. The second objective

was completed though the evaluation of trim performance and OLFQ. The induction of off-nominal states, objective three, was done 1) in simulation by rewriting the JSBSim C++ code to include a variety of switches and autopilots, and 2) in flight by using a variable stability aircraft capable of freezing various control surfaces. Data was collected and evaluated against FAA *14 CFR 60* metrics for full flight simulator qualification to support the forth objective. Additionally, a comparison between a proprietary MITL model of the test aircraft was completed using Theil's Inequality Coefficient, when appropriate, to determine which model provided a better match. Finally, through the efforts of the first four objectives, recommendations for model improvements is contained within the following sections.

5.1 Results Summary

A condensed summary of results for JSBSim is shown by Table 31, which covers trim and OLFQ. Table 32 covers the baseline performance maneuver predictions.

Table 31. Results Summary: Trim and OLFQ

	Parameter	Evaluation Criteria	Result
Nominal Trim	Pitch (θ)	$\pm 1.5^\circ$	MET
	Elevator (δ_{es})	$\pm 1.0^\circ$	MET
	Net Thrust (δ_t)	$\pm 5\%$	MET (240-260 KIAS)
Off-Nominal Trim	Pitch (θ)	$\pm 1.5^\circ$	MET
	Elevator (δ_{es})	$\pm 1.0^\circ$	MET
	Net Thrust (δ_t)	$\pm 5\%$	MET (240-260 KIAS)
	Rudder (δ_r)	$\pm 1.0^\circ$	MET
	Sideslip (β)	$\pm 2.0^\circ$	MET
Short Period Dynamics	Pitch (θ)	$\pm 1.5^\circ$	MET
	Pitch Rate (Q)	$\pm 2^\circ/\text{sec}$	MET
Phugoid Dynamics	Period (T_{ph})	$\pm 10\%$	MET
	Damping Ratio (ζ_{ph})	± 0.02	MET
Roll Response	Roll Deviation (ϕ)	$\pm 10\%$	MET
	Roll Rate (P)	$\pm 2^\circ/\text{sec}$	MET
Dutch Roll Dynamics	Yaw Rate (R)	$\pm 2^\circ/\text{sec}$	MET
	Period (T_{dr})	$\pm 0.5 \text{ sec}$	MET
	Damping Ratio (ζ_{dr})	± 0.02	MET
Spiral Stability	Bank (ϕ)	Correct Trend and $\pm 2^\circ/\text{sec}$	MET

Table 32. Results Summary: Performance Maneuver

Configuration	ROC (fpm)	Turn Dir.	P	Q	R	δ_{cs}	δ_r	δ_t	Notes
Nominal	500	R	MET	MET	MET	MET	MET	MET	
	1000	R	MET	MET	MET	MET	MET	MET	
	2000	R	MET	MET	MET	MET	MET	MET	
Rudder Failed	500	R	MET	MET	MET	MET	N/A	MET	With rudder bias, no split required
	1000	R	MET	MET	MET	MET	N/A	MET	Split thrust under-predicted ($\Delta=200$ lbf)
	2000	R	MET	MET	MET	MET	N/A	MET	With rudder bias, no split required
Ailerons Failed	500	R	MET	MET	MET	MET	MET	MET	Aileron bias (-0.2°)
	1000	R	MET	MET	MET	MET	MET	MET	Aileron bias (-0.2°)
	2000	R	MET	MET	MET	MET	MET	MET	Aileron bias (-0.2°)
Single Engine Failed	500	R	MET	MET	MET	MET	FAIL	MET	δ_r over-estimated ($+2.6^\circ$)
	1000	R	MET	MET	MET	MET	FAIL	MET	δ_r over-estimated ($+2.6^\circ$)
	2000	R	-	-	-	-	-	-	Insufficient P_s
	500	L	MET	MET	MET	MET	FAIL	MET	δ_r over-estimated ($+2.0^\circ$)
	1000	L	MET	MET	MET	MET	FAIL	MET	δ_r over-estimated ($+2.1^\circ$)
	2000	L	-	-	-	-	-	-	Insufficient P_s
Single Engine and Ailerons Failed	500	R	MET	MET	MET	MET	FAIL	MET	Aileron bias (0.35°), δ_r slightly over-estimated (1.39°)
	1000	R	MET	MET	MET	MET	MET	MET	Aileron bias (2.47°)
	2000	R	-	-	-	-	-	-	Insufficient P_s
	500	L	MET	MET	MET	MET	MET	MET	Aileron bias (0.65°)
	1000	L	MET	MET	MET	MET	MET	MET	Aileron bias (0.65°)
	2000	L	-	-	-	-	-	-	Insufficient P_s
Single Engine and Rudder Failed	500	R	MET	MET	MET	MET	N/A	MET	Rudder bias (-0.33°)
	1000	R	MET	MET	MET	MET	N/A	MET	Rudder bias (-0.49°)
	2000	R	-	-	-	-	-	-	Insufficient P_s
	500	L	MET	MET	MET	MET	N/A	MET	Rudder bias (-0.5°)
	1000	L	MET	MET	MET	MET	N/A	MET	Rudder bias (-0.5°)
	2000	L	-	-	-	-	-	-	Insufficient P_s
Ailerons and Rudder Failed	500	R	MET	MET	MET	MET	N/A	MET	Rudder bias (-0.37°), aileron bias (-0.22°), Split thrust over-predicted ($\Delta=400$ lbf)
	1000	R	MET	MET	MET	MET	N/A	MET	Rudder bias (-0.37°), aileron bias (-0.22°), Split thrust over-predicted ($\Delta=530$ lbf)
	2000	R	MET	MET	MET	MET	N/A	MET	Rudder bias (-0.37°), aileron bias (1.22°), Split thrust over-predicted ($\Delta=160$ lbf)
Elevator Failed	N/A	N/A	-	-	-	-	-	-	Not possible

5.2 Conclusions

TIC Utility. TIC was developed as a summary measure of forecasting accuracy, largely applied to economic research. TIC was chosen for this test based on three primary advantages over some alternative summary methods; TIC is bounded to the interval 0 to 1 which provides a normalized criterion, TIC covers the case where forecasts are consistently over or underestimated (i.e. identifies systematic linear biases), and TIC is easy to use with very small amounts of calculation. However, the TIC method has disadvantages and limitations as well. A large constant bias (e.g. trim or instrumentation bias), constant phase error (e.g. time-shifted frequency response), mismatching units, or relative magnitudes of outputs were sources of error capable of making TIC values unusable or uninterpretable. The TIC weighting factor (W) is incorporated into the calculation process to balance the relative output amplitudes of various units, but was not necessary for this particular test as all factors were output in degree, degrees/second, or lbf. An identity matrix was used in place of the W matrix. While mismatching units was not a factor, magnitudes were an issue based on the presence of biases and phase shifts. When output values were very small, very small biases greatly affected TIC values, driving them closer to 1. Therefore, it was important to remember that TIC values were only useful in context of each particular predictive comparison. TIC values could not be compared or averaged across multiple parameters, test points, or as a whole. In cases of significant known biases or phase shift extra care must be taken in interpreting TIC results, or consideration should be given to using an alternative comparison method.

Flight Test Bias. One of the main problems with comparing flight test results to simulation is the presence of biases within the flight test data. The Calspan Corporation investigated the source of the biases with a command gain of zero and

determined a trim loop outside of the VSS computer, controlled by the left seat (safety) pilot, to be the source of the biases. This fact was missed in the research methodology build-up and was not captured until post-flight data review. To reduce error in future tests for comparing simulator predictions to in-flight results, a “trim-shot” should be completed prior to each maneuver to identify trim biases (e.g. a non-zero rudder/aileron deflection due to a bent airframe) in addition to starting maneuvers with a stable platform.

FAA 14 CFR 60 Criteria Utility. The FAA 14 CFR 60 criteria was used in this research as a metric threshold for simulator prediction accuracy. The FAA uses this same criteria to determine if simulators are an accurate representation of the real-world aircraft, but allows a wide range for satisfactory performance in many parameters. In many cases aircraft simulators purposely operate with different P&FQ characteristics to counter human factor deficiencies which arise due to the inherent nature of motion or fixed-motion simulators. While lower-fidelity training simulators may suitably represent real-world aircraft for the purposes of pilot training, a higher degree of fidelity may be desired for the purpose of early-stage multistate aircraft design. An engineering qualification criteria, with stricter tolerances, should be considered for further testing in this field.

Proprietary versus Open-Source Simulation. The analysis of OLFQ and trim performance reveals that the open-source simulator outperformed the proprietary MITL simulator (the HBL) in every parameter evaluated. This analysis shows that development of a costly simulation model may be avoided and nominal performance can be predicted to a high degree of accuracy with a very simple aerodynamic and flight dynamics model (like JSBSim).

Asymmetric Thrust and Single Engine Performance. The center-line of the Learjet 25D's engines lie ≈ 44 inches away from the longitudinal x-axis of the aircraft. With a wingspan of 35.7 ft, the engines are considered to be very close to the aircraft center-line. As a result, asymmetric thrust situations produce a large variability in the control deflections required to maintain desired roll, pitch, and sideslip angles. For example, during the stabilized turn with the rudder failed, the throttles were used to control the angle of sideslip. JSBSim under-predicted the split thrust required in post-flight analysis. The reason for this under-prediction is two-fold: 1) the sideslip tolerances were $\pm 2^\circ$, which was chosen based on analog display limitations, and 2) asymmetric thrust changes of a few hundred pounds produced very little sideslip change. Since the tolerances were large, based on the pilot's ability to read the gauge, variations in stabilized sideslip angle can account for the differences in predicted asymmetric thrust, in this case.

In the ailerons and rudder failed case, the angle of sideslip is ignored and the aircraft is rolled using differential throttles. This maneuver was by far the most difficult and pilot technique dependent to complete within tolerances. The initial rolling moment was generated by aggressively placing one engine at idle and then attempting to "catch" the desired bank angle by aggressively setting the idle engine at some intermediate thrust setting. The bank angle tolerances were $\pm 3^\circ$ based on analog gauge limitations. This 6° window of acceptable bank angle allows for a large variance in split thrust required.

Concerning the single engine performance failures regarding rudder deflection, the coupled nature of the Learjet 25D flight controls drive errors in predicted control deflection. During the single engine maneuvers three control effectors are producing rolling and yawing moments on the aircraft: the ailerons, the rudder, and the moment produced by asymmetric thrust. It is difficult to isolate a single cause of the failure in

this case since multiple solutions of required aileron and rudder deflection may exist.

5.3 Recommendations for Future Work

Although JSBSim was shown to be a viable method for predicting off-nominal performance, several areas can improve on the methods used in this research.

R1. The integration of control law in this research was not accomplished due to time and funding constraints. Hand flying the maneuver, while it did provide an accurate comparison to controller driven simulation in most cases, it introduced several sources of error that could be isolated by integrating autopilots in the test aircraft. If the test aircraft can integrate user programmed autopilot controllers, it should be considered to reduce the amount of error from maneuver tolerances and pilot technique. The focus of the autopilot development should start with controlling: sideslip with rudder, airspeed with throttle, sideslip with throttle, and roll angle with throttle.

R2. A flight-test technique that can be adapted to improve the simulation's asymmetric thrust predictive ability is static air minimum control speed (V_{MCA}) determination. One method to determine V_{MCA} is the thrust moment method. This method consists of a series of speed stable points flown with full rudder applied. Bank angle is used to maintain a constant heading. Starting with symmetric power, asymmetric thrust is applied incrementally against the rudder until maximum asymmetry occurs. Bank angle, EPR, temperature, altitude, velocity, gross weight, and Mach are recorded at each stable point. This method is typically used to determine the minimum controllable airspeed, which is the minimum airspeed that should a critical engine failure occur, the aircraft would still be controllable. This method also gives insight into the thrust moments produced by the aircraft, with thrust moment being the difference in thrust between left and right engines times the lateral distance the engines are from the c.g.. In addition to thrust moments, the rudder and aileron deflection required

to hold the steady heading will help determine the rolling moment generated by the thrust asymmetry. This maneuver should be considered to calibrate the simulation in asymmetric thrust scenarios. Further information on the method is contained within [17, p. 603].

R3. To further improve the predictive capability of the simulation with regards to thrust, an accurate thrust map should be utilized rather than use a thrust calculation derived from EPR. Using a ground test stand with the Learjet’s engine, EPR could be directly related to engine output and not estimated using pressure ratios. An accurate thrust map is also required for the thrust moment method described in **R2**.

R4. A universal metric for comparing simulation time histories to flight test data should be considered (or developed), with direct ties to FAA simulator evaluation criteria, in order to overcome the limitations of TIC, discussed in Section 5.2.

Summary. In summary, JSBSim provided an excellent model for nominal predictive capability, outperforming the proprietary simulation model for trim performance and OLFQ. Concerning the off-nominal predictive ability, the body-axis rates (P, Q, and R), elevator-stabilizer deflection (δ_{es}), and net thrust (δ_t) was accurately predicted in each configuration. They also both predicted maneuver feasibility; in no case was a maneuver impossible in the simulator, but possible in-flight or vice-versa. The simulation’s predictive ability was reduced in asymmetric thrust configurations as discussed in Section 5.2. The failures of the evaluation criteria described in Table 32 can be attributed to any or all of the following error sources: broad maneuver tolerances, multiple possible solutions resulting from coupled nature of flight controls, or incorrect asymmetric thrust moment predictions by JSBSim. While there were a few exceedances in the evaluation criteria, the solutions predicted by the simulator in those cases were within reason, assuming multiple possible solutions were possible. Overall, the process should be considered effective for application of mulitstate design

of aircraft.

Appendix A. JSBSim 6-DoF Simulation Model

A Introduction

JSBSim [9] is a nonlinear, high-fidelity, 6-DoF FDM used to model the dynamics of a user supplied aircraft or spacecraft. The software is open source and is written in the C++ programming language. Work on this software started in 1996 by Jon S. Berndt (hence JSB) and has since evolved in to an international group of developers and contributors. The software library now contains over 70,000+ lines of C++ code over 185 files. JSBSim is licensed under GNU General Public License, which guarantees users the right to legally modify, copy, study, and share the software. The general public release of JSBSim can be accessed online at <http://jsbsim.sourceforge.net/>.

The JSBSim has been utilized in many applications over the past 15 years. Most notably, JSBSim is integrated into the flight simulator [28] as a selectable FDM, along with LaRCSim (“Langely Research Center Simulator”) and YASim (“Yet Another Simulator”). MathWorks’ Simulink (coupled with MATLAB) provides output ability to FlightGear for simulation visualization. In addition to FlightGear, JSBSim has been integrated into OpenEagles (Open Extensible Architecture for Analysis and Generation of Linked Simulations) [57], ArenaLogic [5], VirtualAir [73] and the University of Naples driving/flight simulator [10]. JSBSim was also utilized for two HWIL simulators: the Aerocross System Echo Hawk [1] and the duPont Aerospace DP-1 testbed [34]. In addition to simulation purposes, human pilot math models were created and simulated for the U.S. Department of Transportation (D.O.T) using JSBSim [9].

The following sections outline the JSBSim FDM and how it was integrated into this research. The intent of this appendix is not to provide a user-guide for the software (this can be accessed online at [9]) but to offer a brief overview of the

methods utilized to model the Learjet 25D.

A.1 Equations of Motion

The purpose of JSBSim is to propagate a vector of aircraft states over time, given the forces and moments that act on the aircraft. This process is implemented in the JSBSim class *FGPropagate* using quaternion, matrix, and vector notation. Four reference frames are utilized to describe the states based on desired use of the end user. These standard frames are NED (North, East, Down), the body-fixed frame, ECEF (Earth Centered, Earth Fixed), and ECI (Earth Centered Inertial). Rotations between the various frames are accomplished using a variety of direction-cosine matrices. The equations of motion for translation and angular motion are based on equations in Stevens and Lewis's *Aircraft Control and Simulation* [70]. In JSBSim, the translational acceleration is given in the body-fixed frame by Equation A.1.

$${}^b\dot{v}_{CM/e} = \frac{F_{A,T}}{m} - (\omega_{b/e} + 2\omega_{e/i}) \times v_{CM/e} + g \quad (\text{A.1})$$

where $F_{A,T}$ is the vector sum of aerodynamic and thrust forces at the center of mass, $\omega_{b/e}$ is the angular velocity vector of the body-fixed frame relative to the ECEF frame, $\omega_{e/i}$ is the angular velocity vector of the ECEF frame relative to the ECI frame, $v_{CM/e}$ is the velocity of the center of mass with respect to the ECEF frame, m is the mass of the aircraft, and g is the gravitational force vector.

The angular acceleration of the aircraft is described by Equation A.2.

$${}^b\dot{\omega}_{b/i} = (I_p^b)^{-1} [M_{A,T}^b - \Omega_{b/i}^b I_p^b \omega_{b/i}^b] \quad (\text{A.2})$$

where I_p is the inertial matrix in the body-fixed frame, $M_{A,T}^b$ is the sum of aerody-

namic an thrust moments about the center of mass, and $\Omega_{b/i}^b$ is the angular rotation rate vector of the Earth.

Translational velocity and angular velocity are found by integrating Equations A.1 and A.2, respectively. Due to the highly nonlinear nature of these equations, no analytical solution exists. As a result, numerical approaches are used to solve for the velocity and angular information of the aircraft. JSBSim uses several types of integration methods for this process, which can be selected by the user. These integration methods are rectangular, trapezoidal, and Adams-Bashforth 2nd and 3rd order. The rectangular integrator approximates a definite integral using a rectangle for each time step; it is also known as Euler’s numerical method. Similarly, the trapezoidal integrator uses a trapezoidal approximation for the area under the curve, similar to Runge-Kutta and linear multistep methods. Adams-Bashford (2nd and 3rd order) methods are linear multistep integration algorithms that approximate the integrand with a polynomial, within the time interval (i.e. time-step). JSBSim allows for simultaneous solving of different integration methods during the propagation of the state vector. As a default, the program integrates translational motion using the Adams-Bashforth 3rd order method and angular motion using a trapezoidal method. Additional integration of Equations A.1 and A.2, allows JSBSim to determine the 6 degree-of-freedom state vector necessary to describe the full motion of the aircraft: translational position (x, y, z) , and the Euler-angles (ϕ, θ, ψ) .

A.2 Aircraft Modeling

The front-end aircraft configuration file is authored in the XML programming language and contains all the relevant information about the aircraft to be modeled. This file can be loaded into a real-time simulator (e.g. FlightGear) or run as part of a batch simulation process, as is the case with JSBSim. The configuration file is

divided into various sections. Relevant areas of the configuration file are described in the following sections.

A.2.1 Aerodynamics

In order to propagate the state vector of the aircraft, the forces and moments applied must be calculated at each instant in time. JSBSim utilizes a coefficient build-up method to calculate the resulting forces and moments. For example, the lift force is calculated by summing all the contributions to lift, which would include contributions from flaps, elevators and the wing. These forces are calculated by multiplying a specific aerodynamic property by an aerodynamic coefficient particular to the aircraft. These aerodynamic coefficients are known as stability and control derivatives. Stability derivatives are measures of how a force or moment changes with a small change in aircraft state (airspeed, angle of attack, altitude, etc.). Control derivatives are measures of how a force or moment changes with deflections of the control surfaces (ailerons, elevators, and rudder) [63]. The longitudinal force coefficients (lift, pitch and drag) are determined using Equations 6, 7, and 8. The lateral force coefficients (side force, roll, and yaw) are determined using Equations 9, 10, and 11, in section 2.3.

Full variable descriptions and values are contained in Appendix B, Table C.3. The classical method for determining individual parameters of these stability and control equations is through modeling the aircraft in a wind tunnel and extracting each parameter individually. More modern approaches have shown CFD to also be an effective method to calculate these derivatives [58]. Though costly, real-world flight test is the best method to achieve the most accurate coefficients. For the Learjet used in this research, the aerodynamic data was based on real-world flight test data provided by the Calspan Corporation. If flight test data did not exist for a specific

parameter, data from Jay Roskam’s Airplane Flight Dynamics and Automatic Control [63], pages 617-621, was utilized to fill in the gaps. Table C.3 shows the variable descriptions, the XML parameter label, the value, and the source of the information.

A.2.2 Propulsion

Several different types of engines can be modeled in JSBSim: piston, rocket, turbine, turboprop, and electric. Engine modules are specified by *engine name* and placement information (where the engine is located on the aircraft) in the propulsion section of the aircraft configuration file. The *engine name* points to an additional XML engine configuration file, which contains the thrust profile and fuel consumption of each engine. The power output of each engine comes from a look-up table and is a function of altitude, airspeed and the position of the throttle lever. The Learjet 25D modeled, contains two General Electric CJ-610 turbojet engines mounted symmetrically on the side of the aft fuselage. Each engine is rated at 2870 lbs thrust at military power [13].

A.2.3 Mass and Balance

The mass and balance of the aircraft is contained within the metrics section of the configuration file. Exact values of the mass of the aircraft, center of gravity, and aerodynamic center were determined from information provided in the specific aircraft’s weight and balance sheets. The inertia matrix for the simulation, I_p (Equation A.3), was calculated for each test maneuver to account for variances in fuel distribution and aircraft configuration.

$$I_p = \begin{pmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{pmatrix} \Rightarrow \begin{pmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{xz} & 0 & I_{zz} \end{pmatrix} \quad (\text{A.3})$$

Aircraft symmetry allowed the neglecting of the I_{xy} and I_{yz} products of inertia. When comparing the simulation data to the flight test data, the metrics section of the configuration file was updated to mirror the actual conditions of the flight.

A.2.4 Flight Control and Failure Mode Implementation

JSBSim has the ability to incorporate various flight control routines in the form of filters, switches, summers, gains, limiters, PID controllers, etc. These components can be built into a separate XML autopilot configuration file and act as autopilots. For better comparison with flight-test data, a series of controllers are incorporated on each of the input channels (throttle, aileron, elevator, and rudder). The controller design is discussed in greater detail in Chapter III.

Since this research deals with prediction of off-nominal performance, a method was needed to *fail* a control surface or engine. Failure of the flight control was accomplished through the autopilot *switch components* within the autopilot XML aircraft file. Switch components can represent a physical cockpit switch or a logical statement switch. This is similar to creating a switch in the cockpit that enables the autopilot to hold a specific altitude or heading. Figure A.1 shows an example of a switch on and off an airspeed hold autopilot. The input property of the switch is the switch name, *fcs/airspeed_error*, and takes a default value of 0.0 when the switch is off. When activated the switch applies a zero multiplier to the output property of the component *fcs/airspeed_ap_autoswitch_throttle*. Similar methods are used to send no inputs to a specific control surface (ailerons, rudder, throttles, etc.).

```
<switch name="fcs/airspeed-ap-autoswitch-throttle">
  <default value="0.0"/>
  <test value="fcs/airspeed-error">
    ap/airspeed_hold_throt = 1
  </test>
  <output>fcs/airspeed-ap-autoswitch-throttle</output>
</switch>
```

Figure A.1. JSBSim Switch Component

A.3 MATLAB Integration

JSBSim was initially designed to run as a batch simulation program from the command line through the use of various XML-based configuration files and scripts. The open source nature of the project evolved to include incorporation into various flight-simulators (e.g. FlightGear) and graphical programming environments (e.g. ScicosLab and Simulink). In 2009, work began on compiling the JSBSim library as a Simulink function block, also known as an S-function [38]. An S-function is created using a special compiler and syntax that allows for interaction with the Simulink engine through the MATLAB workspace [43]. The JSBSim library is essentially compiled into a MEX-File (MATLAB executable), which can be inserted into the Simulink workspace and manipulated and integrated with the built-in Simulink toolboxes. Integration into the MATLAB workspace allows for easier manipulation of input/output data and for the use of MATLAB's built-in optimizers with very little C++ programming knowledge. Figure A.2, shows pictorially how the S-function was integrated into the Simulink workspace.

Two versions of the S-function were created: *JSBSim_Sfunction.mex* and *JSBSim_Sfunction_trim.mex*. *JSBSim_Sfunction.mex* uses discrete time and its internal integrators to calculate the resulting state outputs. *JSBSim_Sfunction_trim.mex* uses continuous states computed using Simulink's solvers. The *trim* version of the S-function allowed MATLAB access to the state derivatives, a requirement for using its own internal trim function (*trim.m*).

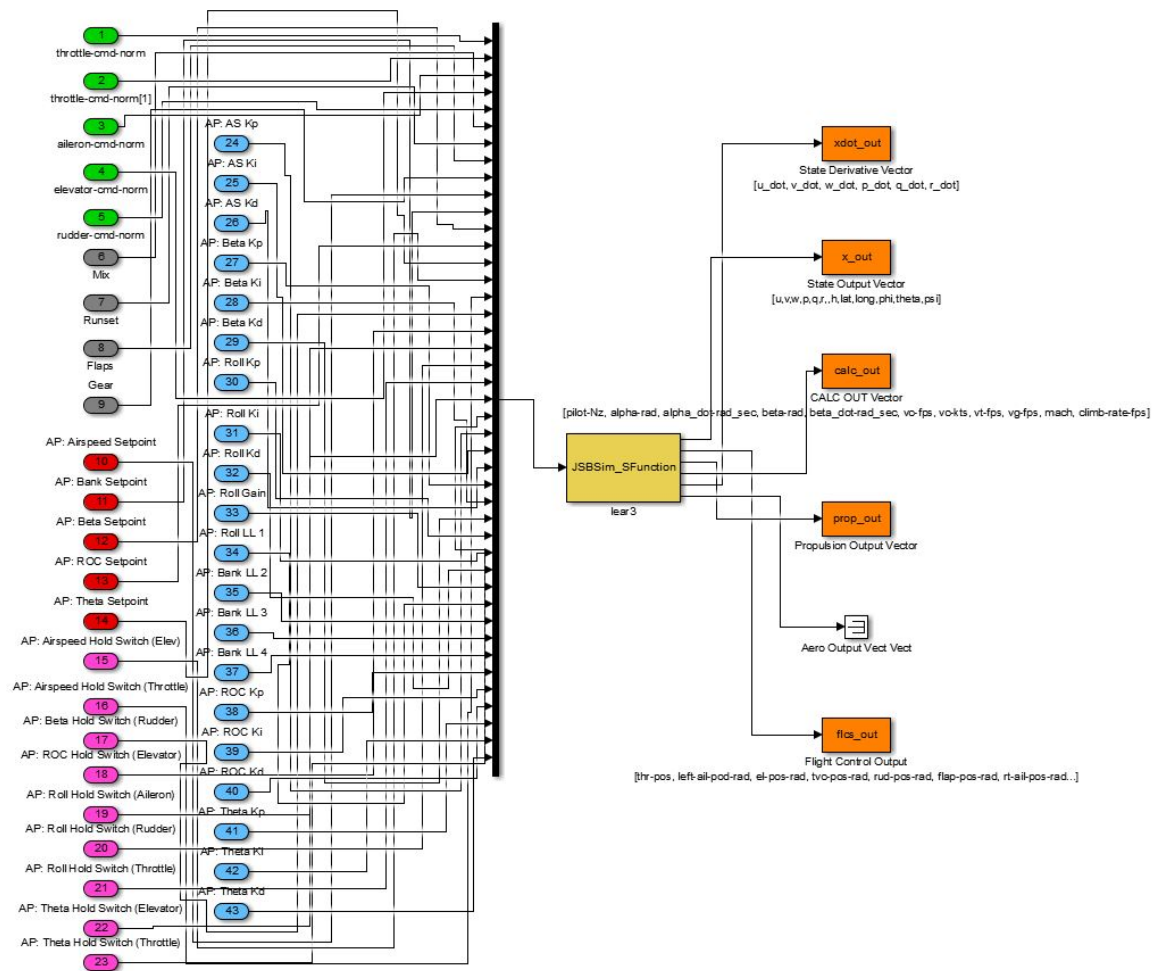


Figure A.2. JSBSim S-Function Example

Appendix B. JSBSim C++ Code Modification

This appendix describes the methodology to modify the JSBSim S-function MEX (MATLAB executable) file for use within the Simulink workspace. This tutorial will allow the the addition of a variety of input channels (i.e. throttle control, autopilot switches, autopilot controller gains, etc.) but can also be followed to modify the output structure of the S-function to suit a specific need.

The JSBSim.S-function.mex64 file was developed by Brian Mills and Augustino de Marco and can be retrieved at [38]. The C++ code was modified starting from version 1.1 of the S-function. The open source version of the S-function contains only one input channel for throttle, which when used with a dual engine aircraft applies the same thrust command to both engines. To add an input channel for independent control of both engines steps 1 and 2 below need to be applied. Since the throttle control is already defined within the JSBSim library, steps 3 and 4 are not required. To add new variables for control law or autopilot switches, steps 3 and 4 need to be applied.

The C++ modification code is contained in the subsequent sections, the modified sections are bracketed by *Modified* and *EndModified* comments. The previous line of original code is included for completeness.

1. JSBSim_SFunction.cpp

Locate the JSBSim_SFunction Source folder and the file JSBSile_SFunction.cpp. Modifications are required in three areas.

Add the new input variables to *#define* section:

```
#define throttle_1          mxGetPr(ssGetSFcnParam(S, 2))[0]
/*-- Modified[0]--*/
#define throttle_2          mxGetPr(ssGetSFcnParam(S, 2))[1]
/*-- EndModified[0]--*/
```

```

#define gear                mxGetPr(ssGetSFcnParam(S, 2))[8]
/*--- Modified[2]---*/
#define AS.setpoint         mxGetPr(ssGetSFcnParam(S, 2))[9]
#define BANK.setpoint       mxGetPr(ssGetSFcnParam(S, 2))[10]
#define BETA.setpoint       mxGetPr(ssGetSFcnParam(S, 2))[11]
#define ROC.setpoint        mxGetPr(ssGetSFcnParam(S, 2))[12]
#define THETA.setpoint      mxGetPr(ssGetSFcnParam(S, 2))[13]
#define AS.hold             mxGetPr(ssGetSFcnParam(S, 2))[14]
#define AS.hold_throt       mxGetPr(ssGetSFcnParam(S, 2))[15]
#define Beta.hold           mxGetPr(ssGetSFcnParam(S, 2))[16]
#define ROC.hold            mxGetPr(ssGetSFcnParam(S, 2))[17]
#define Roll.hold_ail       mxGetPr(ssGetSFcnParam(S, 2))[18]
#define Roll.hold_rud       mxGetPr(ssGetSFcnParam(S, 2))[19]
#define Roll.hold_throt     mxGetPr(ssGetSFcnParam(S, 2))[20]
#define Theta.hold_elev     mxGetPr(ssGetSFcnParam(S, 2))[21]
#define Theta.hold_throt    mxGetPr(ssGetSFcnParam(S, 2))[22]
#define AS.kp               mxGetPr(ssGetSFcnParam(S, 2))[23]
#define AS.ki               mxGetPr(ssGetSFcnParam(S, 2))[24]
#define AS.kd               mxGetPr(ssGetSFcnParam(S, 2))[25]
#define Beta.kp             mxGetPr(ssGetSFcnParam(S, 2))[26]
#define Beta.ki             mxGetPr(ssGetSFcnParam(S, 2))[27]
#define Beta.kd             mxGetPr(ssGetSFcnParam(S, 2))[28]
#define Roll.kp             mxGetPr(ssGetSFcnParam(S, 2))[29]
#define Roll.ki             mxGetPr(ssGetSFcnParam(S, 2))[30]
#define Roll.kd             mxGetPr(ssGetSFcnParam(S, 2))[31]
#define Roll.gain           mxGetPr(ssGetSFcnParam(S, 2))[32]
#define Roll.lag            mxGetPr(ssGetSFcnParam(S, 2))[33]
#define Roll_LL.two        mxGetPr(ssGetSFcnParam(S, 2))[34]
#define Roll_LL.three      mxGetPr(ssGetSFcnParam(S, 2))[35]
#define Roll_LL.four       mxGetPr(ssGetSFcnParam(S, 2))[36]
#define ROC.kp              mxGetPr(ssGetSFcnParam(S, 2))[37]
#define ROC.ki              mxGetPr(ssGetSFcnParam(S, 2))[38]
#define ROC.kd              mxGetPr(ssGetSFcnParam(S, 2))[39]
#define Theta.kp            mxGetPr(ssGetSFcnParam(S, 2))[40]
#define Theta.ki            mxGetPr(ssGetSFcnParam(S, 2))[41]
#define Theta.kd            mxGetPr(ssGetSFcnParam(S, 2))[42]
/*--- EndModified[2]---*/

```

Change the number of input channels, should be index +1 from last #define statement:

```

ssSetNumInputPorts(S, 1);
/*--- Modified[2]---*/
ssSetInputPortWidth(S, 0, 43);
/*--- EndModified[2]---*/

```

Add new variables and ties to initial conditions section:

```

const char *field_names[] = {"name", "value"};
/*--- Modified[2]---*/
struct init_cond ic[] = {{ "u-fps", u_fps }, { "v-fps", v_fps }, { "w-fps", w_fps },
{ "p-rad-sec", p_radsec }, { "q-rad-sec", q_radsec }, { "r-rad-sec", r_radsec },
{ "phi-rad", phi_rad }, { "theta-rad", theta_rad }, { "psi-rad", psi_rad },
{ "long-gc-deg", long_gc_deg }, { "lat-gc-deg", lat_gc_deg }, { "h-sl-ft", h_sl_ft },

```

```

{"aileron-cmd-norm", aileron}, {"elevator-cmd-norm", elevator},
{"rudder-cmd-norm", rudder}, {"fcs/throttle-cmd-norm", throttle_1},
{"fcs/throttle-cmd-norm[1]", throttle_2}, {"fcs/mixture-cmd-norm", mixture},
{"set-running", runset}, {"fcs/flap-cmd-norm", flaps}, {"gear/gear-cmd-norm", gear},
{"ap/airspeed-setpoint", AS_setpoint}, {"ap/bank-setpoint", BANK_setpoint},
{"ap/beta-setpoint", BETA_setpoint}, {"ap/roc-setpoint", ROC_setpoint},
{"ap/theta-setpoint", THETA_setpoint}, {"ap/airspeed-hold", AS_hold},
{"ap/airspeed-hold-throt", AS_hold_throt}, {"ap/beta-hold", Beta_hold},
{"ap/roc-hold", ROC_hold}, {"ap/bank-hold-aileron", Roll_hold_ail},
{"ap/bank-hold-rud", Roll_hold_rud}, {"ap/bank-hold-throt", Roll_hold_throt},
{"ap/theta-hold-elev", Theta_hold_elev}, {"ap/theta-hold-throt", Theta_hold_throt},
{"ap/airspeed-pid-kp", AS_kp}, {"ap/airspeed-pid-ki", AS_ki},
{"ap/airspeed-pid-kd", AS_kd}, {"ap/beta-pid-kp", Beta_kp},
{"ap/beta-pid-ki", Beta_ki}, {"ap/beta-pid-kd", Beta_kd}, {"ap/roll-pid-kp", Roll_kp},
{"ap/roll-pid-ki", Roll_ki}, {"ap/roll-pid-kd", Roll_kd}, {"ap/roll-gain", Roll_gain},
{"ap/roll-lag", Roll_lag}, {"ap/roll-LL-two", Roll_LL_two}, {"ap/roll-LL-three",
Roll_LL_three}, {"ap/roll-LL-four", Roll_LL_four}, {"ap/roc-pid-kp", ROC_kp},
{"ap/roc-pid-ki", ROC_ki}, {"ap/roc-pid-kd", ROC_kd}, {"ap/theta-pid-kp", Theta_kp},
{"ap/theta-pid-ki", Theta_ki}, {"ap/theta-pid-kd", Theta_kd}, {"multiplier",
multiplier}, {"rate_rot", rate_rot}, {"rate_trans", rate_trans}, {"pos_rot", pos_rot},
{"pos_trans", pos_trans}};
/*-- EndModified[2]--*/

```

2. JSBSimInterface.cpp

Within the JSBSim_SFunction Source folder locate the file JSBSimInterface.cpp.

Add fdmExec states to the bool JSBSimInterface::UpdateStates command:

```

/*-- Modified[0]--*/
for (unsigned i=0; i<fdmExec->GetPropulsion()->GetNumEngines(); i++)
    fdmExec->GetFCS()->SetThrottleCmd(0, u_ptr[0]);
    fdmExec->GetFCS()->SetThrottleCmd(1, u_ptr[1]);
/*-- EndModified[0]--*/

```

```

fdmExec->GetFCS()->SetGearCmd(u_ptr[8]);
/*-- Modified[2]--*/
fdmExec->GetFCS()->SetASset(u_ptr[9]);
fdmExec->GetFCS()->SetBANKset(u_ptr[10]);
fdmExec->GetFCS()->SetBETAsset(u_ptr[11]);
fdmExec->GetFCS()->SetROCset(u_ptr[12]);
fdmExec->GetFCS()->SetTHETAsset(u_ptr[13]);
fdmExec->GetFCS()->SetAShold(u_ptr[14]);
fdmExec->GetFCS()->SetASholdThrot(u_ptr[15]);
fdmExec->GetFCS()->SetBetaHold(u_ptr[16]);
fdmExec->GetFCS()->SetROCHold(u_ptr[17]);
fdmExec->GetFCS()->SetBankHoldAil(u_ptr[18]);
fdmExec->GetFCS()->SetBankHoldRud(u_ptr[19]);
fdmExec->GetFCS()->SetBankHoldThrot(u_ptr[20]);
fdmExec->GetFCS()->SetThetaHoldElev(u_ptr[21]);
fdmExec->GetFCS()->SetThetaHoldThrot(u_ptr[22]);
fdmExec->GetFCS()->SetASKp(u_ptr[23]);
fdmExec->GetFCS()->SetASKi(u_ptr[24]);

```

```

fdmExec->GetFCS()->SetASkd(u_ptr[25]);
fdmExec->GetFCS()->SetBETAkp(u_ptr[26]);
fdmExec->GetFCS()->SetBETAKi(u_ptr[27]);
fdmExec->GetFCS()->SetBETAKd(u_ptr[28]);
fdmExec->GetFCS()->SetROLLkp(u_ptr[29]);
fdmExec->GetFCS()->SetROLLki(u_ptr[30]);
fdmExec->GetFCS()->SetROLLkd(u_ptr[31]);
fdmExec->GetFCS()->SetRollGain(u_ptr[32]);
fdmExec->GetFCS()->SetRollLag(u_ptr[33]);
fdmExec->GetFCS()->SetRollLLtwo(u_ptr[34]);
fdmExec->GetFCS()->SetRollLLthree(u_ptr[35]);
fdmExec->GetFCS()->SetRollLLfour(u_ptr[36]);
fdmExec->GetFCS()->SetROCKp(u_ptr[37]);
fdmExec->GetFCS()->SetROCKi(u_ptr[38]);
fdmExec->GetFCS()->SetROCKd(u_ptr[39]);
fdmExec->GetFCS()->SetThetakp(u_ptr[40]);
fdmExec->GetFCS()->SetThetaki(u_ptr[41]);
fdmExec->GetFCS()->SetThetakd(u_ptr[42]);
/*--- End Modified [2] ---*/

```

3. FGFCs.h

Locate the FGFCs.h file in the *JSBSim\HeaderFiles* directory. Changes are required in three locations.

Add new variable names to inline double section:

```

double GetRollTrimCmd(void) const { return RTrimCmd; }
/*--- Modified [2] ---*/

inline double GetASset(void) const { return AS_set_comm; }
inline double GetBANKset(void) const { return BANK_set_comm; }
inline double GetBETAsset(void) const { return BETA_set_comm; }
inline double GetROCset(void) const { return ROC_set_comm; }
inline double GetTHETAsset(void) const { return THETA_set_comm; }
inline double GetAShold(void) const { return AS_hold_comm; }
inline double GetASholdThrot(void) const { return AS_hold_throt_comm; }
inline double GetBetaHold(void) const { return Beta_hold_comm; }
inline double GetROCHold(void) const { return ROC_hold_comm; }
inline double GetBankHoldAil(void) const { return Roll_hold_ail_comm; }
inline double GetBankHoldRud(void) const { return Roll_hold_rud_comm; }
inline double GetBankHoldThrot(void) const { return Roll_hold_throt_comm; }
inline double GetThetaHoldElev(void) const { return Theta_hold_elev_comm; }
inline double GetThetaHoldThrot(void) const { return Theta_hold_throt_comm; }
inline double GetASkp(void) const { return AS_kp_comm; }
inline double GetASki(void) const { return AS_ki_comm; }
inline double GetASkd(void) const { return AS_kd_comm; }
inline double GetBETAkp(void) const { return BETA_kp_comm; }
inline double GetBETAKi(void) const { return BETA_ki_comm; }
inline double GetBETAKd(void) const { return BETA_kd_comm; }
inline double GetROLLkp(void) const { return ROLL_kp_comm; }
inline double GetROLLki(void) const { return ROLL_ki_comm; }
inline double GetROLLkd(void) const { return ROLL_kd_comm; }
inline double GetRollGain(void) const { return RollGain; }

```

```

inline double GetRollLag(void) const { return RollLag; }
inline double GetRollLL_two(void) const { return RollLL_two; }
inline double GetRollLL_three(void) const { return RollLL_three; }
inline double GetRollLL_four(void) const { return RollLL_four; }
inline double GetROCKp(void) const { return ROC_kp_comm; }
inline double GetROCKi(void) const { return ROC_ki_comm; }
inline double GetROCKd(void) const { return ROC_kd_comm; }
inline double GetThetakp(void) const { return Theta_kp_comm; }
inline double GetThetaki(void) const { return Theta_ki_comm; }
inline double GetThetakd(void) const { return Theta_kd_comm; }
/*--EndModified[2]--*/

```

Add new variable names to inline void section:

```

void SetRollTrimCmd(double cmd) { RTrimCmd = cmd; }
/*--Modified[2]--*/
inline void SetASset(double cmd) { AS_set_comm =cmd; }
inline void SetBANKset(double cmd) { BANK_set_comm =cmd; }
inline void SetBETAsset(double cmd) { BETA_set_comm =cmd; }
inline void SetROCset(double cmd) { ROC_set_comm =cmd; }
inline void SetTHETAsset(double cmd) { THETA_set_comm =cmd; }
inline void SetAShold(double cmd) { AS_hold_comm =cmd; }
inline void SetASholdThrot(double cmd) { AS_hold_throt_comm =cmd; }
inline void SetBetaHold(double cmd) { Beta_hold_comm =cmd; }
inline void SetROCHold(double cmd) { ROC_hold_comm =cmd; }
inline void SetBankHoldAil(double cmd) { Roll_hold_ail_comm =cmd; }
inline void SetBankHoldRud(double cmd) { Roll_hold_rud_comm = cmd; }
inline void SetBankHoldThrot(double cmd) { Roll_hold_throt_comm = cmd; }
inline void SetThetaHoldElev(double cmd) { Theta_hold_elev_comm = cmd; }
inline void SetThetaHoldThrot(double cmd) { Theta_hold_throt_comm = cmd; }
inline void SetASkp(double cmd) { AS_kp_comm = cmd; }
inline void SetASki(double cmd) { AS_ki_comm = cmd; }
inline void SetASkd(double cmd) { AS_kd_comm = cmd; }
inline void SetBETAp(double cmd) { BETA_kp_comm = cmd; }
inline void SetBETAKi(double cmd) { BETA_ki_comm = cmd; }
inline void SetBETAKd(double cmd) { BETA_kd_comm = cmd; }
inline void SetROLLkp(double cmd) { ROLL_kp_comm = cmd; }
inline void SetROLLki(double cmd) { ROLL_ki_comm = cmd; }
inline void SetROLLkd(double cmd) { ROLL_kd_comm = cmd; }
inline void SetRollGain(double cmd) { RollGain = cmd; }
inline void SetRollLag(double cmd) { RollLag =cmd; }
inline void SetRollLL_two(double cmd) { RollLL_two =cmd; }
inline void SetRollLL_three(double cmd) { RollLL_three =cmd; }
inline void SetRollLL_four(double cmd) { RollLL_four =cmd; }
inline void SetROCKp(double cmd) { ROC_kp_comm = cmd; }
inline void SetROCKi(double cmd) { ROC_ki_comm = cmd; }
inline void SetROCKd(double cmd) { ROC_kd_comm = cmd; }
inline void SetThetakp(double cmd) { Theta_kp_comm = cmd; }
inline void SetThetaki(double cmd) { Theta_ki_comm = cmd; }
inline void SetThetakd(double cmd) { Theta_kd_comm = cmd; }
/*--EndModified[2]--*/

```

Set the new variables to private:

```

private:
    double DaCmd, DeCmd, DrCmd, DsCmd, DfCmd, DsbCmd, DspCmd;
    double DePos[NForms], DaLPos[NForms], DaRPos[NForms], DrPos[NForms];
    double DfPos[NForms], DsbPos[NForms], DspPos[NForms];
    double PTrimCmd, YTrimCmd, RTrimCmd;
    /*-- Modified[2]--*/
    double AS_set_comm, BANK_set_comm, BETA_set_comm, ROC_set_comm, THETA_set_comm;
    double AS_hold_comm, AS_hold_throt_comm, Beta_hold_comm, ROC_hold_comm, Roll_hold_ail_comm,
    double Roll_hold_rud_comm, Roll_hold_throt_comm, Theta_hold_elev_comm, Theta_hold_throt_comm;
    double AS_kp_comm, AS_ki_comm, AS_kd_comm, BETA_kp_comm, BETA_ki_comm, BETA_kd_comm;
    double ROLL_kp_comm, ROLL_ki_comm, ROLL_kd_comm;
    double RollGain, RollLag, RollLL_two, RollLL_three, RollLL_four;
    double ROC_kp_comm, ROC_ki_comm, ROC_kd_comm, Theta_kp_comm, Theta_ki_comm, Theta_kd_comm;
    /*-- EndModified[2]--*/

```

4. FGFCs.cpp

Locate the FGFCs.cpp file in the *JSBSim\SourceFiles* directory. Changes are required in three locations.

Insert initial conditions into *FGModel(fdmex)* and *InitModel(void)*:

```

PTrimCmd = YTrimCmd = RTrimCmd = 0.0;
/*-- Modified[2]--*/
AS_set_comm = 0.0;
BANK_set_comm = 0.0;
BETA_set_comm = 0.0;
ROC_set_comm = 0.0;
THETA_set_comm = 0.0;
AS_hold_comm = 0.0;
AS_hold_throt_comm = 0.0;
Beta_hold_comm = 0.0;
ROC_hold_comm = 0.0;
Roll_hold_ail_comm = 0.0;
Roll_hold_rud_comm = 0.0;
Roll_hold_throt_comm = 0.0;
Theta_hold_elev_comm = 0.0;
Theta_hold_throt_comm = 0.0;
AS_kp_comm = 0.0;
AS_ki_comm = 0.0;
AS_kd_comm = 0.0;
BETA_kp_comm = 0.0;
BETA_ki_comm = 0.0;
BETA_kd_comm = 0.0;
ROLL_kp_comm = 0.0;
ROLL_ki_comm = 0.0;
ROLL_kd_comm = 0.0;
RollGain = 1.0;
RollLag = 1.0;
RollLL_two = 1.0;
RollLL_three = 1.0;
RollLL_four = 1.0;

```

```

ROC_kp_comm = 0.0;
ROC_ki_comm = 0.0;
ROC_kd_comm = 0.0;
Theta_kp_comm = 0.0;
Theta_ki_comm = 0.0;
Theta_kd_comm = 0.0;
//*--EndModified[2]--*/

```

Add PropertyManager Ties:

```

PropertyManager->Tie(" fcs /yaw-trim-cmd-norm", this, &FGFCS::GetYawTrimCmd, &FGFCS::SetYawTrimCmd);
// Modified [2] //
PropertyManager->Tie("ap/airspeed-setpoint", this, &FGFCS::GetASset, &FGFCS::SetASset);
PropertyManager->Tie("ap/bank-setpoint", this, &FGFCS::GetBANKset, &FGFCS::SetBANKset);
PropertyManager->Tie("ap/beta-setpoint", this, &FGFCS::GetBETAsset, &FGFCS::SetBETAsset);
PropertyManager->Tie("ap/roc-setpoint", this, &FGFCS::GetROCset, &FGFCS::SetROCset);
PropertyManager->Tie("ap/theta-setpoint", this, &FGFCS::GetTHETAsset, &FGFCS::SetTHETAsset);
PropertyManager->Tie("ap/airspeed_hold", this, &FGFCS::GetAShold, &FGFCS::SetAShold);
PropertyManager->Tie("ap/airspeed_hold_throt", this, &FGFCS::GetASHoldThrot, &FGFCS::
SetASHoldThrot);
PropertyManager->Tie("ap/beta_hold", this, &FGFCS::GetBetaHold, &FGFCS::SetBetaHold);
PropertyManager->Tie("ap/roc_hold", this, &FGFCS::GetROCHold, &FGFCS::SetROCHold);
PropertyManager->Tie("ap/bank_hold_ailerons", this, &FGFCS::GetBankHoldAil, &FGFCS::
SetBankHoldAil);
PropertyManager->Tie("ap/bank_hold_rud", this, &FGFCS::GetBankHoldRud, &FGFCS::SetBankHoldRud);
PropertyManager->Tie("ap/bank_hold_throt", this, &FGFCS::GetBankHoldThrot, &FGFCS::
SetBankHoldThrot);
PropertyManager->Tie("ap/theta_hold_elev", this, &FGFCS::GetThetaHoldElev, &FGFCS::
SetThetaHoldElev);
PropertyManager->Tie("ap/theta_hold_throt", this, &FGFCS::GetThetaHoldThrot, &FGFCS::
SetThetaHoldThrot);
PropertyManager->Tie("ap/airspeed-pid-kp", this, &FGFCS::GetASKp, &FGFCS::SetASKp);
PropertyManager->Tie("ap/airspeed-pid-ki", this, &FGFCS::GetASKi, &FGFCS::SetASKi);
PropertyManager->Tie("ap/airspeed-pid-kd", this, &FGFCS::GetASKd, &FGFCS::SetASKd);
PropertyManager->Tie("ap/beta-pid-kp", this, &FGFCS::GetBETAp, &FGFCS::SetBETAp);
PropertyManager->Tie("ap/beta-pid-ki", this, &FGFCS::GetBETAi, &FGFCS::SetBETAi);
PropertyManager->Tie("ap/beta-pid-kd", this, &FGFCS::GetBETAd, &FGFCS::SetBETAd);
PropertyManager->Tie("ap/roll-pid-kp", this, &FGFCS::GetROLLkp, &FGFCS::SetROLLkp);
PropertyManager->Tie("ap/roll-pid-ki", this, &FGFCS::GetROLLki, &FGFCS::SetROLLki);
PropertyManager->Tie("ap/roll-pid-kd", this, &FGFCS::GetROLLkd, &FGFCS::SetROLLkd);
PropertyManager->Tie("ap/roll-gain", this, &FGFCS::GetRollGain, &FGFCS::SetRollGain);
PropertyManager->Tie("ap/roll-lag", this, &FGFCS::GetRollLag, &FGFCS::SetRollLag);
PropertyManager->Tie("ap/roll-LL-two", this, &FGFCS::GetRollLL_two, &FGFCS::SetRollLL_two);
PropertyManager->Tie("ap/roll-LL-three", this, &FGFCS::GetRollLL_three, &FGFCS::SetRollLL_three);
PropertyManager->Tie("ap/roll-LL-four", this, &FGFCS::GetRollLL_four, &FGFCS::SetRollLL_four);
PropertyManager->Tie("ap/roc-pid-kp", this, &FGFCS::GetROCKp, &FGFCS::SetROCKp);
PropertyManager->Tie("ap/roc-pid-ki", this, &FGFCS::GetROCKi, &FGFCS::SetROCKi);
PropertyManager->Tie("ap/roc-pid-kd", this, &FGFCS::GetROCKd, &FGFCS::SetROCKd);
PropertyManager->Tie("ap/theta-pid-kp", this, &FGFCS::GetThetap, &FGFCS::SetThetap);
PropertyManager->Tie("ap/theta-pid-ki", this, &FGFCS::GetThetaki, &FGFCS::SetThetaki);
PropertyManager->Tie("ap/theta-pid-kd", this, &FGFCS::GetThetakd, &FGFCS::SetThetakd);
//EndModified [2]//

```

Finally, to complete the process you have to rebuild the JSBSim Solution, the JSBSimLib Solution, and the JSBSim_Sfunction solution. Then copy the JSBSimS_function.mex64 file over the old one.

Appendix C. Learjet 25D Modeling Data

This appendix contains the data used to aerodynamically model the Learjet. This data was compiled into an front-end *XML* file for use within JSBSim. Learjet 25D stability and control derivatives were supplied by the Calspan corporation, extracted at 250 KIAS and 15,000 ft MSL (units in degrees for applicable derivatives).

Table C.1. Learjet 25D Geometric Parameters [45]

Description	Value	Unit
Wing Area	231.8	ft^2
Wingspan	35.7	ft
Wing Incidence	1	deg
Mean Chord	84.49	in
Horiz. Tail Area	54.0	ft^2
Horiz. Tail Arm	266.8	in
Vert. Tail Area	38.35	ft^2
Vert. Tail Arm	220.3	in
Length	47.85	ft

Table C.2. Learjet 25D Flight Control Parameters [12]

Flight Control	Value	Unit
Elevator	± 15	deg
Aileron	± 18	deg
Rudder	± 30	deg
Flap Settings	8, 20, 40	deg

Table C.3. Learjet 25D Stability and Control Derivatives

Variable	Description	XML Function	Value
C_{L_0}	Lift at zero angle of attack	Lift_alpha	0.050934
C_{L_α}	Lift due to change in angle of attack	Lift_alpha	0.0939507
C_{L_q}	Lift due to pitch rate	Lift_q	0.0
$C_{L_{\delta e}}$	Lift due to change in elevator	Lift_elevator	0.00487973
$C_{L_{\delta s}}$	Lift due to change in stab	Lift_stab	0.0082955
C_{m_0}	Pitching moment at zero control deflection	Pitch_basic	-0.0035174
C_{m_α}	Pitching moment due to angle of attack	Pitch_alpha	-0.0127488241
C_{m_q}	Pitching moment due to pitch rate	Pitch_damp	-0.320942
$C_{m_{\delta e}}$	Pitching moment due to elevator	Pitch_elevator	-0.0158689
$C_{m_{\delta s}}$	Pitching moment due to stab	Pitch_stab	-0.02539
$C_{m_{\delta f}}$	Pitching moment due to flaps	Pitch_flap	-0.0002
C_{D_0}	Drag at zero lift	Drag_basic	0.028646
C_{D_q}	Drag due to pitch rate	Drag_q	0.0
C_{D_α}	Drag due to alpha	Drag_alpha	0.00166917
$C_{D_{\delta e}}$	Drag due to change in elevator	Drag_elevator	0.0014809
$C_{D_{\delta f}}$	Drag due to change in flap	Drag_flap	0.0025
C_{Y_p}	Side force due to roll rate	Side_damp	-0.001571
C_{Y_r}	Side force due to yaw rate	Side_yaw	0.006806
C_{Y_β}	Side force due to beta	Side_beta	-0.001571
$C_{Y_{\delta a}}$	Side force due to aileron	Side_aileron	0.0
$C_{Y_{\delta r}}$	Side force due to rudder	Side_rudder	0.00281

Table C.4. Learjet 25D Stability and Control Derivatives (cont.)

Variable	Description	XML Function	Value
C_{l_p}	Roll moment due to roll rate	Roll_damp	-0.00804484
C_{l_r}	Roll moment due to yaw rate	Roll_yaw	0.00300753
C_{l_β}	Roll moment due to beta	Roll_beta	-0.000142286
$C_{l_{\delta a}}$	Roll moment due to aileron	Roll_aileron	-0.0129857
$C_{l_{\delta r}}$	Roll moment due to rudder	Roll_rudder	0.000303700
C_{n_p}	Yaw moment due to roll rate	Yaw_roll	-0.0017507
C_{n_r}	Yaw moment due to yaw rate	Yaw_damp	-0.00456048
C_{n_β}	Yaw moment due to beta	Yaw_beta	0.00150910
$C_{n_{\delta a}}$	Yaw moment due to aileron	Yaw_aileron	0.0000987520
$C_{n_{\delta r}}$	Yaw moment due to rudder	Yaw_rudder	-0.000999357

Appendix D. JSBSim Learjet 25D XML Aircraft Configuration Code

```

1 <?xml version="1.0"?>
2 <?xml-stylesheet href="JSBSim.xml" type="application/xml"?>
3 <fdm_config name="learjet_24" version="2.0" release="ALPHA">
4 <!-- *****
5     FILE HEADER
6     ***** -->
7     <fileheader>
8         <author> Mark Vahle, Maj, USAF </author>
9         <email> mark.vahle@gmail.com </email>
10        <note>
11            This aircraft was created as part of a Masters thesis at the Air Force Institute of
12            Technology.
13            The aircraft is owned by the Calspan Corporation and leased by USAF Test Pilot School.
14        </note>
15        <filecreationdate> 2015-07-15 </filecreationdate>
16        <description>Model for a Learjet 25 VSS Aircraft</description>
17        <version> $Revision: 2.3$ </version>
18        <reference refID="None" author="Jay Roskam" title="Airplane Flight Dynamics and
19            Control" date="1994"/>
20    </fileheader>
21    <!-- *****
22    METRICS - X-axis datum is 100 inches in front of nose, centered through the fuselage
23    ***** -->
24    <metrics>
25        <wingarea unit="FT2"> 231.8 </wingarea> <!-- CALSPAN 231.8 -->
26        <wingspan unit="FT"> 35.5833 </wingspan> <!-- CALSPAN 35.5833 -->
27        <chord unit="FT"> 6.51 </chord> <!-- CALSPAN -->
28        <wing_incidence> 1 </wing_incidence> <!-- CALSPAN -->
29        <htailarea unit="FT2"> 54 </htailarea> <!-- CALSPAN -->
30        <htailarm unit="FT"> 22.23 </htailarm> <!-- CALSPAN, only used for turbulence -->
31        <vtailarea unit="FT2"> 38.35 </vtailarea> <!-- CALSPAN -->
32        <vtailarm unit="FT"> 18.35 </vtailarm> <!-- CALSPAN , only used for turbulence -->
33        <location name="AERORP" unit="IN"> <!-- The aerodynamic reference point, common point
34            for all aero forces-->
35            <x> 278.66 </x> <!-- datum is 100 inches in front of nose, I subtracted 100 from
36            everything to make the new datum the nose-->
37            <y> 0 </y>
38            <z> 0 </z>
39        </location>
40        <location name="EYEPOINT" unit="IN">
41            <x> 88 </x> <!-- EYE point for pilot, only used when using flightgear -->
42            <y> 0 </y>
43            <z> 44.72 </z>
44        </location>
45        <location name="VRP" unit="IN"> <!-- visual reference point, for 3D model, NOSE-->
46            <x> 0 </x>
47            <y> 0 </y>
48            <z> 7.36 </z>
49        </location>
50    </metrics>

```

```

47 <!-- *****
48     MASS_BALANCE
49     ***** -->
50     <mass_balance>
51         <ixx unit="SLUG*FT2">22921</ixx> <!-- CALSPAN 14420 23164-->
52         <iyy unit="SLUG*FT2">27979</iyy> <!-- CALSPAN 27945-->
53         <izz unit="SLUG*FT2">48714</izz> <!-- CALSPAN 48915 -->
54         <ixz unit="SLUG*FT2">1970</ixz> <!-- CALSPAN 1965 -->
55         <emptywt unit="LBS">9920</emptywt> <!-- CALSPAN, 9920 = 2 pilot + 3 crew -->
56         <location name="CG" unit="IN"> <!-- I changed the datum to be the nose, rather than
            100 inches in front -->
57         <x>272.28</x> <!-- 272.28 EMPTY WEIGHT CG!! - CALSPAN/TCDS, 270.9677 LIMITS
            366.8-385.4 -->
58         <y>0</y>
59         <z>23.63</z> <!-- CALSPAN, near wing root -->
60     </location>
61     <pointmass name="FUEL Weight">
62         <weight unit="LBS">800</weight> <!-- FUEL TOTAL -->
63         <location unit="IN"> <!-- FUEL TOTAL 278.66 -->
64             <x>295.00</x>
65             <y>0</y>
66             <z>23.63</z>
67         </location>
68     </pointmass>
69 </mass_balance>
70 <!-- *****
71     UNDERCARRIAGE: Bogey = ground contact point, structure = location when gear is up
72     ***** -->
73     <ground_reactions>
74         <contact type="BOGEY" name="NOSE">
75             <location unit="IN">
76                 <x> 61.08 </x>
77                 <y> 0 </y>
78                 <z> -31.57 </z>
79             </location>
80             <static_friction> 0.8 </static_friction>
81             <dynamic_friction> 0.5 </dynamic_friction>
82             <rolling_friction> 0.02 </rolling_friction>
83             <spring_coeff unit="LBS/FT"> 4650 </spring_coeff>
84             <damping_coeff unit="LBS/FT/SEC"> 7750 </damping_coeff>
85             <max_steer unit="DEG"> 15 </max_steer>
86             <brake_group> NONE </brake_group>
87             <retractable> 1 </retractable>
88         </contact>
89         <contact type="BOGEY" name="LEFT.MAIN">
90             <location unit="IN">
91                 <x> 294.4 </x>
92                 <y> -44.16</y>
93                 <z> -31.57 </z>
94             </location>
95             <static_friction> 0.8 </static_friction>
96             <dynamic_friction> 0.5 </dynamic_friction>
97             <rolling_friction> 0.02 </rolling_friction>
98             <spring_coeff unit="LBS/FT"> 15500 </spring_coeff>

```

```

99         <damping-coeff unit="LBS/FT/SEC"> 7750 </damping-coeff>
100         <max-steer unit="DEG"> 0 </max-steer>
101         <brake-group> LEFT </brake-group>
102         <retractable> 1 </retractable>
103     </contact>
104     <contact type="BOGEY" name="RIGHT_MAIN">
105         <location unit="IN">
106             <x> 294.4 </x>
107             <y> -44.16 </y>
108             <z> -31.57 </z>
109         </location>
110         <static-friction> 0.8 </static-friction>
111         <dynamic-friction> 0.5 </dynamic-friction>
112         <rolling-friction> 0.02 </rolling-friction>
113         <spring-coeff unit="LBS/FT"> 15500 </spring-coeff>
114         <damping-coeff unit="LBS/FT/SEC"> 7750 </damping-coeff>
115         <max-steer unit="DEG"> 0 </max-steer>
116         <brake-group> RIGHT </brake-group>
117         <retractable> 1 </retractable>
118     </contact>
119     <contact type="STRUCTURE" name="NOSE">
120         <location unit="IN">
121             <x> 44.16 </x>
122             <y> 0 </y>
123             <z> 1.55 </z>
124         </location>
125         <static-friction> 0.2 </static-friction>
126         <dynamic-friction> 0.2 </dynamic-friction>
127         <spring-coeff unit="LBS/FT"> 15500 </spring-coeff>
128         <damping-coeff unit="LBS/FT/SEC"> 15500 </damping-coeff>
129     </contact>
130     <contact type="STRUCTURE" name="LEFT-WING">
131         <location unit="IN">
132             <x> 294.4 </x>
133             <y> -22.08 </y>
134             <z> 1.55 </z>
135         </location>
136         <static-friction> 0.2 </static-friction>
137         <dynamic-friction> 0.2 </dynamic-friction>
138         <spring-coeff unit="LBS/FT"> 15500 </spring-coeff>
139         <damping-coeff unit="LBS/FT/SEC"> 15500 </damping-coeff>
140     </contact>
141     <contact type="STRUCTURE" name="RIGHT-WING">
142         <location unit="IN">
143             <x> 294.4 </x>
144             <y> -22.08 </y>
145             <z> 1.55 </z>
146         </location>
147         <static-friction> 0.2 </static-friction>
148         <dynamic-friction> 0.2 </dynamic-friction>
149         <spring-coeff unit="LBS/FT"> 15500 </spring-coeff>
150         <damping-coeff unit="LBS/FT/SEC"> 15500 </damping-coeff>
151     </contact>
152 </ground-reactions>

```

```

153 <!-- *****
154     PROPULSION
155     ***** -->
156     <propulsion>
157         <engine file="GE_CJ610"> <!-- LEFT ENGINE -->
158             <location unit="IN">
159                 <x> 348.96 </x>
160                 <y> -44.16 </y>
161                 <z> 44.17 </z> <!-- 44.17 -->
162             </location>
163             <orient unit="DEG">
164                 <roll> 0 </roll>
165                 <pitch> 0 </pitch>
166                 <yaw> 0 </yaw>
167             </orient>
168             <feed>3</feed>
169             <thruster file="direct">
170                 <location unit="IN">
171                     <x> 348.96 </x>
172                     <y> -44.16 </y>
173                     <z> 44.17 </z> <!-- 44.17 -->
174                 </location>
175                 <orient unit="DEG">
176                     <roll> 0 </roll>
177                     <pitch> 0 </pitch>
178                     <yaw> 0 </yaw>
179                 </orient>
180             </thruster>
181         </engine>
182         <engine file="GE_CJ610"> <!-- RIGHT ENGINE -->
183             <location unit="IN">
184                 <x> 348.96 </x>
185                 <y> 44.16 </y>
186                 <z> 44.17 </z> <!-- 44.17 -->
187             </location>
188             <orient unit="DEG">
189                 <roll> 0 </roll>
190                 <pitch> 0 </pitch>
191                 <yaw> 0 </yaw>
192             </orient>
193             <feed>2</feed>
194             <thruster file="direct">
195                 <location unit="IN">
196                     <x> 348.96 </x>
197                     <y> 44.16 </y>
198                     <z> 44.17 </z> <!-- 44.17 -->
199                 </location>
200                 <orient unit="DEG">
201                     <roll> 0 </roll>
202                     <pitch> 0 </pitch>
203                     <yaw> 0 </yaw>
204                 </orient>
205             </thruster>
206         </engine>

```

```

207 <tank type="FUEL" number="0"> <!-- RIGHT WINGTIP TANK -->
208     <location unit="IN"> <!-- Number is the order of feeding 0, 1, 2,-->
209         <x> 282.72 </x>
210         <y> 204.96 </y>
211         <z> 14.72</z>
212     </location>
213     <capacity unit="LBS">1212.5</capacity>
214     <contents unit="LBS">0</contents>
215 </tank>
216 <tank type="FUEL" number="1"> <!-- LEFT WINGTIP TANK -->
217     <location unit="IN">
218         <x> 282.72 </x>
219         <y> -204.96 </y>
220         <z> 14.72</z>
221     </location>
222     <capacity unit="LBS">1212.5</capacity>
223     <contents unit="LBS">0</contents>
224 </tank>
225 <tank type="FUEL" number="2"> <!-- RIGHT WING TANK -->
226     <location unit="IN">
227         <x> 278.66 </x> <!-- 282.72, WAG-->
228         <y> 0 </y> <!-- 93.94, WAG-->
229         <z> 14.72 </z> <!-- -23.63, WAG-->
230     </location>
231     <capacity unit="LBS">1162.5</capacity>
232     <contents unit="LBS">200</contents>
233 </tank>
234 <tank type="FUEL" number="3"> <!-- LEFT WING TANK -->
235     <location unit="IN">
236         <x> 278.66 </x> <!-- 273.37, 270.9677 282.72, WAG-->
237         <y> 0 </y> <!-- -93.94, WAG-->
238         <z> 14.72 </z> <!-- -23.63, WAG-->
239     </location>
240     <capacity unit="LBS">1162.5</capacity>
241     <contents unit="LBS">200</contents>
242 </tank>
243 <tank type="FUEL" number="4"> <!-- FUSELAGE TANK, sits between engines, feeds into
244     wingtanks -->
245     <location unit="IN">
246         <x> 350 </x> <!-- 350, WAG-->
247         <y> 0 </y>
248         <z> 44.17 </z> <!-- 23.63, WAG-->
249     </location>
250     <capacity unit="LBS">1307</capacity>
251     <contents unit="LBS">0</contents><!-- LEAVE EMPTY, SHIFTS CG AFT-->
252 </tank>
253 </propulsion>
254 <!-- *****
255     AUTOPILOT
256     ***** -->
257 <autopilot file="lear.ap"/>
258 <!-- *****
259     FLIGHT CONTROLS

```

```

260 ***** -->
261 <flight_control name="FCSLear3">
262
263   <channel name="Pitch">
264
265     <kinematic name="elevator position normal">
266       <input>fcs/elevator-cmd-norm</input>
267       <traverse>
268         <setting>
269           <position>-1</position>
270           <time>0.3</time>
271         </setting>
272         <setting>
273           <position>1</position>
274           <time>0.3</time> <!-- 60 degrees per second -->
275         </setting>
276       </traverse>
277       <output>fcs/elevator-pos-norm</output>
278     </kinematic>
279
280     <summer name="Pitch Trim Sum">
281       <input>ap/elevator-cmd</input>
282       <input>ap/elevator-cmd-airspeed</input>
283       <input>ap/elevator-cmd-pitch</input>
284       <input>fcs/elevator-cmd-norm</input>
285       <input>fcs/pitch-trim-cmd-norm</input>
286       <clipto>
287         <min>-1</min>
288         <max>1</max>
289       </clipto>
290     </summer>
291
292     <aerosurface_scale name="Elevator Control">
293       <input>fcs/pitch-trim-sum</input>
294       <range>
295         <min>-0.2618</min> <!-- 15 degrees -->
296         <max>0.2618</max>
297       </range>
298       <output>fcs/elevator-pos-rad</output>
299     </aerosurface_scale>
300
301     <aerosurface_scale name="elevator normalization">
302       <input>fcs/elevator-pos-rad</input>
303       <domain>
304         <min>-0.2618</min>
305         <max>0.2618</max>
306       </domain>
307       <range>
308         <min>-1</min>
309         <max>1</max>
310       </range>
311       <output>fcs/elevator-pos-norm</output>
312     </aerosurface_scale>
313   </channel>

```

```

314
315 <channel name="Roll">
316
317     <kinematic name="aileron position">
318         <input>fcs/aileron-cmd-norm</input>
319         <traverse>
320             <setting>
321                 <position>-1</position>
322                 <time>0.3</time>
323             </setting>
324             <setting>
325                 <position>1</position>
326                 <time>0.3</time>
327             </setting>
328         </traverse>
329         <output>fcs/left-aileron-pos-norm</output>
330     </kinematic>
331
332     <summer name="Roll Trim Sum">
333         <input>ap/aileron-cmd</input> <!-- Used for Bank Angle Hold -->
334         <input>fcs/aileron-cmd-norm</input>
335         <input>fcs/roll-trim-cmd-norm</input>
336         <clipto>
337             <min>-1</min>
338             <max>1</max>
339         </clipto>
340     </summer>
341
342     <aerosurface_scale name="Left Aileron Control">
343         <input>fcs/roll-trim-sum</input>
344         <range>
345             <min>-0.3142</min> <!-- 18 degrees -->
346             <max>0.3142</max>
347         </range>
348         <output>fcs/left-aileron-pos-rad</output>
349     </aerosurface_scale>
350
351     <aerosurface_scale name="Right Aileron Control">
352         <input>-fcs/left-aileron-pos-rad</input> <!-- Note Negative sign -->
353         <range>
354             <min>-0.3142</min>
355             <max>0.3142</max>
356         </range>
357         <output>fcs/right-aileron-pos-rad</output>
358     </aerosurface_scale>
359
360     <aerosurface_scale name="left aileron normalization">
361         <input>fcs/left-aileron-pos-rad</input>
362         <domain>
363             <min>-0.3142</min>
364             <max>0.3142</max>
365         </domain>
366         <range>
367             <min>-1</min>

```

```

368         <max>1</max>
369     </range>
370     <output>fcs/left-aileron-pos-norm</output>
371 </aerosurface_scale>
372
373 <aerosurface_scale name="right aileron normalization">
374     <input>fcs/right-aileron-pos-rad</input>
375     <domain>
376         <min>-0.3142</min>
377         <max>0.3142</max>
378     </domain>
379     <range>
380         <min>-1</min>
381         <max>1</max>
382     </range>
383     <output>fcs/right-aileron-pos-norm</output>
384 </aerosurface_scale>
385
386 </channel>
387
388 <channel name="Yaw">
389
390 <property value="0">fcs/yaw-damper-enable</property> <!-- GAIN to Turn on(1) and off(0)
391     Yaw Damper -->
392
393     <summer name="Rudder Command Sum">
394         <input>ap/rudder-cmd-roll-norm</input>
395         <input>ap/rudder-cmd-beta</input>
396         <input>fcs/rudder-cmd-norm</input>
397         <input>fcs/yaw-trim-cmd-norm</input>
398     <clipto>
399         <min>-1</min>
400         <max>1</max>
401     </clipto>
402 </summer>
403
404 <scheduled_gain name="Yaw Damper Rate"> <!-- YAW DAMPER EXTRA -->
405     <input>velocities/r-aero-rad-sec</input>
406     <table>
407         <independentVar lookup="row">velocities/ve-cts</independentVar>
408         <tableData>
409             30      0.00
410             60      2.00
411             250     4.00
412         </tableData>
413     </table>
414     <gain>fcs/yaw-damper-enable</gain>
415     <output>fcs/yaw-damper-rate</output> <!-- YAW DAMPER EXTRA -->
416 </scheduled_gain>
417
418 <aerosurface_scale name="Rudder Control">
419     <input>fcs/rudder-command-sum</input>
420     <input>fcs/yaw-damper-rate</input> <!-- YAW DAMPER EXTRA -->
421     <range>

```

```

421         <min>-0.5236</min> <!-- 30 degrees -->
422         <max>0.5236</max>
423     </range>
424     <output>fcs/rudder-pos-rad</output>
425 </aerosurface_scale>
426
427 <aerosurface_scale name="rudder normalization">
428     <input>fcs/rudder-pos-rad</input>
429     <domain>
430         <min>-0.5236</min>
431         <max>0.5236</max>
432     </domain>
433     <range>
434         <min>-1</min>
435         <max>1</max>
436     </range>
437     <output>fcs/rudder-pos-norm</output>
438 </aerosurface_scale>
439
440 </channel>
441
442 <channel name="Flaps">
443
444     <kinematic name="Flaps Control">
445         <input>fcs/flap-cmd-norm</input>
446         <traverse>
447             <setting>
448                 <position>0</position>
449                 <time>0</time>
450             </setting>
451             <setting>
452                 <position>20</position>
453                 <time>4</time>
454             </setting>
455             <setting>
456                 <position>40</position>
457                 <time>3</time>
458             </setting>
459         </traverse>
460         <output>fcs/flap-pos-deg</output>
461     </kinematic>
462
463     <aerosurface_scale name="flap normalization">
464         <input>fcs/flap-pos-deg</input>
465         <domain>
466             <min>0</min>
467             <max>40</max>
468         </domain>
469         <range>
470             <min>0</min>
471             <max>1</max>
472         </range>
473         <output>fcs/flap-pos-norm</output>
474     </aerosurface_scale>

```

```

475
476     </channel>
477
478     <channel name="Landing Gear">
479
480         <kinematic name="Gear Control">
481             <input>gear/gear-cmd-norm</input>
482             <traverse>
483                 <setting>
484                     <position>0</position>
485                     <time>0</time>
486                 </setting>
487                 <setting>
488                     <position>1</position>
489                     <time>5</time>
490                 </setting>
491             </traverse>
492             <output>gear/gear-pos-norm</output>
493         </kinematic>
494
495     </channel>
496
497     <channel name="Speedbrake">
498
499         <kinematic name="Speedbrake Control">
500             <input>fcs/speedbrake-cmd-norm</input>
501             <traverse>
502                 <setting>
503                     <position>0</position>
504                     <time>0</time>
505                 </setting>
506                 <setting>
507                     <position>1</position>
508                     <time>1</time>
509                 </setting>
510             </traverse>
511             <output>fcs/speedbrake-pos-norm</output>
512         </kinematic>
513
514     </channel>
515
516 </flight_control>
517 <!-- *****
518     AERODYNAMIC FORCES
519     ***** -->
520 <aerodynamics>
521 <!-- *****
522     LIFT
523     ***** -->
524 <axis name="LIFT" unit="LBS">
525     <function name="aero/force/Lift.alpha">
526         <description>Lift due to alpha</description>
527         <product>
528             <property>aero/qbar-psf</property>

```

```

529         <property>metrics/Sw-sqft</property>
530     <table>
531         <independentVar>aero/alpha-rad</independentVar>
532         <tableData> <!-- UPDATED 5 SEPT -->
533             -0.03491      0.03491
534             0             0.050934
535             0.1012       1.051
536             0.2628       1.915
537         </tableData>
538     </table>
539 </product>
540 </function>
541 <function name="aero/force/Lift.flap">
542     <description>Delta Lift due to flaps</description>
543     <product>
544         <property>aero/qbar-psf</property>
545         <property>metrics/Sw-sqft</property>
546         <property>fcs/flap-pos-deg</property>
547         <value>0.01475</value> <!-- UPDATED 5 SEPT -->
548     </product>
549 </function>
550 <function name="aero/force/Lift.speedbrake">
551     <description>Delta Lift due to speedbrake</description>
552     <product>
553         <property>aero/qbar-psf</property>
554         <property>metrics/Sw-sqft</property>
555         <property>fcs/speedbrake-pos-norm</property>
556         <value>0.0</value> <!-- UPDATED 5 SEPT -->
557     </product>
558 </function>
559 <function name="aero/force/Lift.elevator">
560     <description>Lift due to Elevator Deflection</description>
561     <product>
562         <property>aero/qbar-psf</property>
563         <property>metrics/Sw-sqft</property>
564         <property>fcs/elevator-pos-rad</property>
565         <value>0.2796</value> <!-- UPDATED 5 SEPT, 0.00487973*57.3 -->
566     </product>
567 </function>
568
569 <function name="aero/force/Lift.stab"> <!-- ADDED -->
570     <description>Lift due to Stab Deflection</description>
571     <product>
572         <property>aero/qbar-psf</property>
573         <property>metrics/Sw-sqft</property>
574         <value>-1.9</value> <!-- STAB ANGLE 1 of 2, 5 SEPT-->
575         <value>0.0082955</value>
576     </product>
577 </function> <!-- ADDED -->
578
579 </axis>
580 <!-- *****
581 DRAG
582 ***** -->

```

```

583 <axis name="DRAG" unit="LBS">
584   <function name="aero/force/ Drag_basic">
585     <description>Drag polar </description>
586     <product>
587       <property>aero/qbar-psf</property>
588       <property>metrics/Sw-sqft</property>
589       <table>
590         <independentVar>aero/alpha-rad</independentVar>
591         <tableData> <!-- UPDATED 5 SEPT -->
592           -0.03491    0.02531
593           0           0.028646
594           0.1012     0.03833
595           0.2339     0.05101
596         </tableData>
597       </table>
598     </product>
599   </function>
600   <function name="aero/force/ Drag_induced">
601     <description>Induced drag</description>
602     <product>
603       <property>aero/qbar-psf</property>
604       <property>metrics/Sw-sqft</property>
605       <property>aero/cl-squared</property>
606       <value>0.052</value> <!-- historical value, 0.043 UPDATED .052 5 SEPT -->
607     </product>
608   </function>
609   <function name="aero/force/ Drag_flap">
610     <description>Drag due to flaps</description>
611     <product>
612       <property>aero/qbar-psf</property>
613       <property>metrics/Sw-sqft</property>
614       <property>fcs/flap-pos-deg</property>
615       <value>0.0025</value> <!-- UPDATED 5 SEPT -->
616     </product>
617   </function>
618   <function name="aero/force/ Drag_gear">
619     <description>Drag due to gear</description>
620     <product>
621       <property>aero/qbar-psf</property>
622       <property>metrics/Sw-sqft</property>
623       <property>gear/gear-pos-norm</property>
624       <value>0.03</value> <!-- UPDATED 5 SEPT -->
625     </product>
626   </function>
627   <function name="aero/force/ Drag_speedbrake">
628     <description>Drag due to speedbrakes</description>
629     <product>
630       <property>aero/qbar-psf</property>
631       <property>metrics/Sw-sqft</property>
632       <property>fcs/speedbrake-pos-norm</property>
633       <value>0.02</value> <!-- UPDATED 5 SEPT -->
634     </product>
635   </function>
636

```

```

637     <function name="aero/force/Drag-elevator">
638         <description>Drag due to Elevator Deflection</description>
639         <product>
640             <property>aero/qbar-psf</property>
641             <property>metrics/Sw-sqft</property>
642             <abs>
643                 <property>fcs/elevator-pos-rad</property>
644             </abs>
645             <value>0.0849</value> <!-- UPDATED 5 SEPT, 0.0014809*57.3-->
646         </product>
647     </function>
648 </axis>
649 <!-- *****
650 SIDE FORCE
651 ***** -->
652 <axis name="SIDE" unit="LBS">
653     <function name="aero/force/Side.beta">
654         <description>Side force due to beta</description>
655         <product>
656             <property>aero/qbar-psf</property>
657             <property>metrics/Sw-sqft</property>
658             <property>aero/beta-rad</property>
659             <value>-0.6578</value> <!-- UPDATED 5 SEPT, -0.0114807*57.3=-0.6578-->
660         </product>
661     </function>
662     <function name="aero/force/Side.damp">
663         <description>Side Force due to roll rate</description>
664         <product>
665             <property>aero/qbar-psf</property>
666             <property>metrics/Sw-sqft</property>
667             <property>metrics/bw-ft</property>
668             <property>aero/bi2vel</property>
669             <property>velocities/p-aero-rad.sec</property>
670             <value>-0.090018</value> <!-- UPDATED 5 SEPT, -0.001571*57.3-->
671         </product>
672     </function>
673     <function name="aero/force/Side.yaw">
674         <description>Side Force due to yaw rate</description>
675         <product>
676             <property>aero/qbar-psf</property>
677             <property>metrics/Sw-sqft</property>
678             <property>metrics/bw-ft</property>
679             <property>aero/bi2vel</property>
680             <property>velocities/r-aero-rad.sec</property>
681             <value>0.39</value> <!-- UPDATED 5 SEPT, 0.006806*57.3=0.3900-->
682         </product>
683     </function>
684     <function name="aero/force/Side.rudder">
685         <description>Side force due to rudder</description>
686         <product>
687             <property>aero/qbar-psf</property>
688             <property>metrics/Sw-sqft</property>
689             <property>fcs/rudder-pos-rad</property>
690             <value>0.161013</value> <!-- UPDATED 5 SEPT, 0.00281*57.3-->

```

```

691         </product>
692     </function>
693 </axis>
694 <!-- *****
695 ROLL MOMENT
696 ***** -->
697 <axis name="ROLL" unit="LBS*FT">
698     <function name="aero/moment/Roll_beta">
699         <description>Roll moment due to beta</description>
700         <product>
701             <property>aero/qbar-psf</property>
702             <property>metrics/Sw-sqft</property>
703             <property>metrics/bw-ft</property>
704             <property>aero/beta-rad</property>
705             <value>-0.081529878</value> <!-- UPDATED 5 SEPT,
              -0.00142286*57.3=-0.081529878-->
706         </product>
707     </function>
708     <function name="aero/moment/Roll_damp">
709         <description>Roll moment due to roll rate</description>
710         <product>
711             <property>aero/qbar-psf</property>
712             <property>metrics/Sw-sqft</property>
713             <property>metrics/bw-ft</property>
714             <property>aero/bi2vel</property>
715             <property>velocities/p-aero-rad.sec</property>
716             <value>-0.460969332</value> <!-- UPDATED 5 SEPT,
              -0.00804484*57.3=-0.460969332-->
717         </product>
718     </function>
719     <function name="aero/moment/Roll_yaw">
720         <description>Roll moment due to yaw rate</description>
721         <product>
722             <property>aero/qbar-psf</property>
723             <property>metrics/Sw-sqft</property>
724             <property>metrics/bw-ft</property>
725             <property>aero/bi2vel</property>
726             <property>velocities/r-aero-rad.sec</property>
727             <value>0.172331469</value> <!-- UPDATED 5 SEPT,
              0.00300753*57.3=0.172331469-->
728         </product>
729     </function>
730     <function name="aero/moment/Roll_aileron">
731         <description>Roll moment due to aileron</description>
732         <product>
733             <property>aero/qbar-psf</property>
734             <property>metrics/Sw-sqft</property>
735             <property>metrics/bw-ft</property>
736             <property>fcs/left-aileron-pos-rad</property>
737             <value>-0.074408</value> <!-- UPDATED 5 SEPT, -0.00129857*57.3, was
              0.05615-->
738         </product>
739     </function>
740     <function name="aero/moment/Roll_rudder">

```

```

741         <description>Roll moment due to rudder</description>
742         <product>
743             <property>aero/qbar-psf</property>
744             <property>metrics/Sw-sqft</property>
745             <property>metrics/bw-ft</property>
746             <property>fcs/rudder-pos-rad</property>
747             <value>0.01740201</value> <!-- UPDATED 5 SEPT, 0.000303700*57.3-->
748         </product>
749     </function>
750 </axis>
751 <!-- *****
752 PITCH MOMENT
753 ***** -->
754 <axis name="PITCH" unit="LBS*FT">
755     <function name="aero/moment/Pitch-alpha">
756         <description>Pitch moment due to alpha</description>
757         <product>
758             <property>aero/qbar-psf</property>
759             <property>metrics/Sw-sqft</property>
760             <property>metrics/cbarw-ft</property>
761             <property>aero/alpha-rad</property>
762             <value>-0.73050762</value> <!-- UPDATED 5 SEPT, -0.0127488241*57.3-->
763         </product>
764     </function>
765     <function name="aero/moment/Pitch-elevator">
766         <description>Pitch moment due to elevator</description>
767         <product>
768             <property>aero/qbar-psf</property>
769             <property>metrics/Sw-sqft</property>
770             <property>metrics/cbarw-ft</property>
771             <property>fcs/elevator-pos-rad</property>
772             <value>-0.90928797</value> <!-- UPDATED 5 SEPT, -0.0158689*57.3-->
773         </product>
774     </function>
775     <function name="aero/moment/Pitch_stab"> <!-- ADDED -->
776         <description>Pitch moment due to Stab</description>
777         <product>
778             <property>aero/qbar-psf</property>
779             <property>metrics/Sw-sqft</property>
780             <property>metrics/cbarw-ft</property>
781             <value>-1.9</value> <!-- STAB ANGLE 2 of 2, 5 SEPT-->
782             <value>-.02539</value> <!-- UPDATED 5 SEPT-->
783         </product>
784     </function> <!-- ADDED -->
785     <function name="aero/moment/Pitch_damp">
786         <description>Pitch moment due to pitch rate</description>
787         <product>
788             <property>aero/qbar-psf</property>
789             <property>metrics/Sw-sqft</property>
790             <property>metrics/cbarw-ft</property>
791             <property>aero/ci2vel</property>
792             <property>velocities/q-aero-rad.sec</property>
793             <value>-18.3899766</value> <!-- UPDATED 5 SEPT, -.320942*57.3=-18.3899766
-->

```

```

794         </product>
795     </function>
796     <function name="aero/moment/Pitch_alphadot">
797         <description>Pitch moment due to alpha rate</description>
798         <product>
799             <property>aero/qbar-psf</property>
800             <property>metrics/Sw-sqft</property>
801             <property>metrics/cbarw-ft</property>
802             <property>aero/ci2vel</property>
803             <property>aero/alphadot-rad_sec</property>
804             <value>0</value> <!-- UPDATED 5 SEPT -->
805         </product>
806     </function>
807     <function name="aero/moment/Pitch-basic">
808         <description>Pitch moment at zero angle of attack</description>
809         <product>
810             <property>aero/qbar-psf</property>
811             <property>metrics/Sw-sqft</property>
812             <value>-0.0035174</value> <!-- UPDATED 5 SEPT -->
813         </product>
814     </function>
815 </axis>
816 <!-- *****
817 YAW MOMENT
818 ***** -->
819 <axis name="YAW" unit="LBS*FT">
820     <function name="aero/moment/Yaw_beta">
821         <description>Yaw moment due to beta</description>
822         <product>
823             <property>aero/qbar-psf</property>
824             <property>metrics/Sw-sqft</property>
825             <property>metrics/bw-ft</property>
826             <property>aero/beta-rad</property>
827             <value>0.08647143</value> <!-- UPDATED 5 SEPT, 0.00150910*57.3 =
828                 0.08647143-->
829         </product>
830     </function>
831     <function name="aero/moment/Yaw_damp">
832         <description>Yaw moment due to yaw rate</description>
833         <product>
834             <property>aero/qbar-psf</property>
835             <property>metrics/Sw-sqft</property>
836             <property>metrics/bw-ft</property>
837             <property>aero/bi2vel</property>
838             <property>velocities/r-aero-rad_sec</property>
839             <value>-0.261315504</value> <!-- UPDATED 5 SEPT,
840                 -0.00456048*57.3=-0.261315504-->
841         </product>
842     </function>
843     <function name="aero/moment/Yaw_roll">
844         <description>Yaw moment due to roll rate</description>
845         <product>
846             <property>aero/qbar-psf</property>
847             <property>metrics/Sw-sqft</property>

```

```

846         <property>metrics/bw-ft</property>
847         <property>aero/bi2vel</property>
848         <property>velocities/p-aero-rad-sec</property>
849         <value>-0.10031511</value> <!-- UPDATED 5 SEPT, -0.0017507*57.3-->
850     </product>
851 </function>
852 <function name="aero/moment/Yaw-rudder">
853     <description>Yaw moment due to rudder</description>
854     <product>
855         <property>aero/qbar-psf</property>
856         <property>metrics/Sw-sqft</property>
857         <property>metrics/bw-ft</property>
858         <property>fcs/rudder-pos-rad</property>
859         <value>-0.0572631561</value> <!-- UPDATED 5 SEPT,
            -0.000999357*57.3=-0.0572631561-->
860     </product>
861 </function>
862 <function name="aero/moment/Yaw-aileron">
863     <description>Adverse yaw</description>
864     <product>
865         <property>aero/qbar-psf</property>
866         <property>metrics/Sw-sqft</property>
867         <property>metrics/bw-ft</property>
868         <property>fcs/left-aileron-pos-rad</property>
869         <value>-0.0056584896</value> <!-- UPDATED 5 SEPT, -0.0000987520*57.3-->
870     </product>
871 </function>
872 </axis>
873 </aerodynamics>
874
875 <!-- <output name="learjet_datalog.csv" type="CSV" rate="20">
876     <simulation> OFF </simulation>
877     <atmosphere> OFF </atmosphere>
878     <massprops> OFF </massprops>
879     <aerosurfaces> OFF </aerosurfaces>
880     <rates> OFF </rates>
881     <velocities> OFF </velocities>
882     <forces> OFF </forces>
883     <moments> OFF </moments>
884     <position> OFF </position>
885     <coefficients> OFF </coefficients>
886     <ground_reactions> OFF </ground_reactions>
887     <fcs> OFF </fcs>
888     <propulsion> OFF </propulsion>
889     <property>fcs/heading-setpoint</property>
890     <property>attitude/heading-true-rad</property>
891     <property>aero/qbar-psf</property>
892     <property>velocities/u-aero-fps</property>
893     <property>velocities/v-aero-fps</property>
894     <property>velocities/w-aero-fps</property>
895     <property>position/h-sl-ft</property>
896     <property>fcs/elevator-pos-deg</property>
897     <property>fcs/left-aileron-pos-deg</property>
898     <property>fcs/right-aileron-pos-deg</property>

```

```

899     <property>fcs/rudder-pos-deg</property>
900     <property>propulsion/engine/thrust-lbs</property>
901     -->
902 <output name="learjet_datalog.csv" type="CSV" rate="20">
903     <!--<property>velocities/u-aero-fps</property>
904     <property>ap/airspeed-setpoint</property>
905     <property>fcs/airspeed-error</property>
906     <property>ap/throttle_cmd_airspeed</property>
907     <property>fcs/airspeed-ap-error-pid-throttle</property>
908     <property>velocities/v-aero-fps</property>
909     <property>velocities/w-aero-fps</property>
910     <property>position/h-sl-ft</property>
911     <property>ap/roc_hold</property>
912     <property>ap/ROC_setpoint</property>
913     <property>fcs/ROC_hold-pid</property>
914     <property>fcs/ROC-error</property>
915     <property>ap/elevator_cmd</property>
916     <property>fcs/windup-trigger</property>
917     <property>velocities/h-dot-fps</property>
918     <property>fcs/elevator-pos-deg</property>
919     <property>fcs/left-aileron-pos-deg</property>
920     <property>fcs/right-aileron-pos-deg</property>
921     <property>fcs/rudder-pos-deg</property>-->
922     <!--<property>fcs/throttle_cmd-norm[0]</property>
923     <property>fcs/throttle_cmd-norm[1]</property>-->
924     <property>fcs/throttle-pos-norm[0]</property>
925     <property>fcs/throttle-pos-norm[1]</property>
926     <!--<property>ap/throttle_cmd-roll-norm</property>-->
927 </output>
928
929 <!-- This output structure is for FLIGHT GEAR coupled with playbackzog.xml - do not delete -->
930 <!--<output name="learjet_datalog.csv" type="CSV" rate="20"> \
931     <property>fcs/aileron_cmd-norm</property>
932     <property>fcs/elevator_cmd-norm</property>
933     <property>fcs/rudder_cmd-norm</property>
934     <property>fcs/flap_cmd-norm</property>
935     <property>fcs/throttle_cmd-norm[0]</property>
936     <property>fcs/throttle_cmd-norm[1]</property>
937     <property>propulsion/starter_cmd</property>
938     <property>propulsion/starter_cmd</property>
939     <property>propulsion/cutoff_cmd</property>
940     <property>propulsion/cutoff_cmd</property>
941     <property>propulsion/cutoff_cmd</property>
942     <property>propulsion/cutoff_cmd</property>
943     <property>gear/gear-pos-norm</property>
944     <property>position/lat-gc-deg</property>
945     <property>position/long-gc-deg</property>
946     <property>position/h-sl-ft</property>
947     <property>attitude/phi-rad</property>
948     <property>attitude/theta-rad</property>
949     <property>attitude/psi-rad</property>
950     <property>aero/beta-rad</property>
951     <property>velocities/vc-kts</property>
952     <property>velocities/mach</property>

```

```

953     <property>velocities /v-down-fps</property>
954     <property>velocities /v-east-fps</property>
955     <property>velocities /v-north-fps</property>
956     <property>velocities /u-aero-fps</property>
957     <property>velocities /v-aero-fps</property>
958         <property>velocities /w-aero-fps</property>
959         <property>velocities /h-dot-fps</property>
960     <property>accelerations /a-pilot -x-ft_sec2</property>
961     <property>accelerations /a-pilot -y-ft_sec2</property>
962     <property>accelerations /a-pilot -z-ft_sec2</property>
963     <property>fcs /elevator-pos-norm</property>
964         <property>fcs /flap-pos-norm</property>
965         <property>fcs /left-aileron-pos-norm</property>
966         <property>fcs /right-aileron-pos-norm</property>
967         <property>fcs /rudder-pos-norm</property>
968 </output> -->
969
970
971 </fdm-config>

```

Appendix E. JSBSim Learjet 25D XML Autopilot Configuration Code

```
1 <?xml version="1.0"?>
2
3 <autopilot name="Learjet Autopilot">
4
5   <property value="0"> ap/r.setpoint </property> <!-- HEADING SETPOINT -->
6   <property value="0"> ap/heading_hold </property> <!-- SWITCH FOR CONTROLLING HEADING HOLD -->
7   <property value="0"> ap/beta_hold_throt </property> <!-- SWITCH FOR CONTROLLING BETA WITH THROTTLE -->
8   <property value="1"> ap/throttle1_switch </property> <!-- SWITCH FOR turning off engine[1] -->
9
10  <!-- PROPERTY DEFINITIONS -->
11  <property>fcs/throttle-cmd-norm[0]</property>
12  <property>fcs/throttle-cmd-norm[1]</property>
13  <property>fcs/airspeed-ap-error-pid-throttle</property>
14  <property>ap/throttle-cmd-airspeed</property>
15  <property>ap/throttle-cmd-roll-norm</property>
16  <property>fcs/ROC-ap-autoswitch</property>
17  <property>fcs/airspeed-error</property>
18  <property>ap/airspeed-setpoint</property>
19  <property>ap/roc_hold</property>
20  <property>fcs/ROC-error</property>
21  <property>fcs/ROC-hold-pid</property>
22  <property>ap/heading_hold</property>
23  <property>ap/heading_setpoint</property>
24  <property>ap/aileron-cmd</property>
25  <property>ap/elevator-cmd</property>
26  <property>ap/elevator-cmd-airspeed</property>
27  <property>ap/rudder-cmd.beta</property>
28  <property>ap/throttle-cmd-norm</property>
29  <property>ap/throttle-cmd-roll-norm</property>
30  <property>ap/rudder-cmd-roll-norm</property>
31  <property>fcs/ROC-hold-pid</property>
32  <property>fcs/ROC-error</property>
33  <property>velocities/h-dot-fps</property>
34  <property>velocities/r-aero-rad.sec</property>
35  <property>attitude/psi-rad</property>
36
37  <!-- INITIAL GAIN VALUES -->
38
39  <property value="0.5"> ap/hdg-roll-err-cl </property>
40
41  <!-- SENSOR -->
42
43  <!-- <sensor name="fcs/attitude/sensor/phi-rad">
44    <input> attitude/phi-rad </input>
45    <lag> 0.5 </lag>
46    <noise variation="PERCENT"> 0.05 </noise>
47    <quantization name="attitude/sensor/quantized/phi-rad">
```

```

48     <bits> 12 </bits>
49     <min> -180 </min>
50     <max> 180 </max>
51     </quantization>
52     <bias> 0.001 </bias>
53 </sensor> -->
54
55 <!--
56 =====
57 BETA HOLD
58 =====
59 -->
60
61 <!-- Rudder Beta Hold -->
62
63 <channel name="Beta hold">
64
65     <switch name="fcs/beta-hold-ap-on-off">
66         <default value="0" />
67         <test value="0">
68             ap/beta_hold == 1
69         </test>
70     </switch>
71
72     <summer name="fcs/beta-error">
73         <input> aero/beta-rad</input>
74         <input> -ap/beta_setpoint </input>
75         <output>fcs/beta-error</output>
76     </summer>
77
78     <pid name="fcs/beta-ap-error-pid">
79         <input> fcs/beta-error </input>
80         <kp> ap/beta-pid-kp </kp>
81         <ki> ap/beta-pid-ki </ki>
82         <kd> ap/beta-pid-kd </kd>
83         <trigger> fcs/beta-hold-ap-on-off </trigger>
84     </pid>
85
86     <switch name="fcs/beta-ap-autoswitch">
87         <default value="0.0" />
88         <test value="-fcs/beta-ap-error-pid">
89             ap/beta_hold == 1
90         </test>
91     </switch>
92
93     <pure_gain name="fcs/rudder.beta">
94         <input> fcs/beta-ap-autoswitch </input>
95         <gain> 1.0 </gain>
96         <output> ap/rudder_cmd_beta </output>
97     </pure_gain>
98
99 </channel>
100
101 <!-- Throttle Beta Hold -->

```

```

102
103 <channel name="Beta hold Throttle">
104
105   <summer name="fcs/beta-error-throt">
106     <input> aero/beta-rad</input>
107     <input> -ap/beta.setpoint </input>
108     <output>fcs/beta-error-throt</output>
109   </summer>
110
111   <switch name="fcs/beta-hold-autoswitch">
112     <default value="0" />
113     <test value="fcs/beta-error-throt">
114       ap/beta.hold.throt == 1
115     </test>
116     <output>fcs/beta-hold-autoswitch</output>
117   </switch>
118
119
120   <pid name="fcs/beta-ap-error-pid-throt">
121     <input> fcs/beta-hold-autoswitch </input>
122     <kp> ap/beta-pid-kp </kp>
123     <ki> ap/beta-pid-ki </ki>
124     <kd> ap/beta-pid-kd </kd>
125     <clipto>
126       <min> -.1 </min>
127       <max> .1 </max>
128     </clipto>
129     <output> fcs/beta-ap-error-pid-throt </output>
130   </pid>
131
132   <pure_gain name="fcs/throttle_beta">
133     <input> fcs/beta-ap-error-pid-throt </input>
134     <gain> 1.0 </gain>
135     <output> fcs/throttle_beta </output>
136   </pure_gain>
137 </channel>
138
139
140   <!--
141   =====
142   AIRSPEED HOLD (THROTTLE) & PITCH HOLD (THROTTLE)
143   =====
144   -->
145
146   <channel name="AIRSPEED Hold or PITCH Hold with Throttle">
147
148     <summer name="fcs/airspeed-error">
149       <input> velocities/u-fps</input>
150       <input> -ap/airspeed.setpoint </input>
151       <output>fcs/airspeed-error</output>
152     </summer>
153
154     <switch name="fcs/airspeed-ap-autoswitch-throttle">
155       <default value="0.0" />

```

```

156     <test value="fcs/airspeed-error">
157         ap/airspeed_hold_throt == 1
158     </test>
159     <output>fcs/airspeed-ap-autoswitch-throttle</output>
160 </switch>
161
162
163 <pid name="fcs/airspeed-ap-error-pid-throttle">
164     <input> -fcs/airspeed-ap-autoswitch-throttle </input>
165     <kp> ap/airspeed-pid-kp </kp> <!-- 0.15 -->
166     <ki> ap/airspeed-pid-ki </ki> <!-- 0.005 -->
167     <kd> ap/airspeed-pid-kd </kd> <!-- 0.001 -->
168     <clipto>
169         <min> -.4 </min>
170         <max> .4 </max>
171     </clipto>
172     <output>fcs/airspeed-ap-error-pid-throttle</output>
173 </pid>
174
175 <!-- START Throttle Pitch Hold -->
176     <summer name="fcs/theta-error-pre">
177         <input> attitude/theta-rad</input>
178         <input> -ap/theta_setpoint </input>
179     <output> fcs/theta_error </output>
180 </summer>
181
182 <switch name="fcs/theta_hold_throttle">
183     <default value="0" />
184     <test value="fcs/theta_error">
185         ap/theta_hold_throt == 1
186     </test>
187     <output> fcs/theta_hold_throttle</output>
188 </switch>
189
190 <pid name="fcs/theta-ap-error-pid2">
191     <input> fcs/theta_hold_throttle</input>
192     <kp> ap/theta-pid-kp </kp>
193     <ki> ap/theta-pid-ki </ki>
194     <kd> ap/theta-pid-kd </kd>
195     <clipto>
196         <min> -.55</min>
197         <max> .55 </max>
198     </clipto>
199     <output> fcs/theta_hold_pid_throttle </output>
200 </pid>
201 <!-- END Throttle Pitch Hold -->
202
203
204 <summer name="ap/throttle.cmd.left">
205     <input> fcs/airspeed-ap-error-pid-throttle </input>
206     <input> fcs/theta_hold_pid_throttle </input>
207     <input> fcs/throttle-cmd-norm </input>
208     <input> fcs/throttle_beta </input>
209     <clipto> <min> 0 </min>

```

```

210         <max> 1.0</max>
211     </clipto>
212     <output> ap/throttle_cmd_left </output>
213 </summer>
214
215 <pure_gain name="fcs/throttle_airspeed">
216     <input> ap/throttle_cmd_left </input>
217     <gain> 1 </gain>
218     <output> fcs/throttle_pos_norm </output>
219 </pure_gain>
220
221 <summer name="ap/throttle_cmd_right">
222     <input> fcs/airspeed-ap-error-pid-throttle </input>
223     <input> fcs/theta_hold_pid_throttle </input>
224     <input> fcs/throttle_cmd_norm </input>
225     <clipto> <min> 0 </min>
226     <max> 1.0</max>
227 </clipto>
228     <output> ap/throttle_cmd_right </output>
229 </summer>
230
231 <pure_gain name="fcs/throttle_airspeed1">
232     <input> ap/throttle_cmd_right </input>
233     <gain> ap/throttle1_switch </gain> <!-- Change to 0 if you want to look
234         at failed right engine -->
235     <output> fcs/throttle_cmd_norm[1] </output>
236 </pure_gain>
237
238 </channel>
239
240 <!--
241 =====
242 RATE OF CLIMB HOLD
243 =====
244 -->
245
246 <channel name="ROC hold">
247
248 <summer name="fcs/ROC-error">
249     <input> ap/roc_setpoint</input>
250     <input> -velocities/h-dot-fps </input>
251     <output> fcs/ROC-error </output>
252 </summer>
253
254 <switch name="fcs/ROC-ap-autoswitch">
255     <default value="0.0"/>
256     <test value="fcs/ROC-error">
257         ap/roc_hold == 1
258     </test>
259     <output> fcs/ROC-ap-autoswitch</output>
260 </switch>
261
262 <pid name="fcs/ROC-hold-pid">

```

```

263 <input> fcs/ROC-ap-autoswitch </input>
264 <kp> ap/roc-pid-kp </kp> <!-- .010 -->
265 <ki> ap/roc-pid-ki </ki> <!-- .0055 -->
266 <kd> ap/roc-pid-kd </kd> <!-- .005 -->
267 <trigger> fcs/windup-trigger </trigger>
268 <clipto> <min> -1 </min>
269 <max> 1 </max>
270 </clipto>
271 <output> fcs/ROC-hold-pid </output>
272 </pid>
273
274 <pure_gain name="fcs/elevator">
275 <input> fcs/ROC-hold-pid </input> <!-- fcs/ROC-hold-pid -->
276 <gain> -1 </gain>
277 <output> ap/elevator_cmd </output>
278 </pure_gain>
279
280 </channel>
281
282
283 <!--
284 =====
285 AIRSPEED HOLD ( Elevator )
286 =====
287 -->
288
289 <channel name="Airspeed hold">
290
291 <summer name="fcs/airspeed-error">
292 <input> velocities/u-fps</input>
293 <input> -ap/airspeed_setpoint </input>
294 <clipto>
295 <min>-40</min>
296 <max> 40</max>
297 </clipto>
298 <output>fcs/airspeed-error</output>
299 </summer>
300
301 <lag_filter name="fcs/airspeed-error-lag">
302 <input> fcs/airspeed-error </input>
303 <c1> 1.0 </c1>
304 </lag_filter>
305
306 <pid name="fcs/airspeed-ap-error-pid">
307 <input> fcs/airspeed-error </input>
308 <kp> ap/airspeed-pid-kp </kp>
309 <ki> ap/airspeed-pid-ki </ki>
310 <kd> ap/airspeed-pid-kd </kd>
311 <trigger> fcs/windup-trigger </trigger>
312 <clipto> <min> -1.0 </min>
313 <max> 1.0 </max> </clipto>
314 </pid>
315
316 <switch name="fcs/airspeed-ap-autoswitch">

```

```

317     <default value="0.0"/>
318     <test value="-fcs/airspeed-ap-error-pid">
319         ap/airspeed_hold == 1
320     </test>
321 </switch>
322
323 <pure_gain name="fcs/elevator_airspeed">
324     <input> fcs/airspeed-ap-autoswitch </input>
325     <gain> -0.5 </gain>
326     <output> ap/elevator_cmd_airspeed </output>
327 </pure_gain>
328
329 </channel>
330
331
332 <!--
333 =====
334 ROLL HOLD
335 =====
336 -->
337
338 <!-- Aileron Roll Hold -->
339
340 <channel name="Roll wing leveler">
341
342     <switch name="fcs/wing-leveler-ap-on-off">
343         <default value="-1"/>
344         <test value="0">
345             ap/bank_hold_ailerons == 1
346         </test>
347     </switch>
348
349     <summer name="fcs/roll-error-pre">
350         <input> -attitude/phi-rad</input>
351         <input> -ap/bank_setpoint </input>
352     </summer>
353
354     <lead_lag_filter name="fcs/roll-error-lead-lag">
355         <input> fcs/roll-error-pre </input>
356         <c1> ap/roll-lag </c1>
357         <c2> ap/roll-LL-two </c2>
358         <c3> ap/roll-LL-three </c3>
359         <c4> ap/roll-LL-four </c4>
360     </lead_lag_filter>
361
362     <pid name="fcs/roll-ap-error-pid">
363         <input> fcs/roll-error-lead-lag </input>
364         <kp> ap/roll-pid-kp </kp>
365         <ki> ap/roll-pid-ki </ki>
366         <kd> ap/roll-pid-kd </kd>
367         <trigger> fcs/wing-leveler-ap-on-off </trigger>
368     </pid>
369
370     <switch name="fcs/roll-ap-autoswitch">

```

```

371     <default value="0.0"/>
372     <test value="-fcs/roll-ap-error-pid">
373         ap/bank_hold_ailerons == 1
374     </test>
375 </switch>
376
377 <!-- Heading HOLD -->
378
379 <summer name="fcs/r-error-pre">
380     <input> attitude/psi-rad </input>
381     <input> -ap/r.setpoint </input>
382     <output> fcs/r-error-pre </output>
383 </summer>
384
385 <switch name="fcs/heading-ap-autoswitch">
386     <default value="0.0"/>
387     <test value="fcs/r-error-pre">
388         ap/heading_hold == 1
389     </test>
390     <output>fcs/heading-ap-autoswitch</output>
391 </switch>
392
393     <pid name="fcs/heading-ap-error-pid">
394         <input> fcs/heading-ap-autoswitch </input>
395         <kp> .15 </kp>
396         <ki> .005 </ki>
397         <kd> .001 </kd>
398         <clipto>
399             <min>-0.2</min>
400             <max> 0.2</max>
401         </clipto>
402         <output>fcs/heading-ap-error-pid</output>
403     </pid>
404
405 <summer name="fcs/roll-summer">
406     <input> fcs/roll-ap-autoswitch </input>
407     <input> fcs/heading-ap-error-pid </input>
408     <output> fcs/roll-summer </output>
409 </summer>
410
411 <!-- Heading HOLD -->
412
413
414     <pure_gain name="fcs/aileron_roll">
415         <input> fcs/roll-summer </input>
416         <gain> ap/roll-gain </gain>
417         <output>ap/aileron_cmd</output>
418     </pure_gain>
419 </channel>
420
421 <!-- Rudder Roll Hold -->
422
423 <channel name="Rudder Roll control">
424

```

```

425 <summer name="fcs/roll-error-rud-pre">
426   <input> attitude/phi-rad</input>
427   <input> ap/bank.setpoint </input>
428   <output> fcs/roll-error-rud-pre </output>
429 </summer>
430
431 <lag_filter name="fcs/roll-error-rud-lag">
432   <input> fcs/roll-error-rud-pre </input>
433   <c1> ap/roll-lag </c1>
434   <output>fcs/roll-error-rud-lag</output>
435 </lag_filter>
436
437 <deadband name="fcs/rudder-windup-trigger">
438   <input> fcs/rudder-pos-deg </input>
439   <width>60.0</width>
440   <output>fcs/rudder-windup-trigger</output>
441 </deadband>
442
443 <deadband name="Rudder trigger roll">
444   <input> fcs/roll-error-rud-lag </input>
445   <width> .001 </width>
446   <output> fcs/roll-error-rud </output>]
447 </deadband>
448
449   <pid name="fcs/roll-ap-error-rud-pid">
450     <input> fcs/roll-error-rud </input>
451     <kp> ap/roll-pid-kp </kp>
452     <ki> ap/roll-pid-ki </ki>
453     <kd> ap/roll-pid-kd </kd>
454     <trigger> fcs/rudder-windup-trigger </trigger>
455     <clipto>
456       <min>-1.0</min>
457       <max> 1.0</max>
458     </clipto>
459     <output>fcs/roll-ap-error-rud-pid</output>
460   </pid>
461
462   <switch name="fcs/roll-rud-ap-autoswitch">
463     <default value="0.0"/>
464     <test value="fcs/roll-ap-error-rud-pid">
465       ap/bank_hold_rud == 1
466     </test>
467     <output>fcs/roll-rud-ap-autoswitch</output>
468   </switch>
469
470   <pure_gain name="ap/rudder-cmd-roll-norm">
471     <input> fcs/roll-rud-ap-autoswitch </input>
472     <gain> ap/roll-gain </gain>
473     <output> ap/rudder-cmd-roll-norm </output>
474   </pure_gain>
475
476 </channel>
477
478

```

```

479 <!-- Throttle Roll Hold -->
480
481 <channel name="Throttle Roll control">
482
483   <summer name="fcs/roll-error-thr-pre">
484     <input> attitude/phi-rad</input>
485     <input> -ap/bank-setpoint </input>
486     <output> fcs/roll-error-thr-pre </output>
487   </summer>
488
489   <deadband name="Throttle trigger roll">
490     <input> fcs/roll-error-thr-pre </input>
491     <width> 0.035 </width>
492     <output> fcs/roll-error-thr </output>
493   </deadband>
494
495   <deadband name="fcs/throttle-windup-trigger">
496     <input> fcs/throttle-pos-norm[1] </input>
497     <width> 1.0 </width>
498   </deadband>
499
500   <pid name="fcs/roll-ap-error-thr-pid">
501     <input> fcs/roll-error-thr </input>
502     <kp> ap/roll-pid-kp </kp>
503     <ki> ap/roll-pid-ki </ki>
504     <kd> ap/roll-pid-kd </kd>
505     <trigger> fcs/throttle-windup-trigger </trigger>
506     <clipto>
507       <min> -0.6</min>
508       <max> 0.6</max>
509     </clipto>
510     <output> fcs/roll-ap-error-thr-pid</output>
511   </pid>
512
513   <switch name="fcs/roll-thr-ap-autoswitch">
514     <default value="0.0"/>
515     <test value="fcs/roll-ap-error-thr-pid">
516       ap/bank_hold.throt == 1
517     </test>
518   </switch>
519
520   <pure_gain name="fcs/throttle.roll">
521     <input> fcs/roll-thr-ap-autoswitch </input>
522     <gain> ap/roll-gain </gain>
523     <output> ap/throttle-cmd-roll-norm </output>
524   </pure_gain>
525
526   <summer name="fcs/roll-thr-summer">
527     <input> ap/throttle-cmd-roll-norm </input>
528     <input> fcs/throttle-cmd-norm[1] </input>
529     <clipto>
530       <min> 0</min>
531       <max> 1.0</max>
532     </clipto>

```

```

533     <output>fcs/roll-thr-summer</output>
534 </summer>
535
536 <switch name="throttle switch">
537     <default value="fcs/throttle-cmd-norm[1]" />
538     <test value="fcs/roll-thr-summer">
539         ap/bank_hold_throt == 1
540     </test>
541     <output> fcs/throttle-pos-norm[1] </output> <!-- Roll with one engine -->
542 </switch>
543
544 </channel>
545
546
547
548
549
550 <!--
551 =====
552 PITCH HOLD
553 =====
554 -->
555
556 <!-- Elevator Pitch Hold -->
557
558 <channel name="Theta Hold Elevator">
559     <summer name="fcs/roll-error-pre">
560         <input> attitude/theta-rad</input>
561         <input> -ap/theta_setpoint </input>
562         <output> fcs/theta_error </output>
563     </summer>
564
565     <switch name="fcs/theta_hold-elevator">
566         <default value="0" />
567         <test value="fcs/theta_error">
568             ap/theta_hold_elev == 1
569         </test>
570         <output> fcs/theta_hold-elevator </output>
571     </switch>
572
573     <pid name="fcs/theta-ap-error-pid">
574         <input> fcs/theta_hold-elevator</input>
575         <kp> ap/theta-pid-kp </kp>
576         <ki> ap/theta-pid-ki </ki>
577         <kd> ap/theta-pid-kd </kd>
578         <clipto>
579             <min> -0.1</min>
580             <max> 0.1</max>
581         </clipto>
582         <output> ap/theta_hold_pid </output>
583     </pid>
584
585     <pure_gain name="fcs/elevator-pitch">
586         <input> ap/theta_hold_pid </input>

```

```
587         <gain> 1.0 </gain>
588         <output>ap/elevator.cmd-pitch</output>
589     </pure_gain>
590 </channel>
591
592 </autopilot>
```

Appendix F. Supporting Figures

This appendix contains supporting figures for Chapter IV.

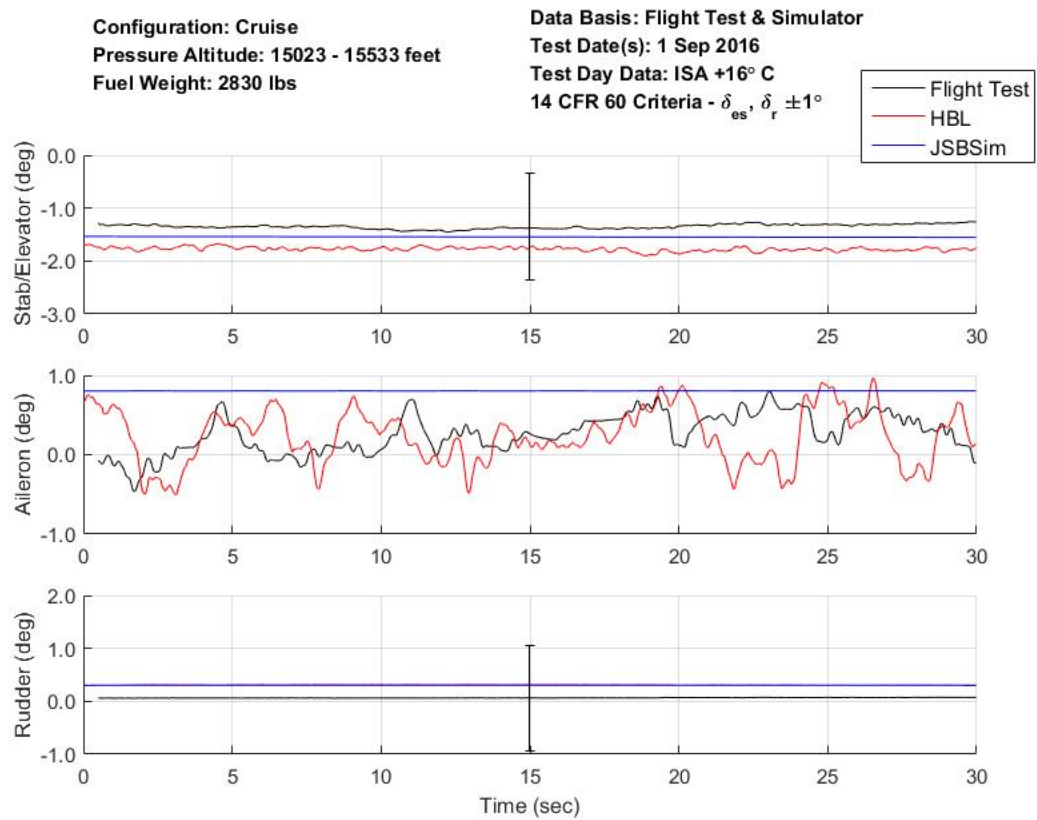


Figure F.1. Nominal Configuration (1000 fpm): Control Deflections

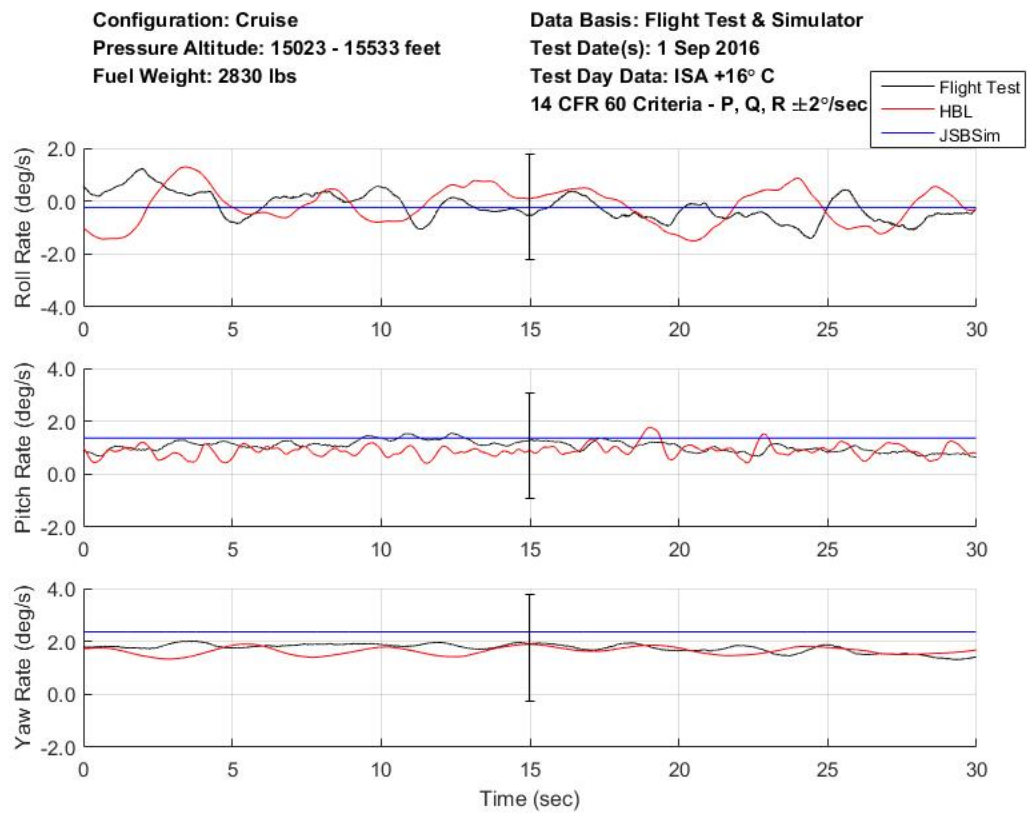


Figure F.2. Nominal Configuration (1000 fpm): Body-Axis Rates

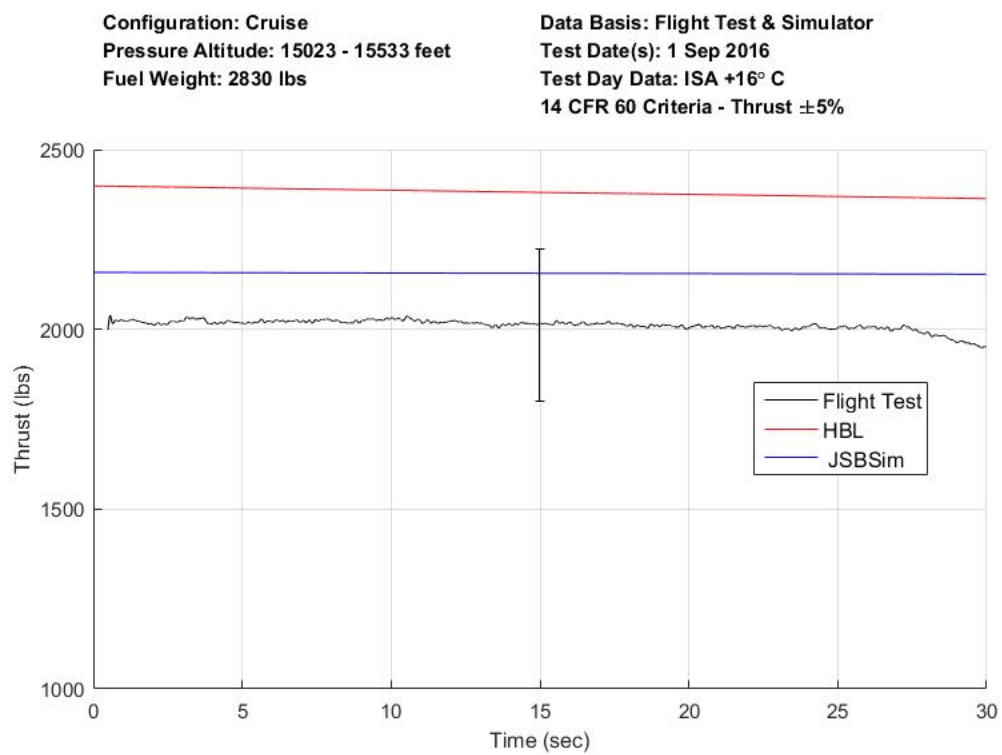


Figure F.3. Nominal Configuration (1000 fpm): Thrust

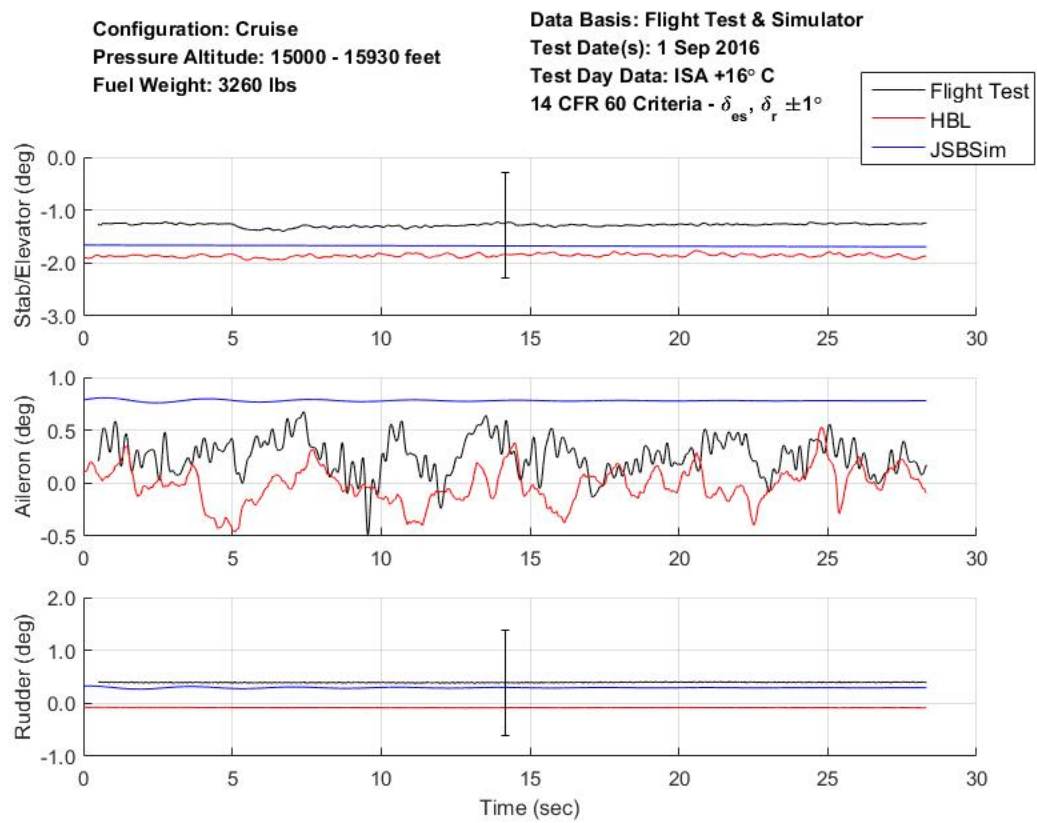


Figure F.4. Nominal Configuration (2000 fpm): Control Deflections

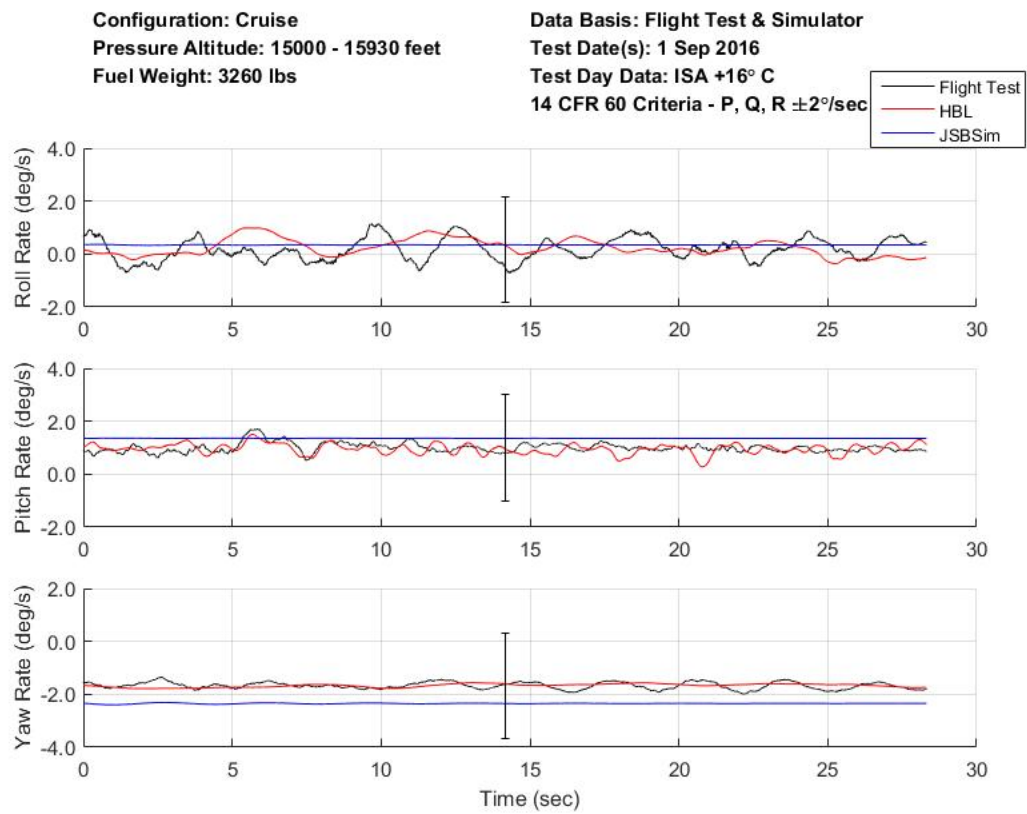


Figure F.5. Nominal Configuration (2000 fpm): Body-Axis Rates

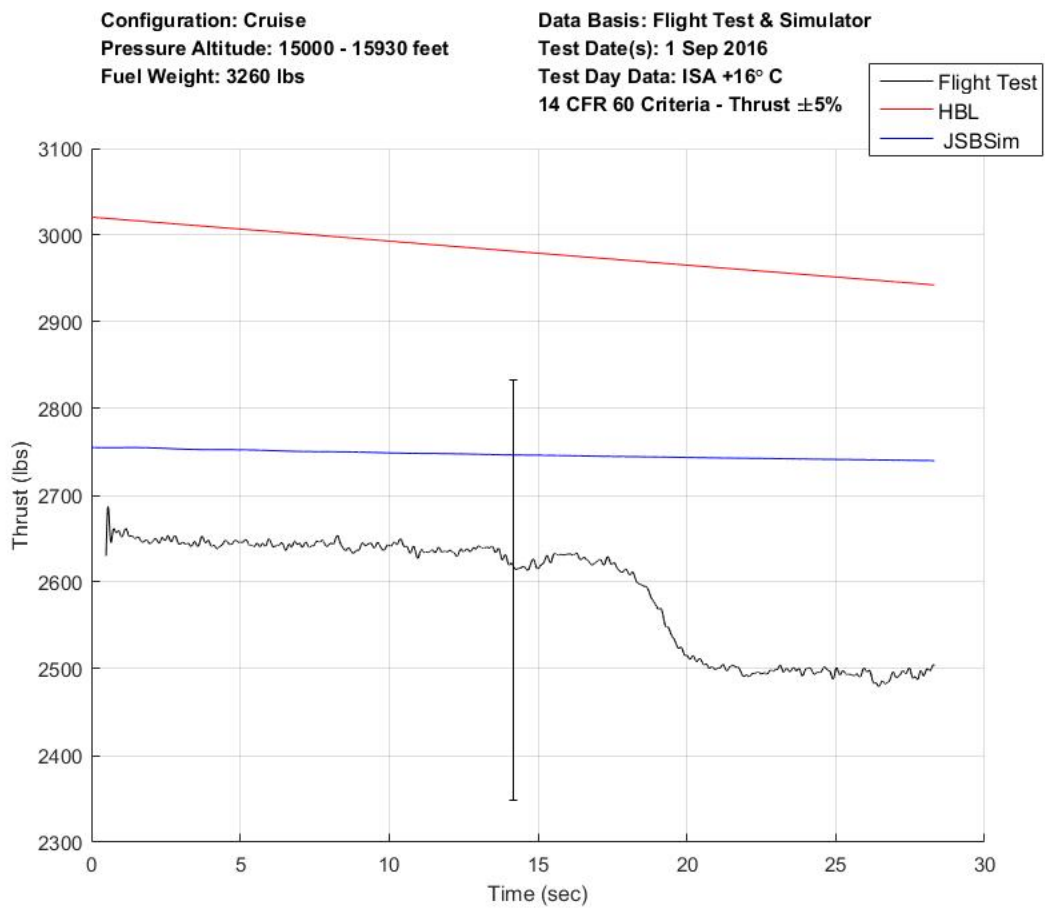


Figure F.6. Nominal Configuration (2000 fpm): Thrust

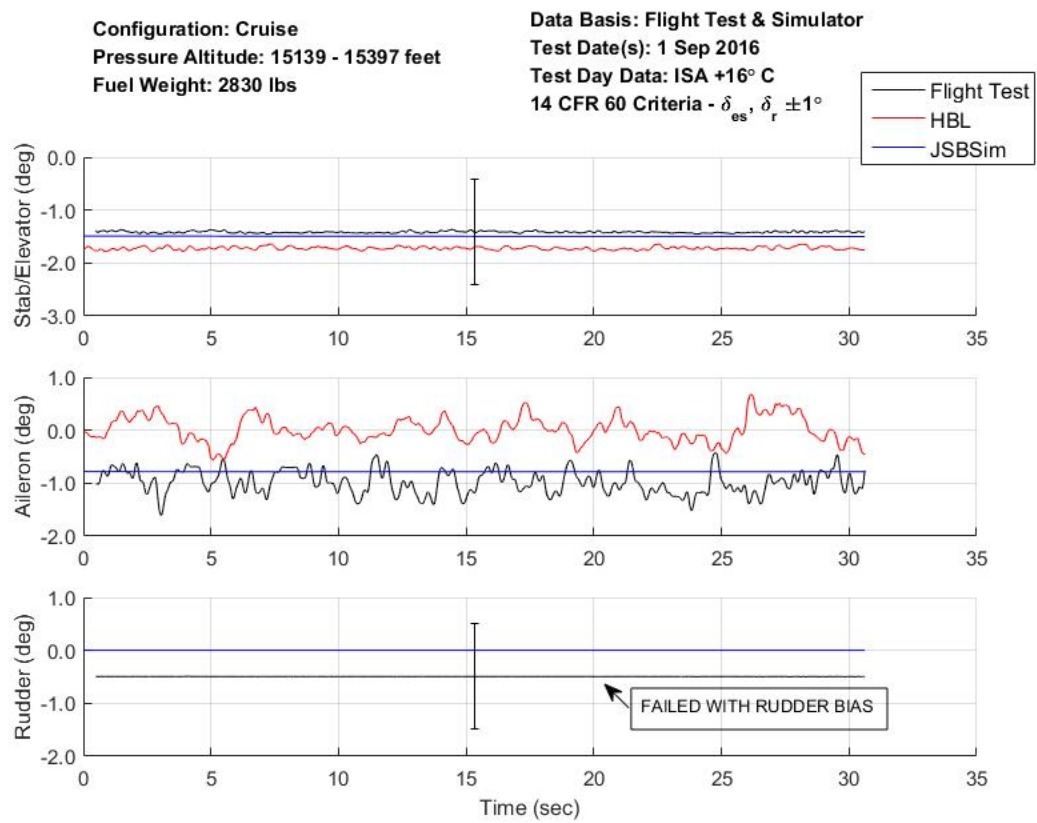


Figure F.7. Rudder Failed Configuration (500 fpm): Control Deflections

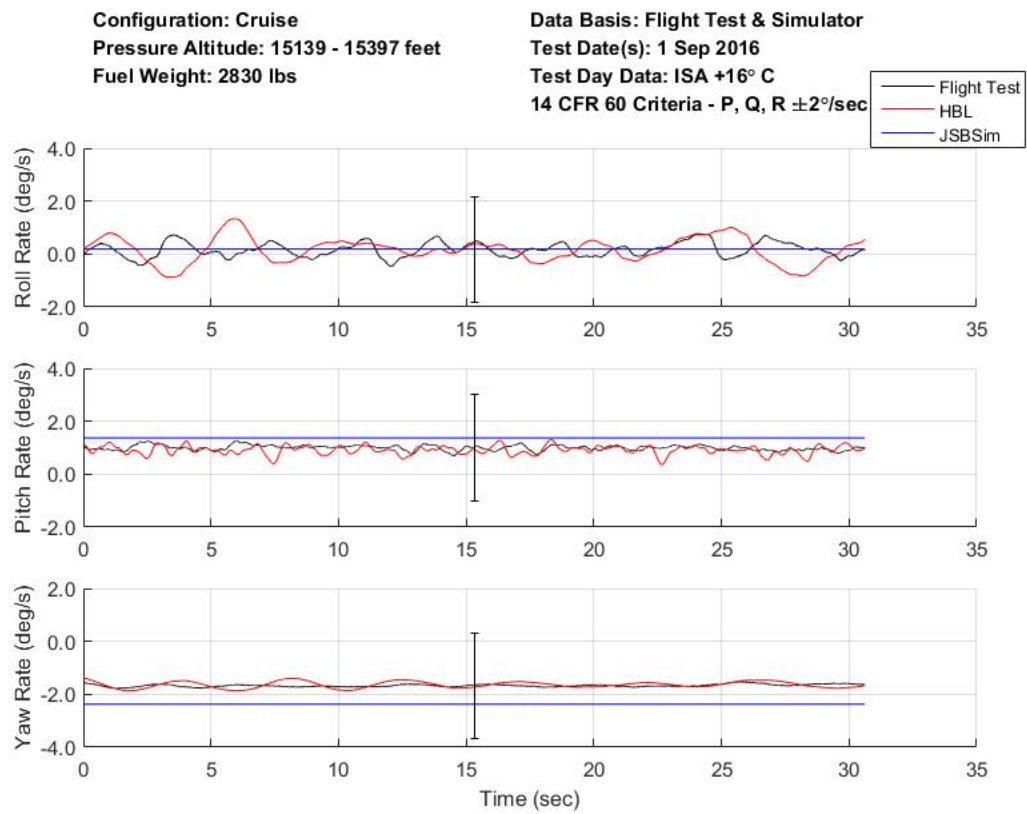


Figure F.8. Rudder Failed Configuration (500 fpm): Body-Axis Rates

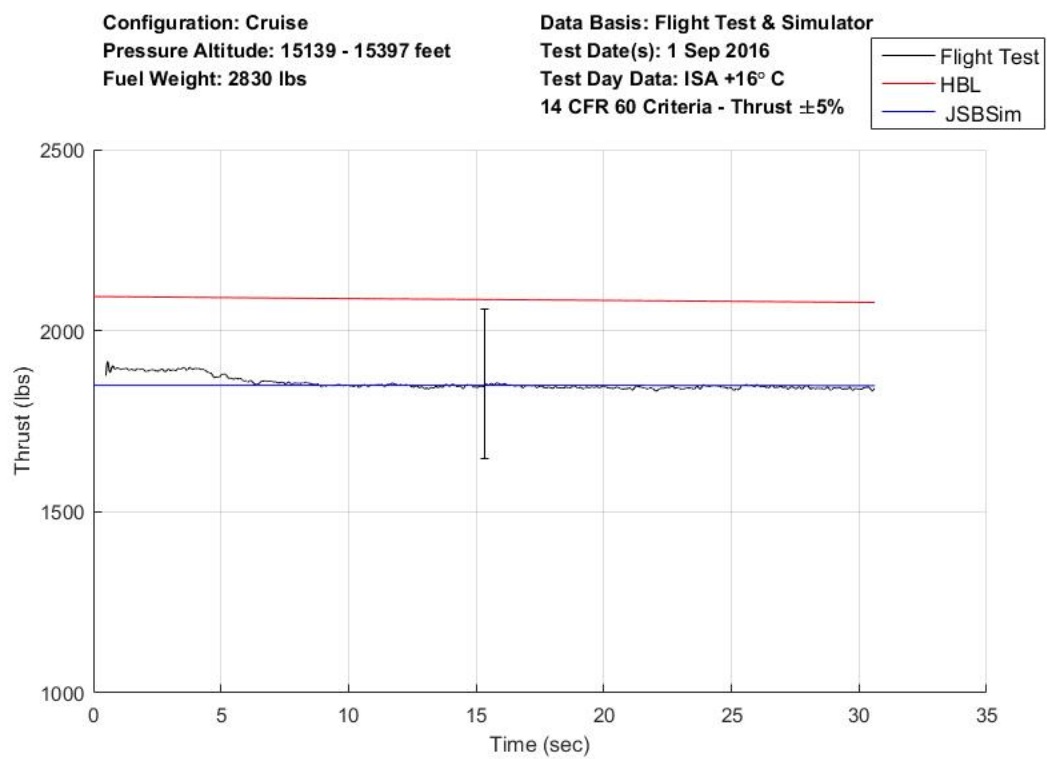


Figure F.9. Rudder Failed Configuration (500 fpm): Thrust

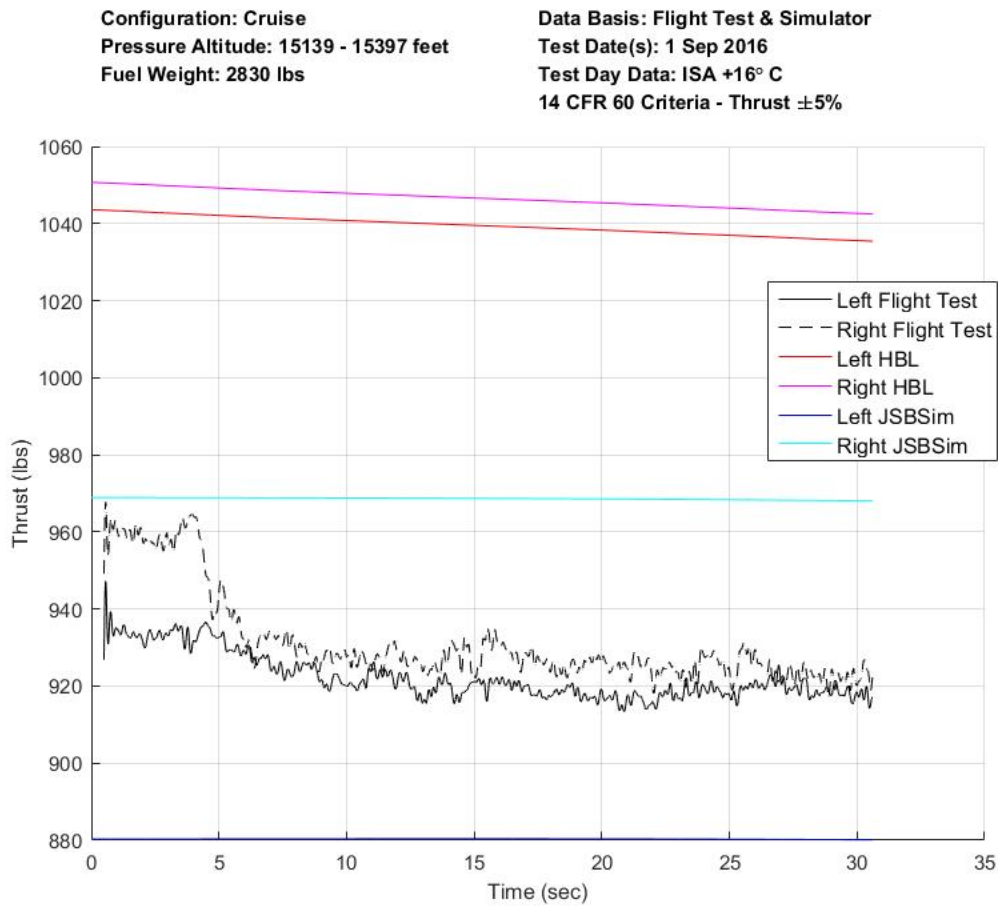


Figure F.10. Rudder Failed Configuration (500 fpm): Individual Engine Thrust

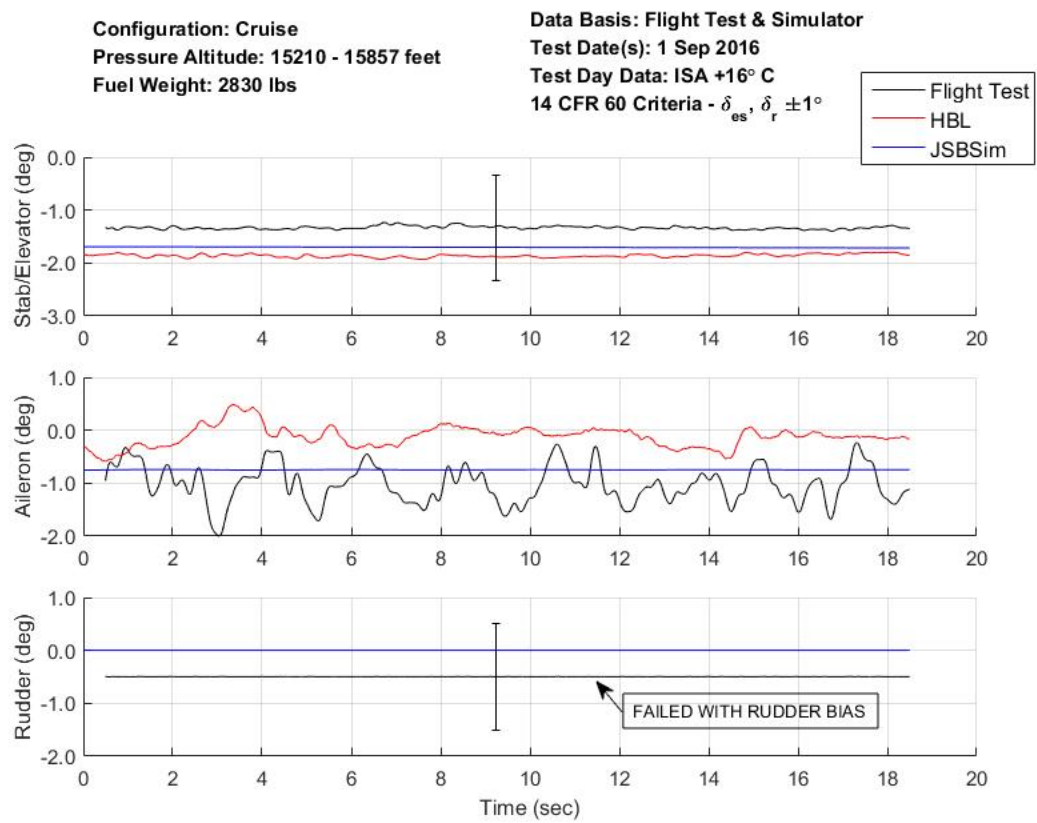


Figure F.11. Rudder Failed Configuration (2000 fpm): Control Deflections

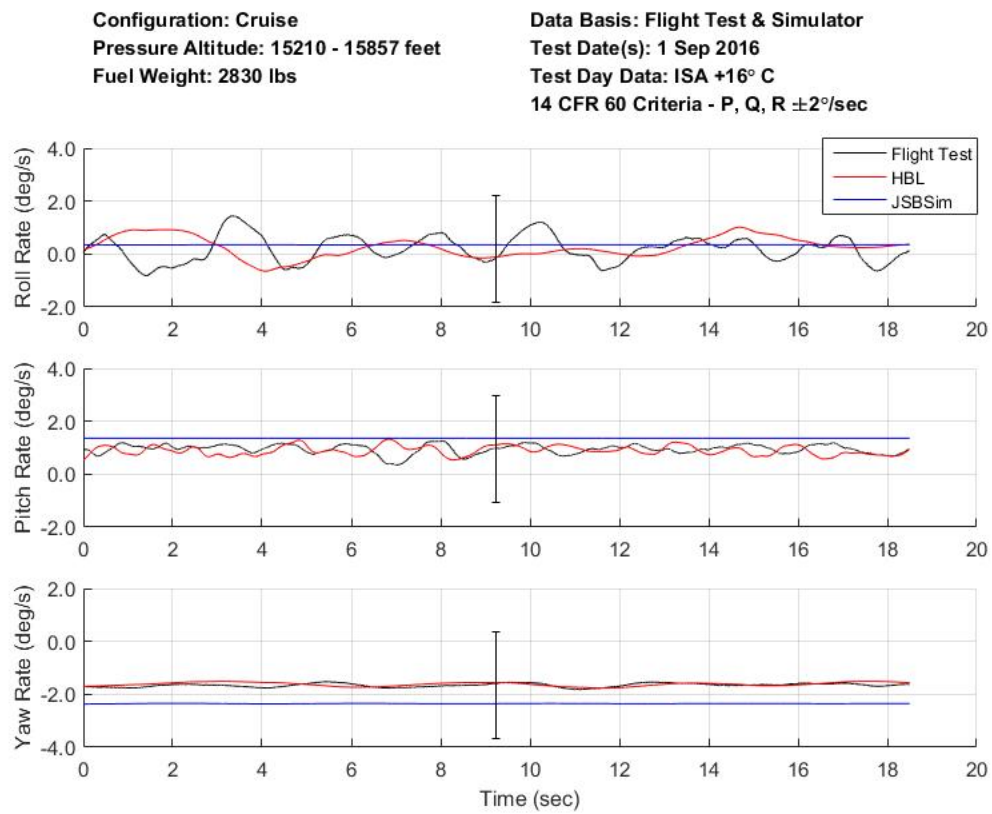


Figure F.12. Rudder Failed Configuration (2000 fpm): Body-Axis Rates

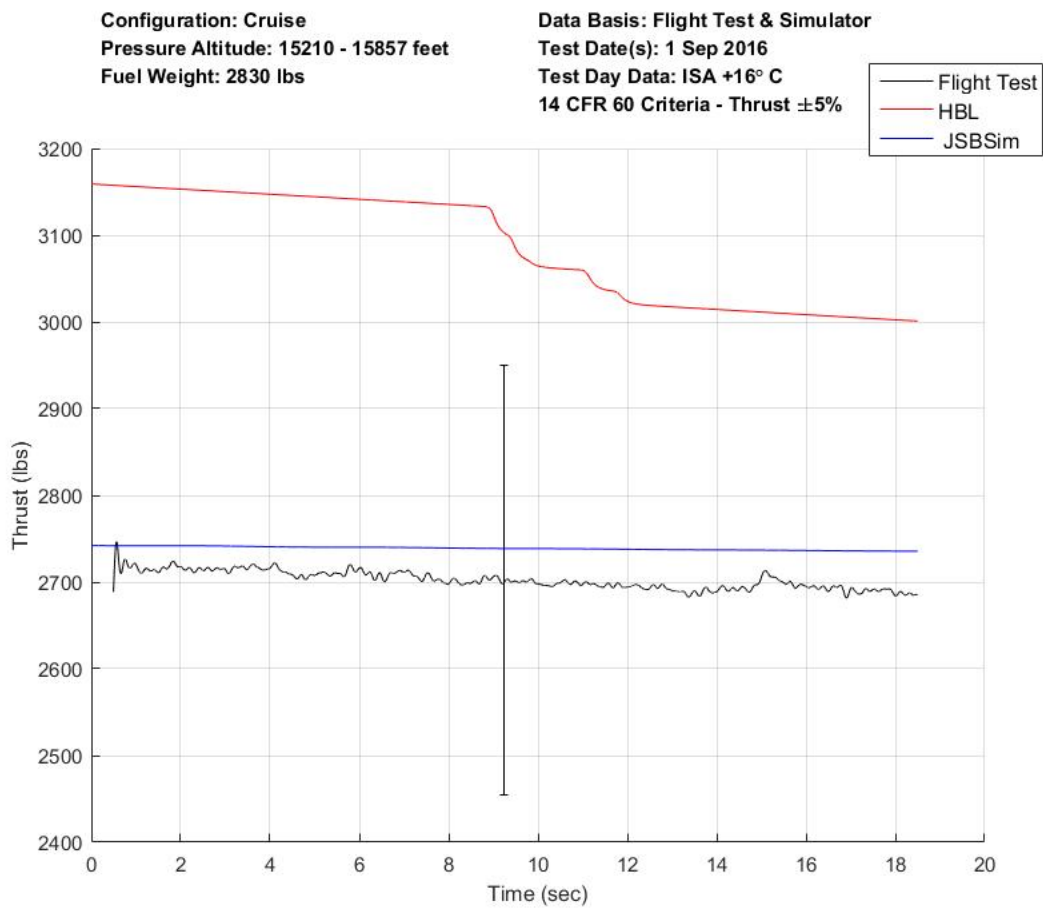


Figure F.13. Rudder Failed Configuration (2000 fpm): Thrust

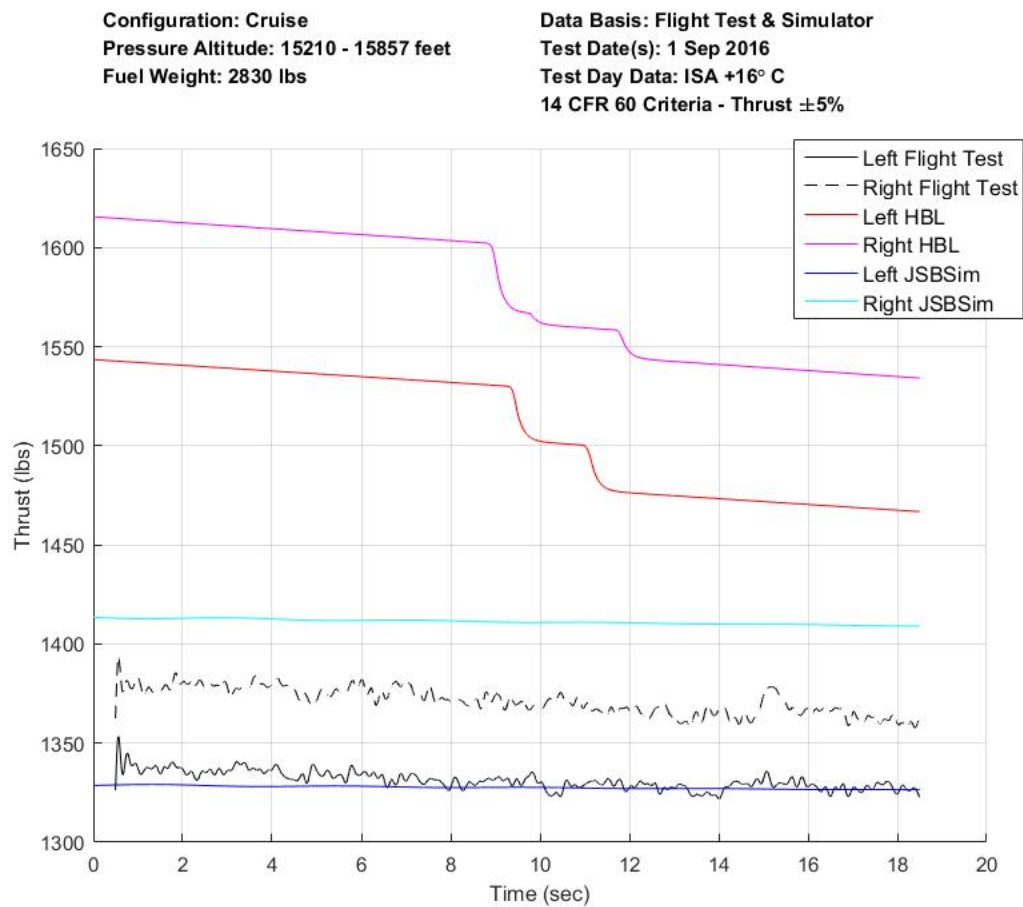


Figure F.14. Rudder Failed Configuration (2000 fpm): Individual Engine Thrust

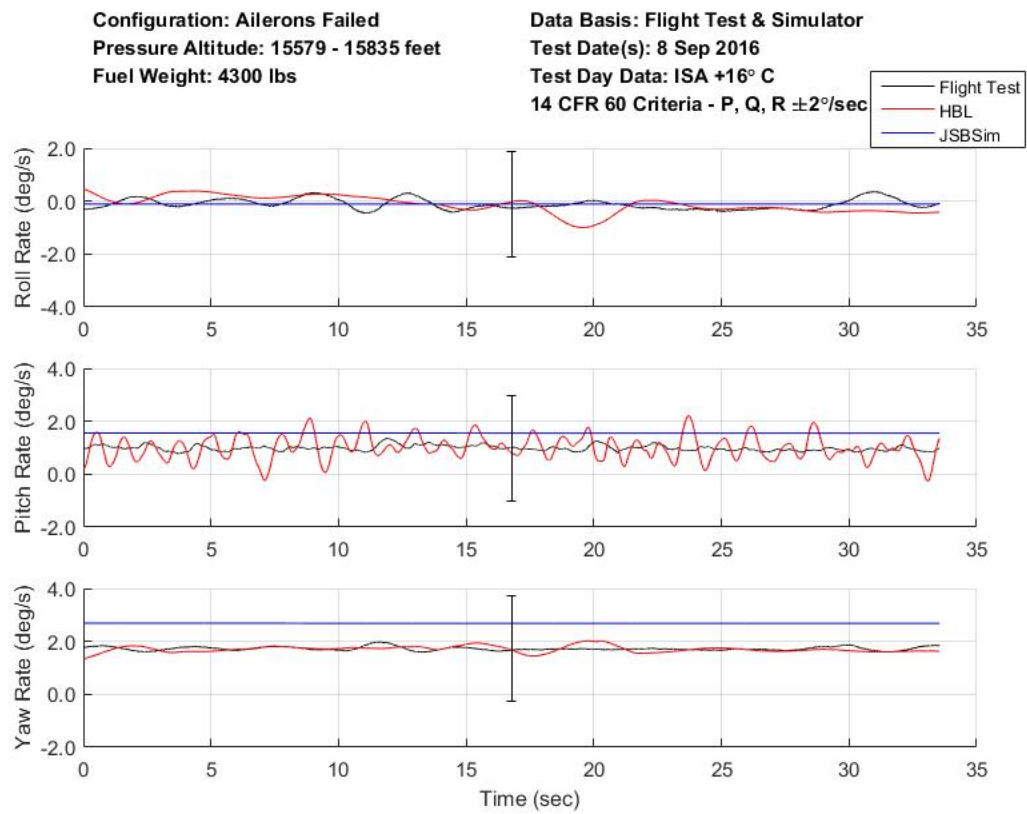


Figure F.15. Aileron Failed Configuration (500 fpm): Rates

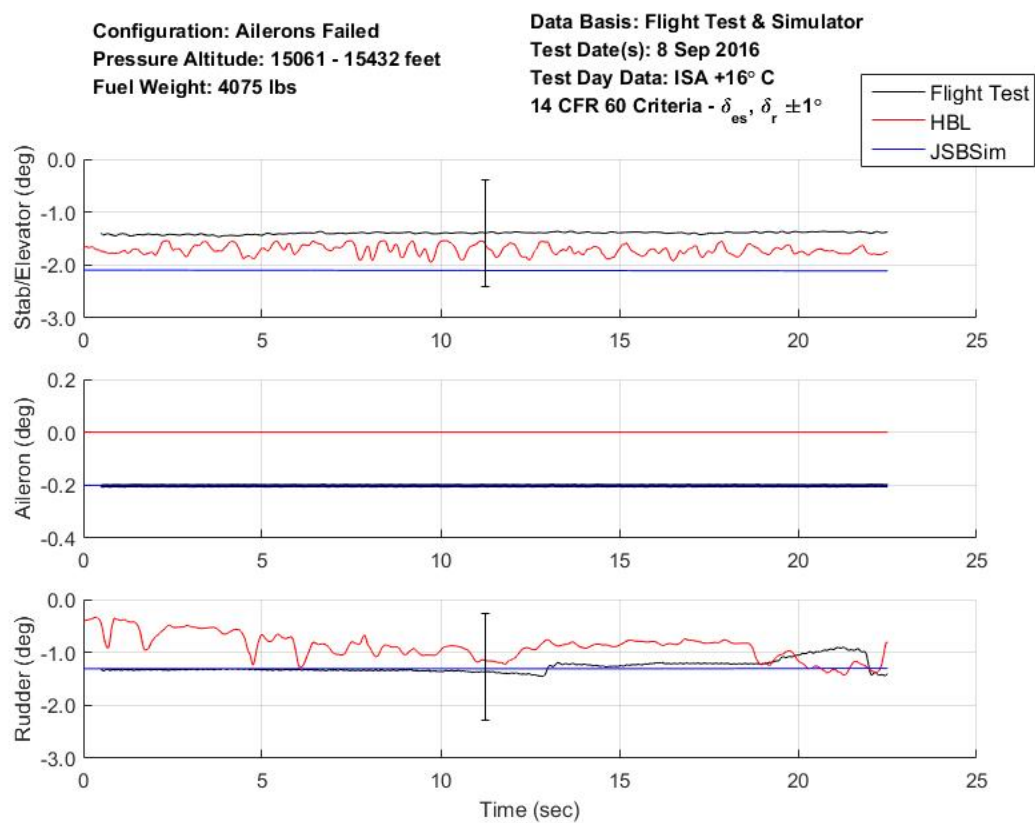


Figure F.16. Aileron Failed Configuration (1000 fpm): Control Deflections

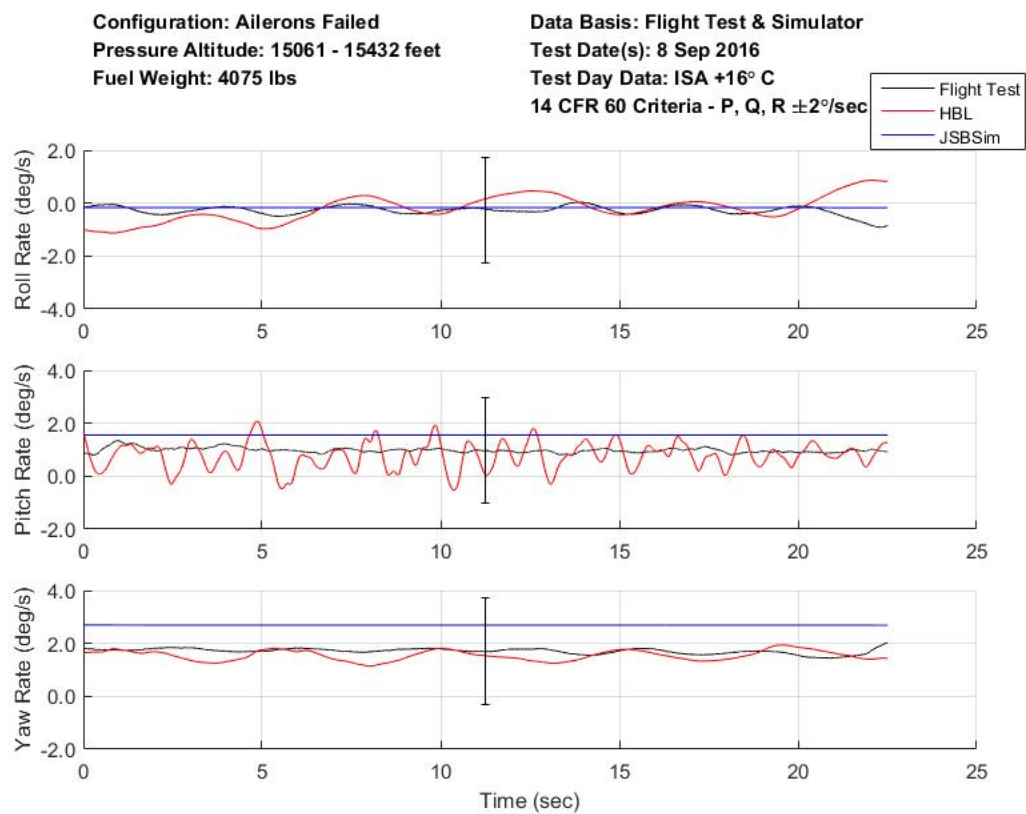


Figure F.17. Aileron Failed Configuration (1000 fpm): Rates

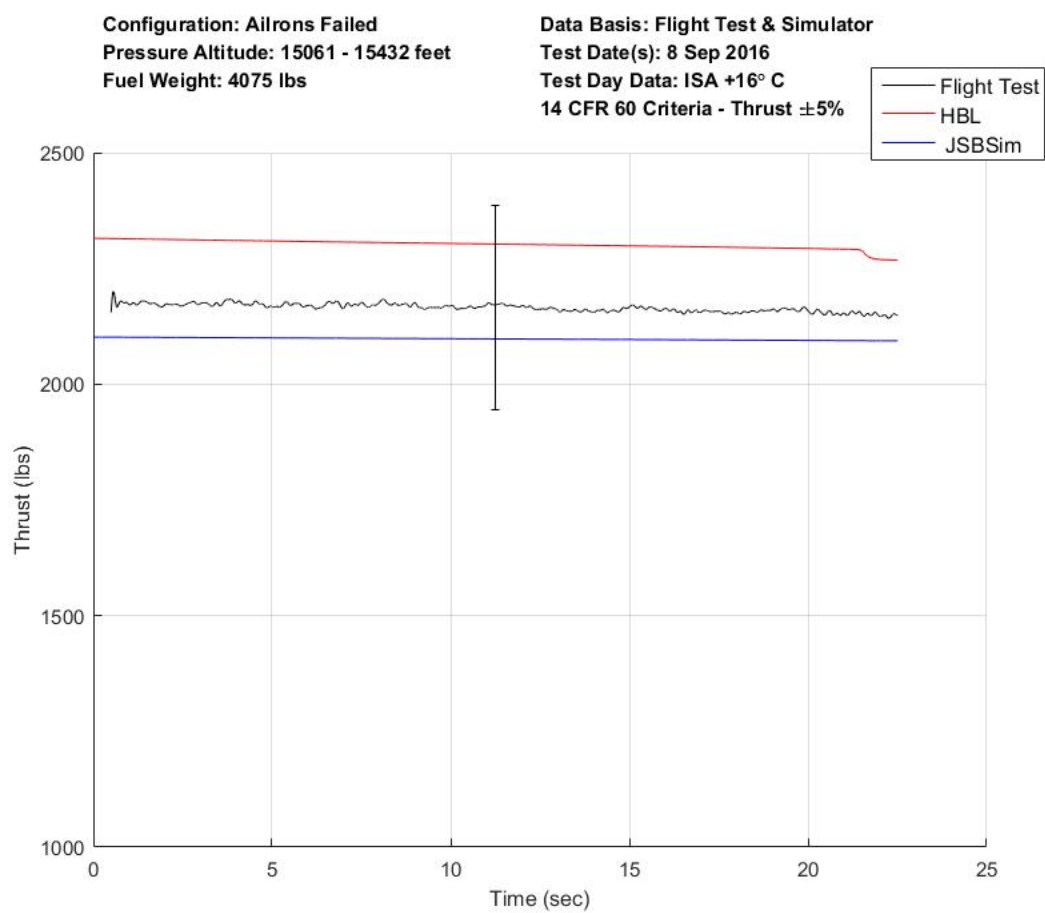


Figure F.18. Aileron Failed Configuration (1000 fpm): Thrust

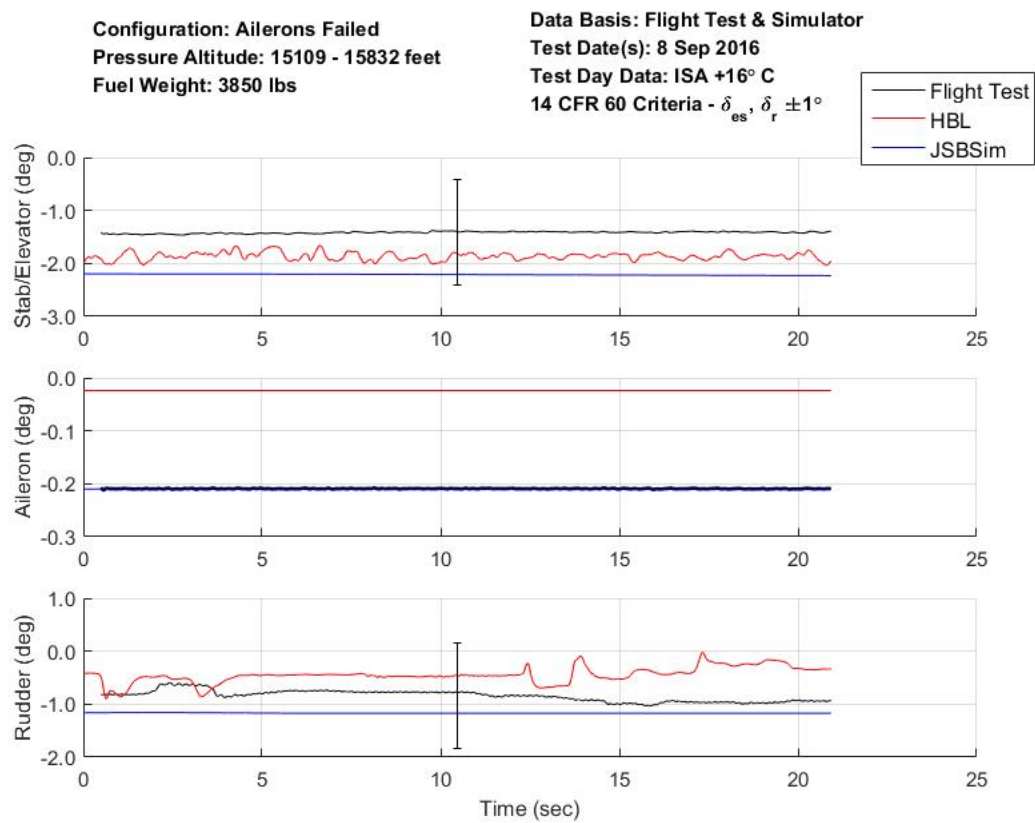


Figure F.19. Aileron Failed Configuration (2000 fpm): Control Deflections

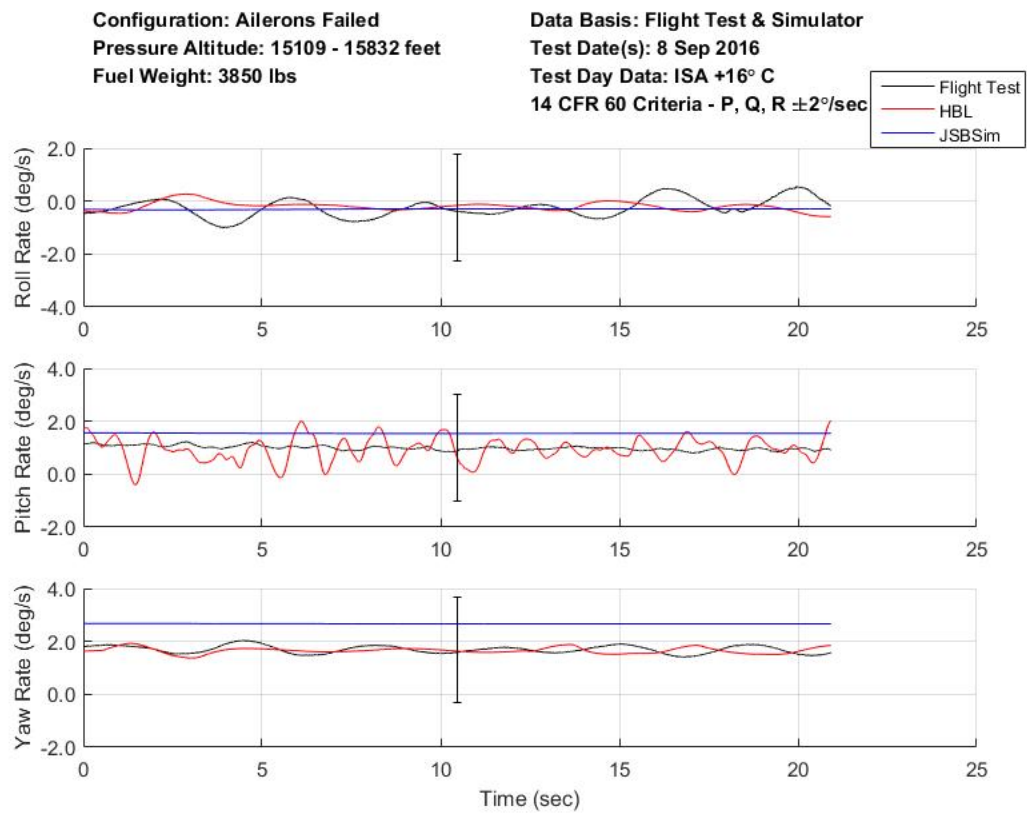


Figure F.20. Aileron Failed Configuration (2000 fpm): Rates

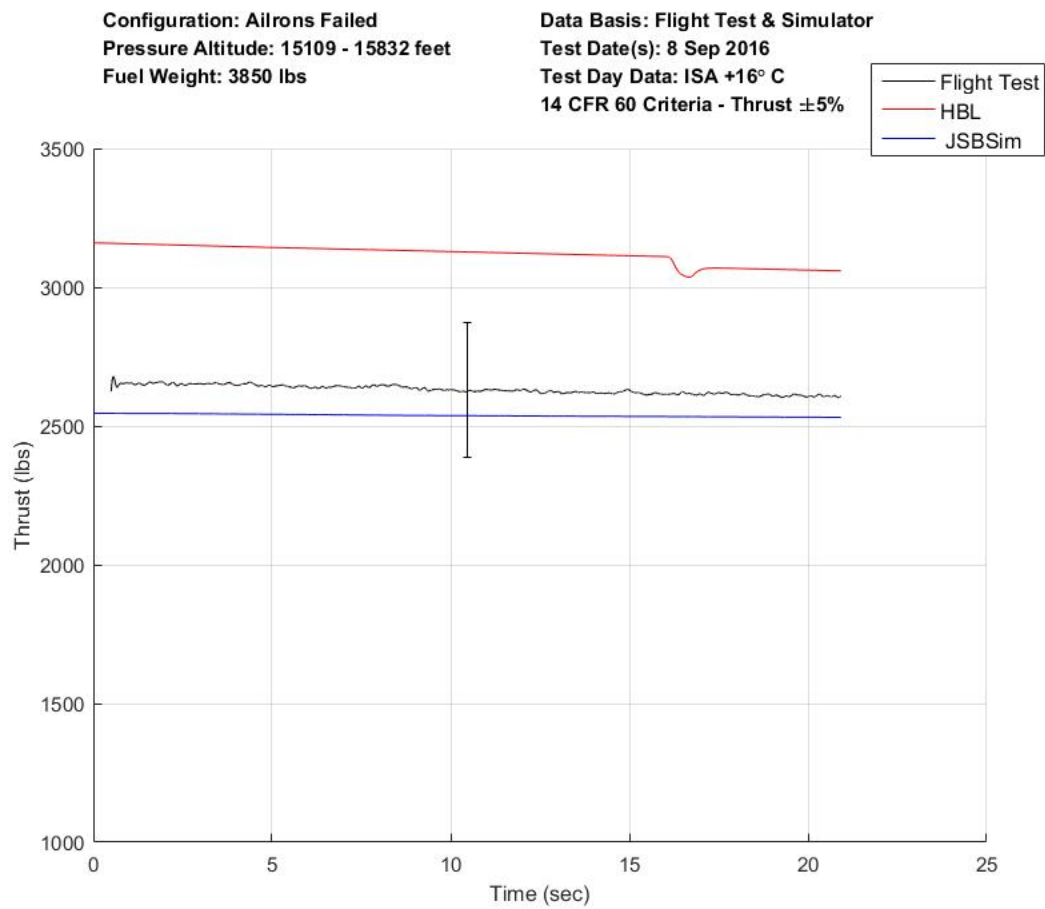


Figure F.21. Aileron Failed Configuration (2000 fpm): Thrust

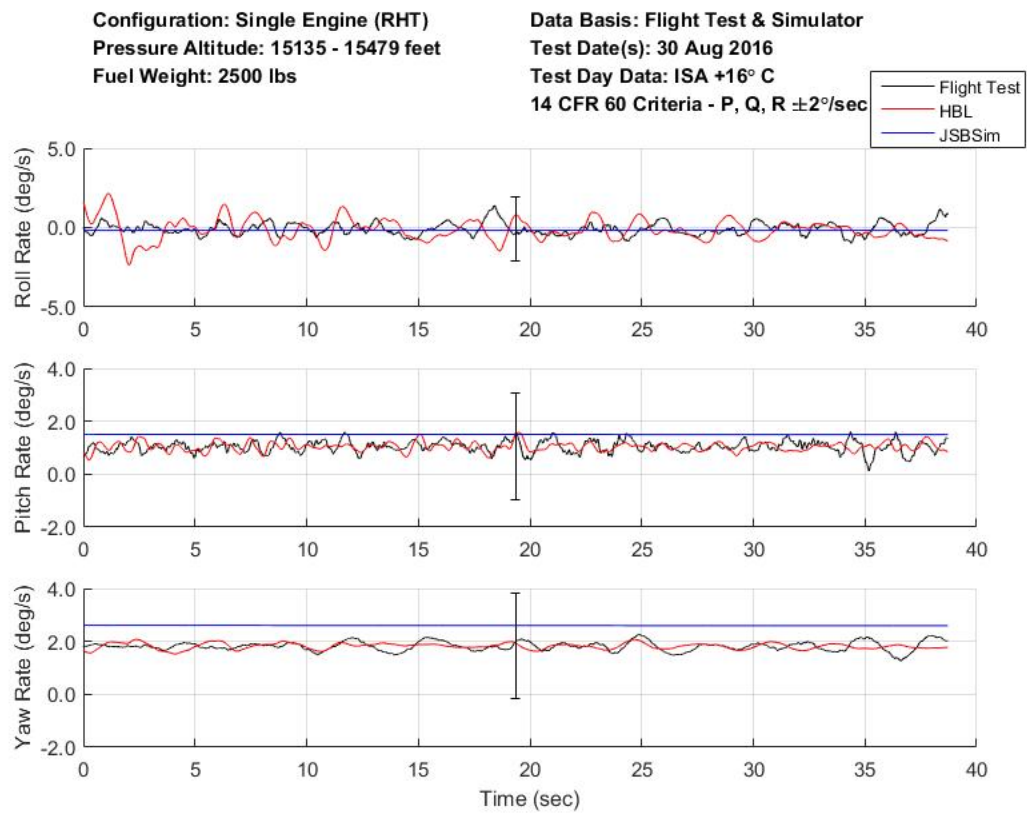


Figure F.22. Single Engine Right-Hand Turn into Idle Engine (500 fpm): Rates

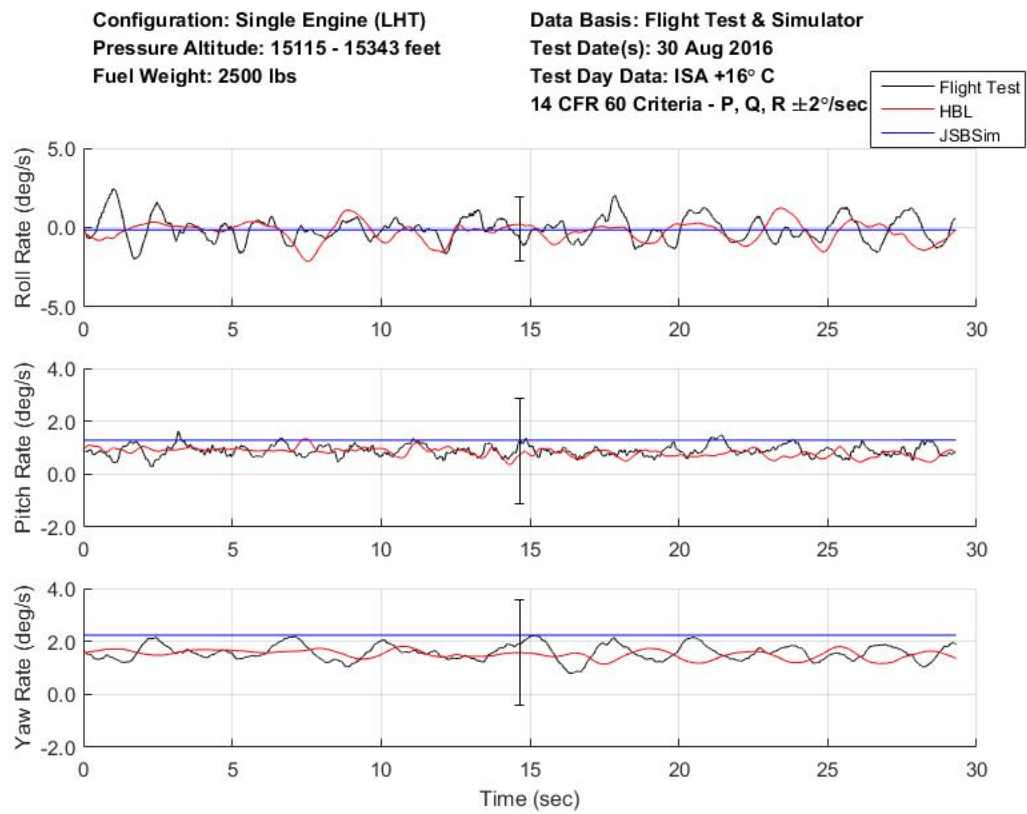


Figure F.23. Single Engine Left-Hand Turn away from Idle Engine (500 fpm): Rates

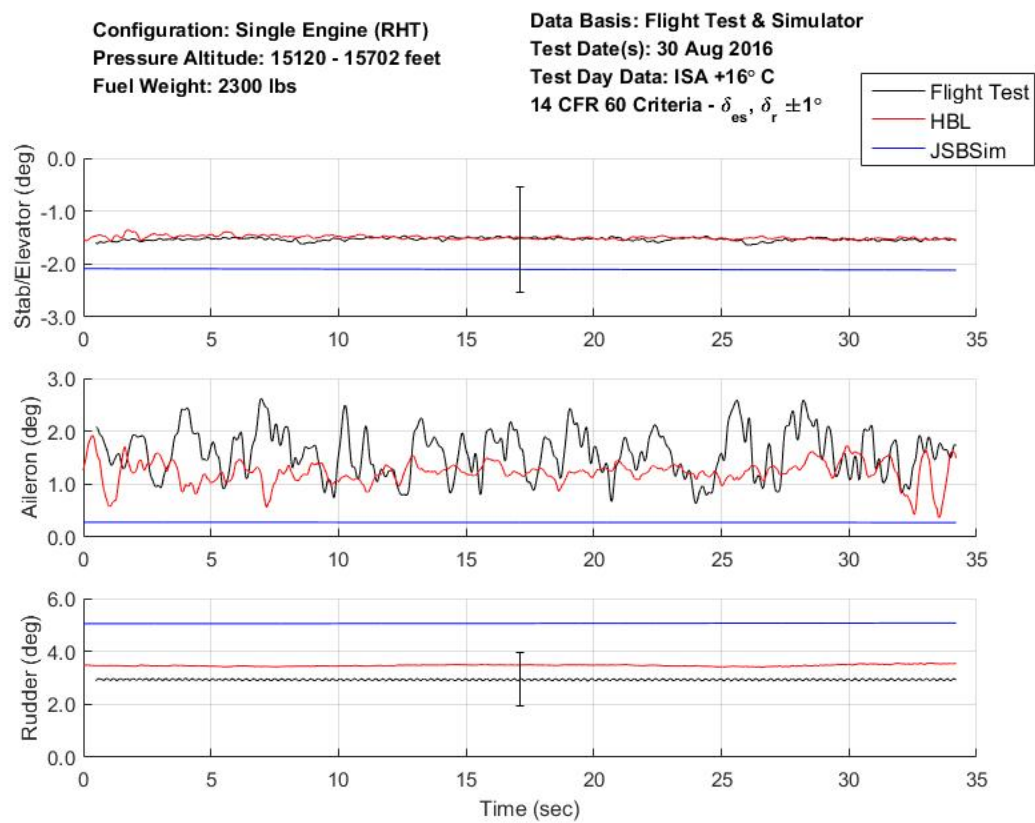


Figure F.24. Single Engine Right-Hand Turn into Idle Engine (1000 fpm): Control Deflections

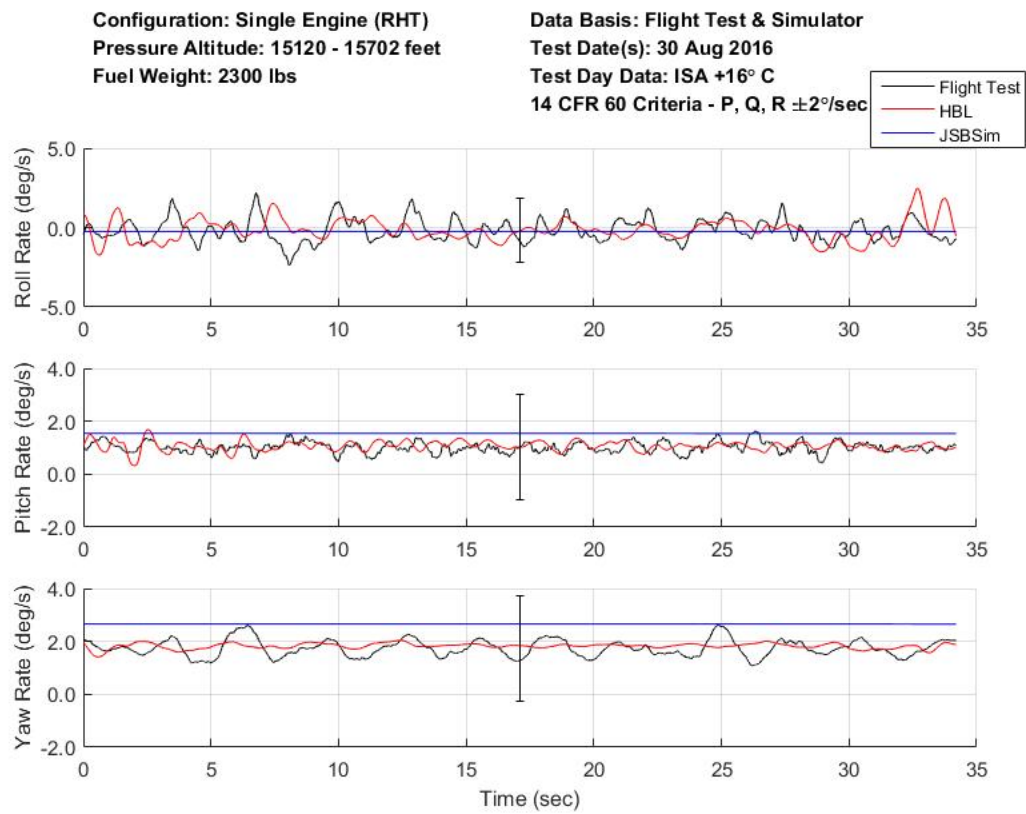


Figure F.25. Single Engine Right-Hand Turn into Idle Engine (1000 fpm): Rates

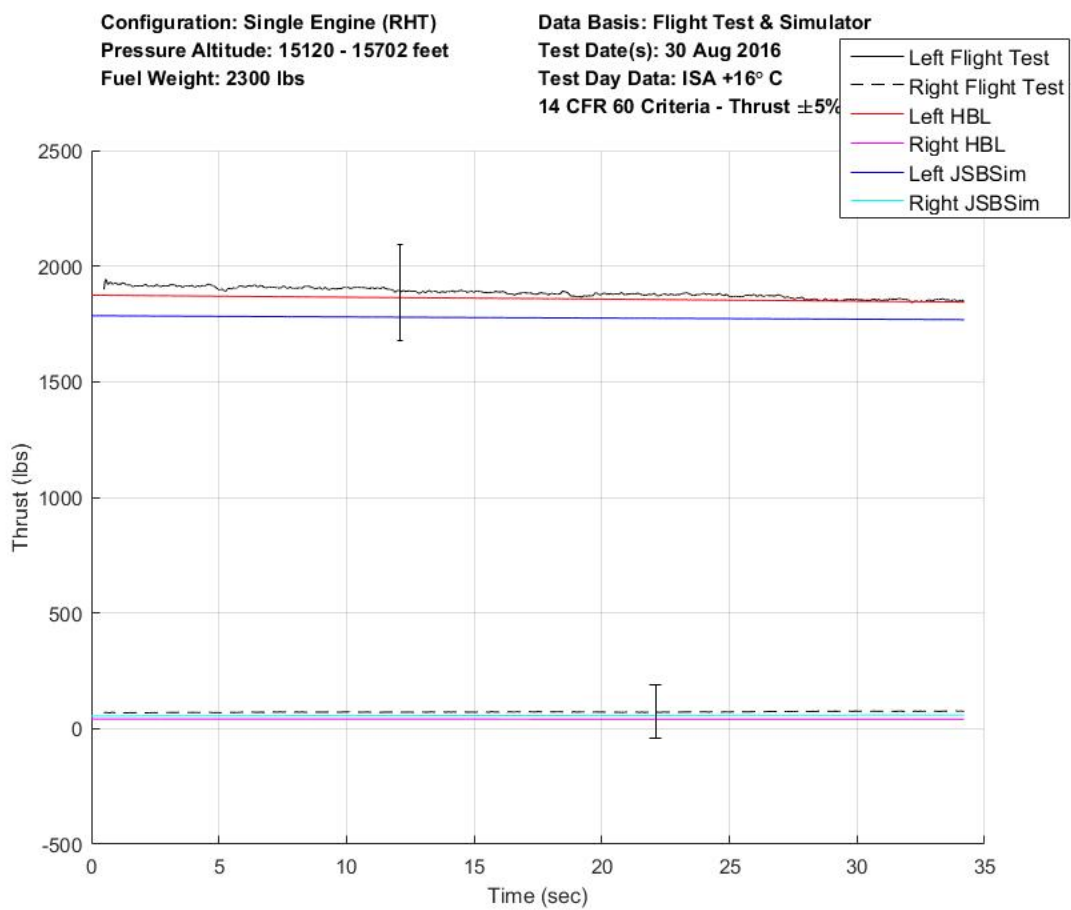


Figure F.26. Single Engine Right-Hand Turn into Idle Engine (1000 fpm): Individual Engine Thrust

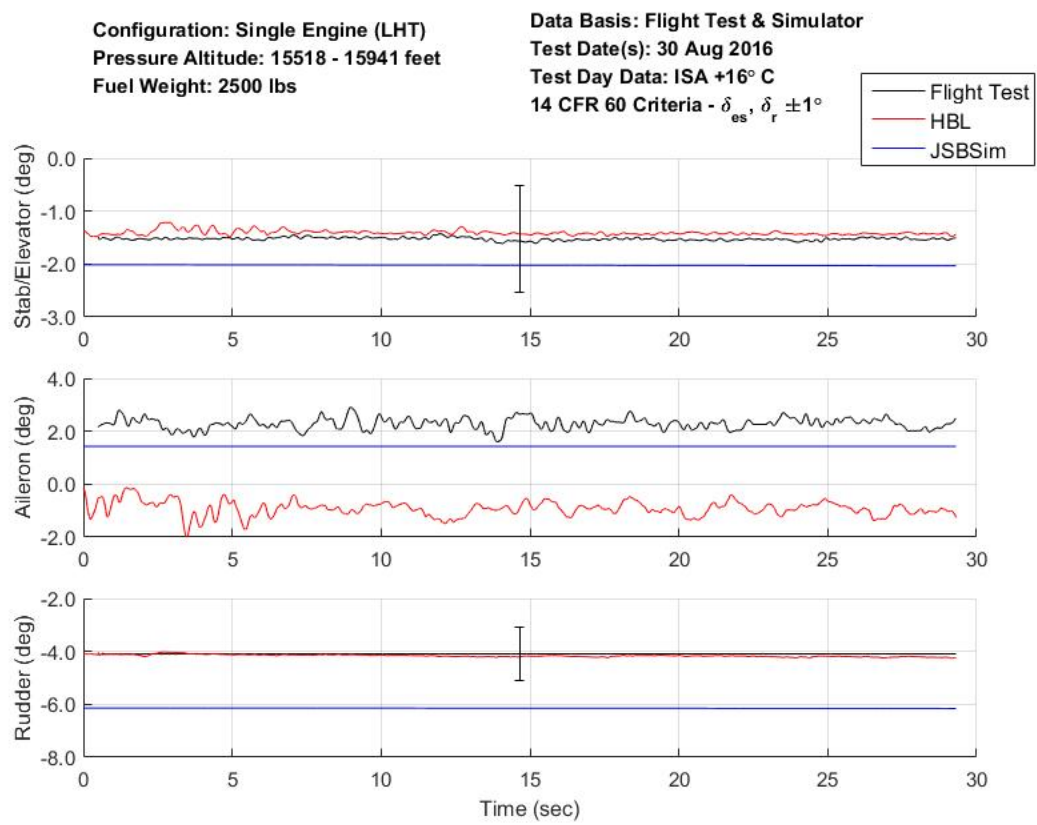


Figure F.27. Single Engine Left-Hand Turn away from Idle Engine (1000 fpm): Control Deflections

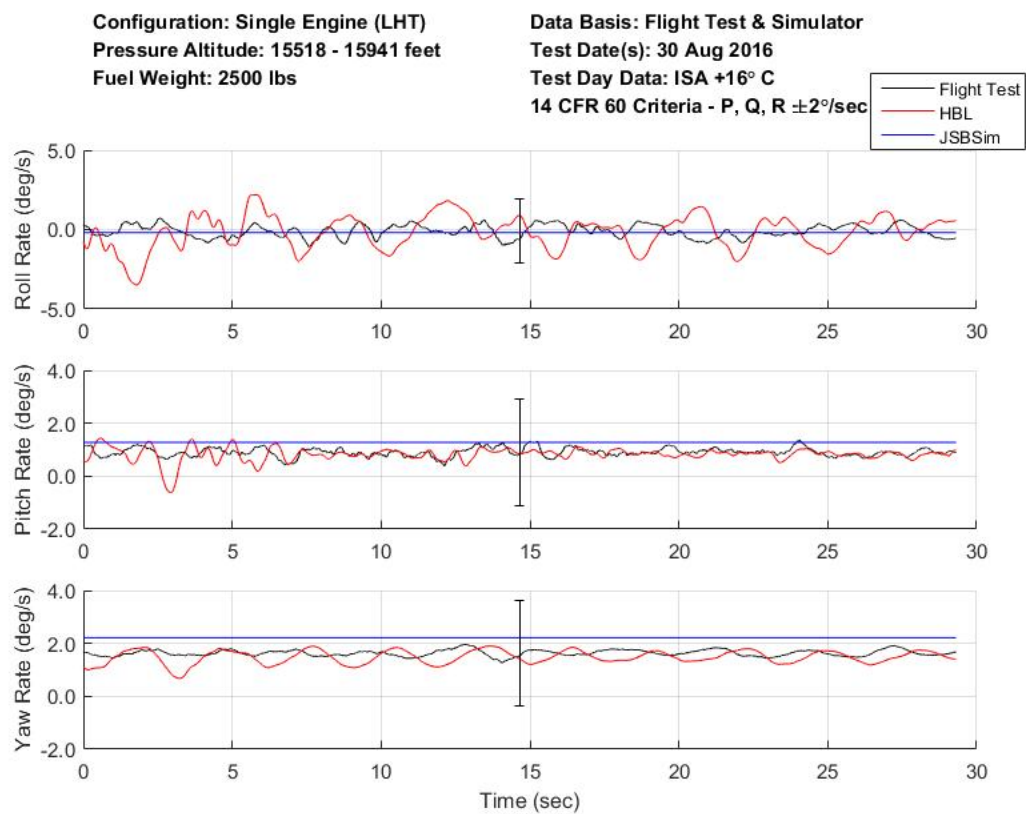


Figure F.28. Single Engine Left-Hand Turn away Idle Engine (1000 fpm): Rates

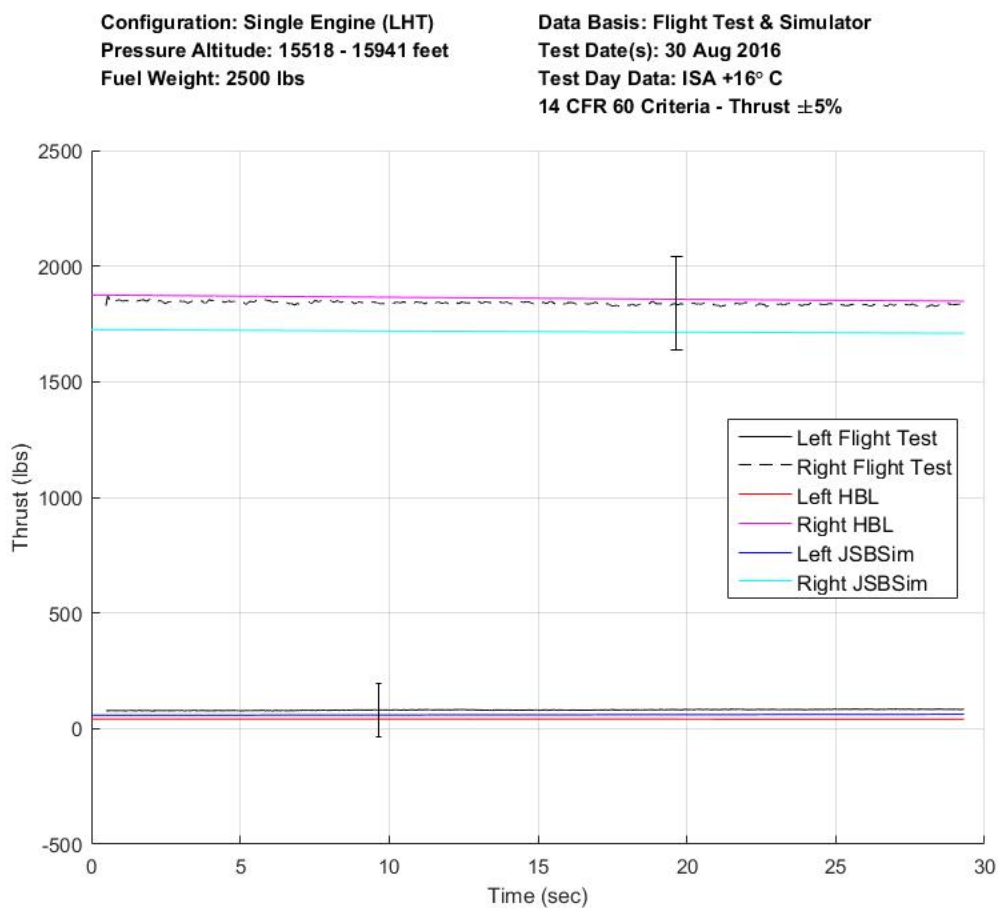


Figure F.29. Single Engine Left-Hand Turn away from Idle Engine (1000 fpm): Individual Engine Thrust

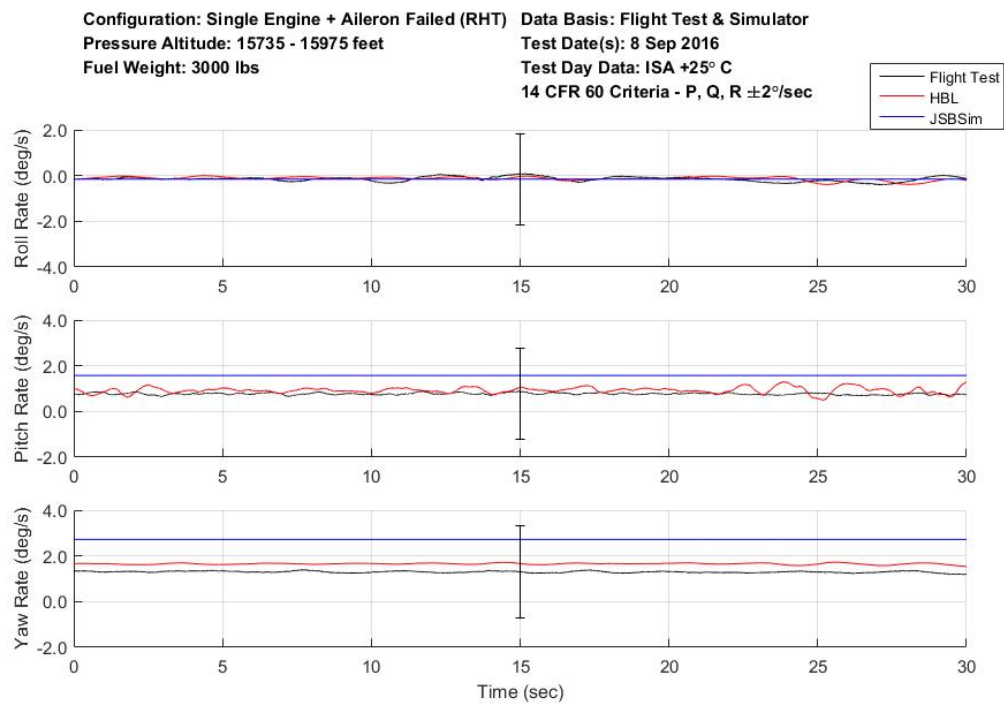


Figure F.30. Single Engine and Aileron Failed (RHT) at 500 fpm: Rates

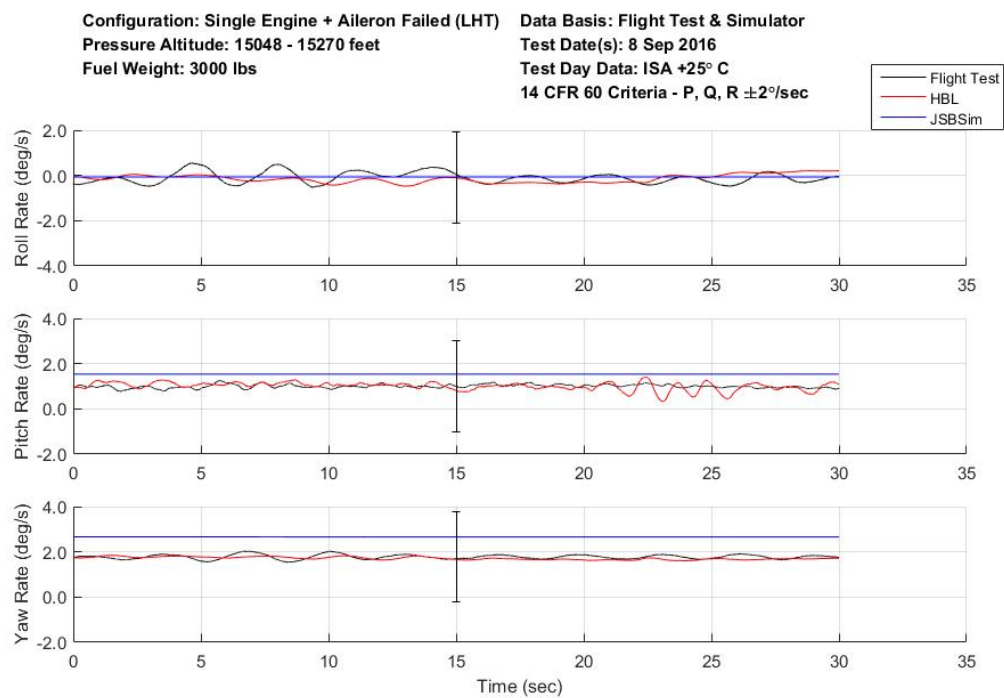


Figure F.31. Single Engine and Aileron Failed (LHT) at 500 fpm: Rates

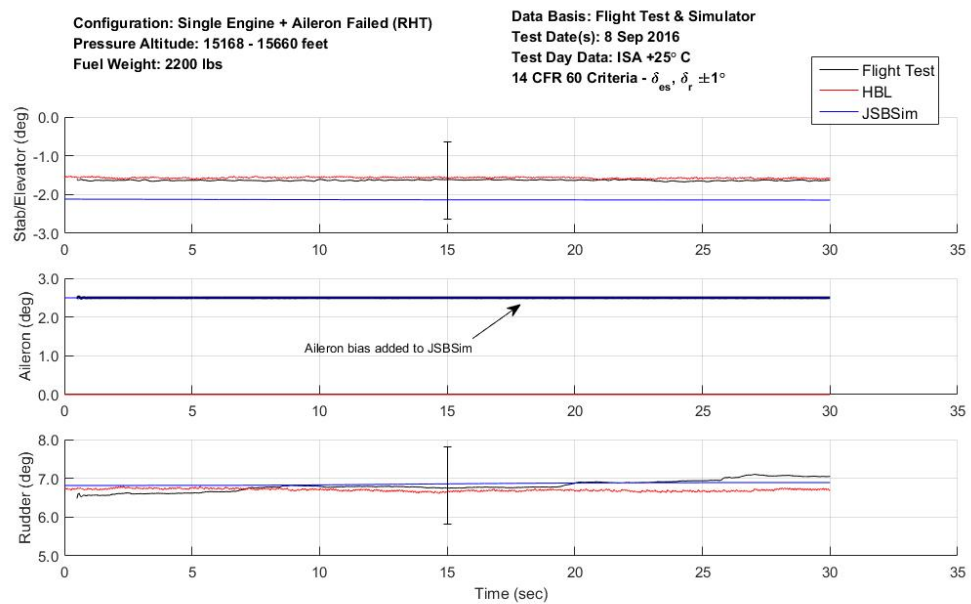


Figure F.32. Single Engine and Aileron Failed (RHT) at 1000 fpm: Control Deflections

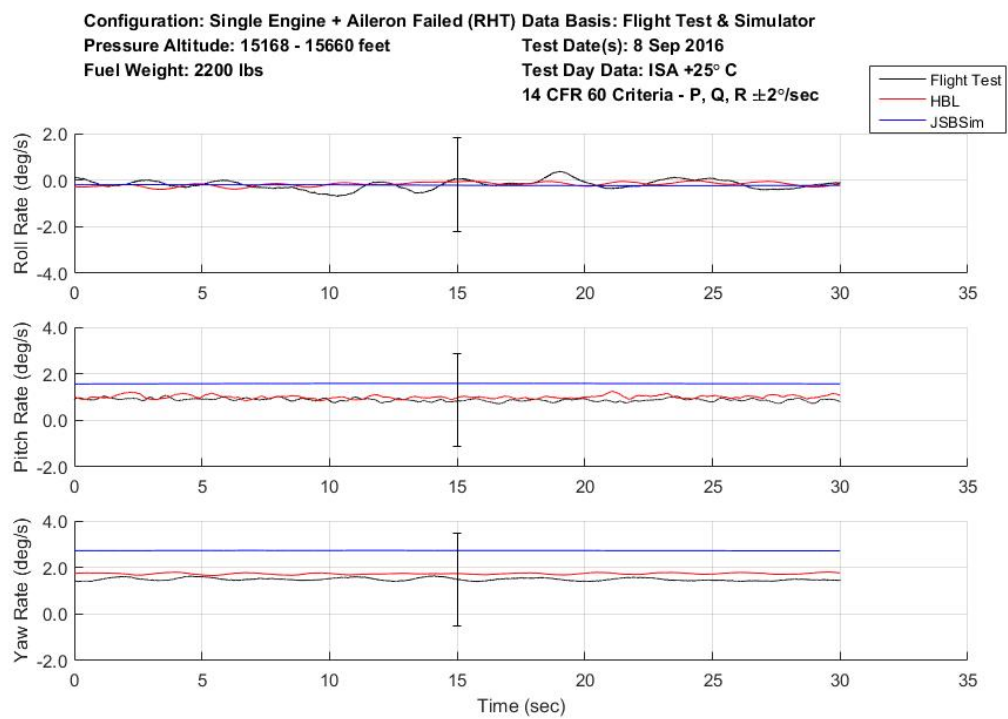


Figure F.33. Single Engine and Aileron Failed (RHT) at 1000 fpm: Rates

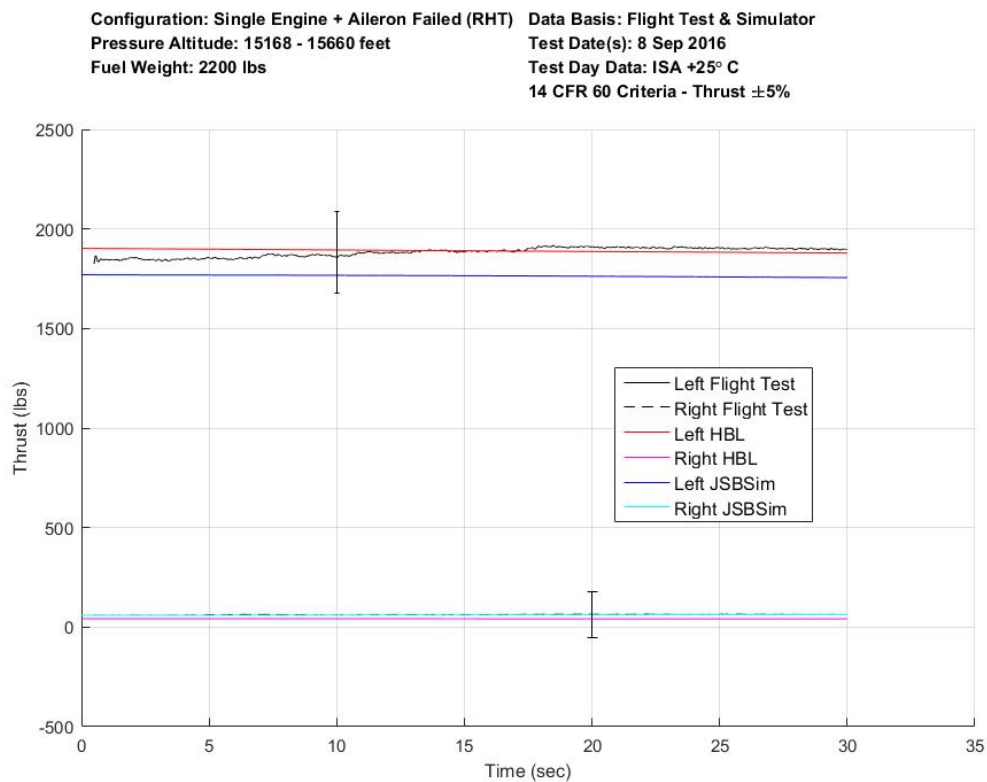


Figure F.34. Single Engine and Aileron Failed (RHT) at 1000 fpm: Split Thrust

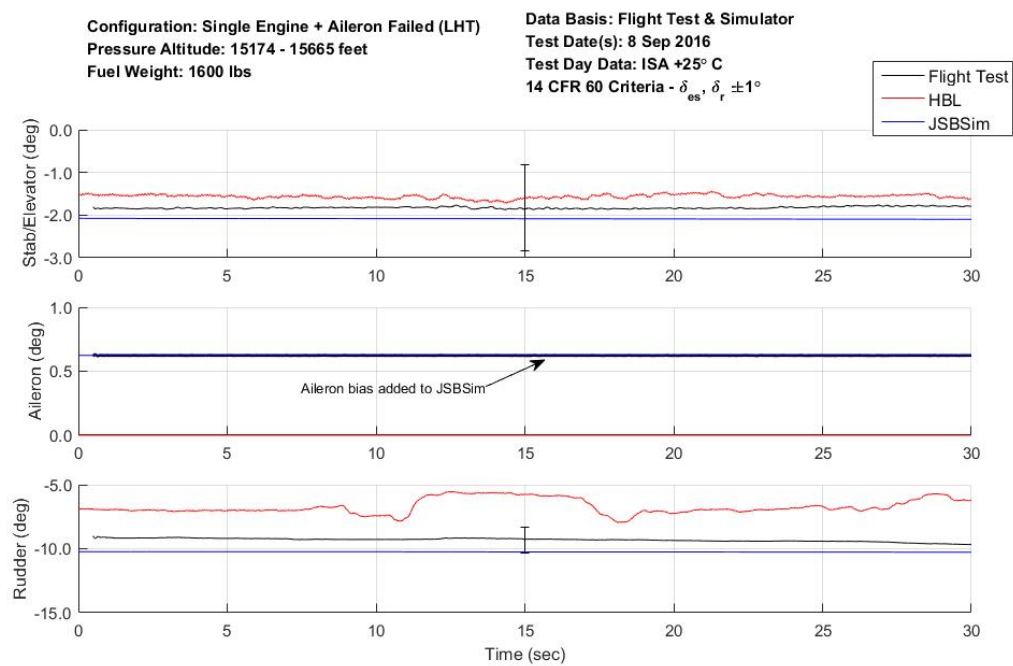


Figure F.35. Single Engine and Aileron Failed (LHT) at 1000 fpm: Control Deflections

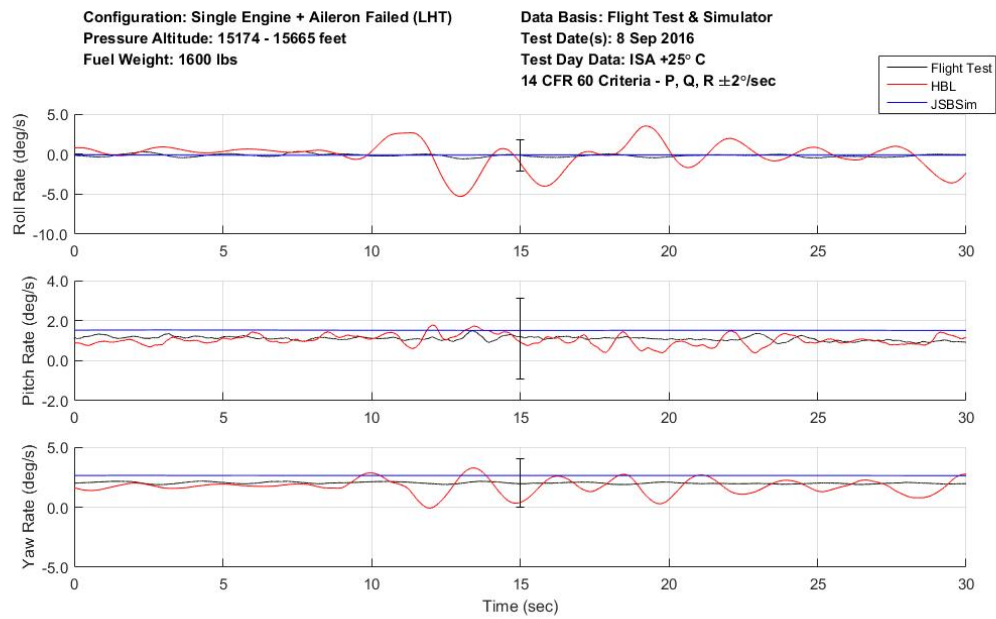


Figure F.36. Single Engine and Aileron Failed (LHT) at 1000 fpm: Rates

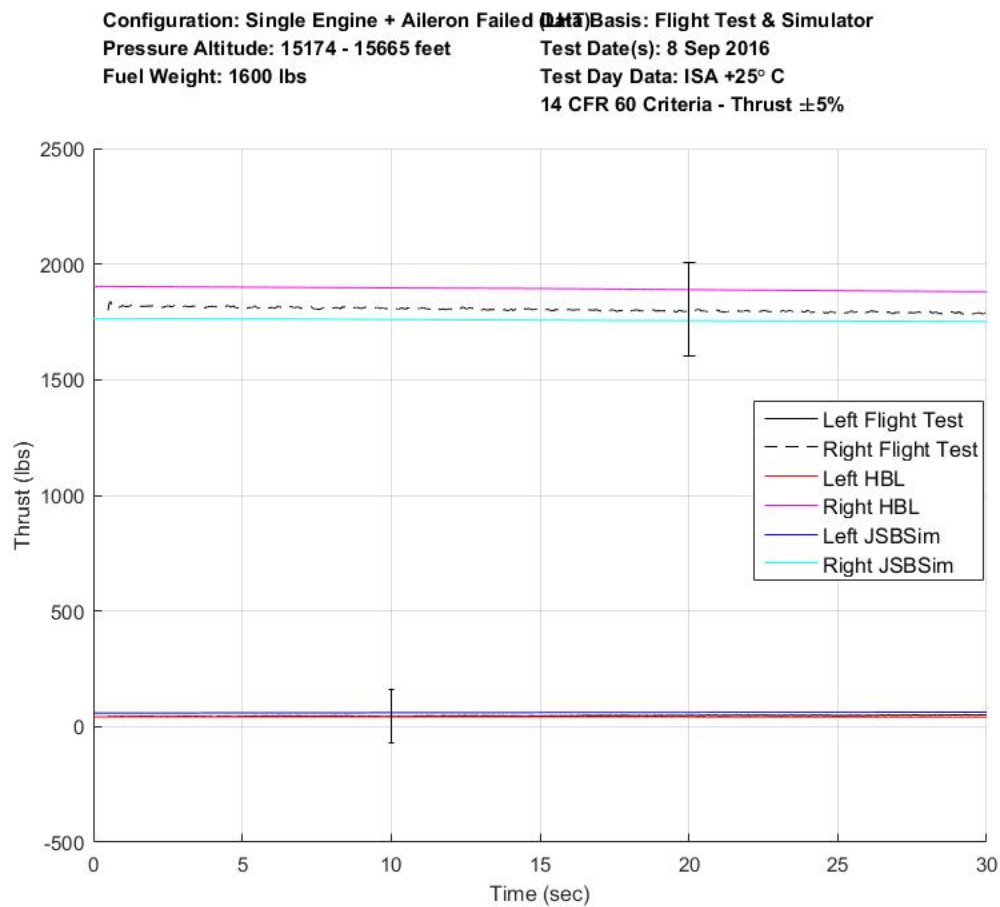


Figure F.37. Single Engine and Aileron Failed (LHT) at 1000 fpm: Split Thrust

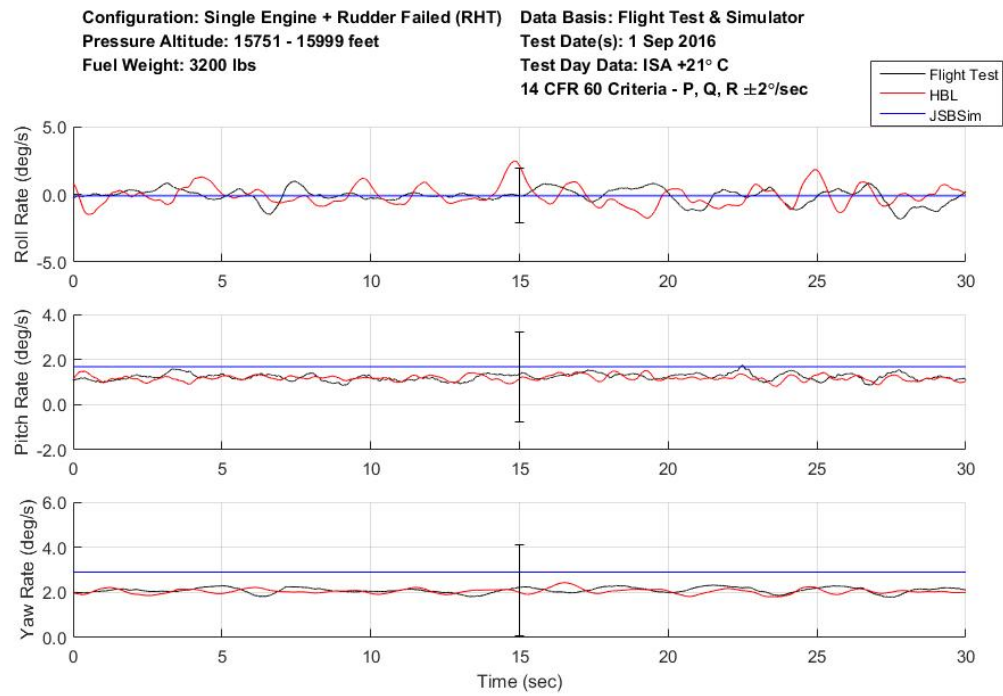


Figure F.38. Single Engine and Rudder Failed (RHT) at 500 fpm: Rates

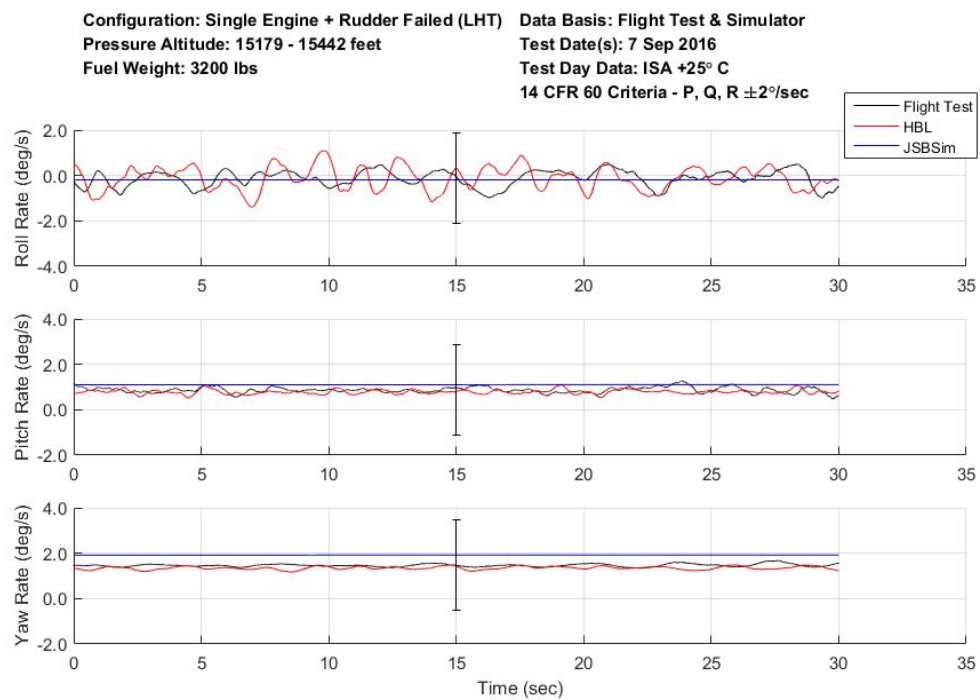


Figure F.39. Single Engine and Rudder Failed (LHT) at 500 fpm: Rates

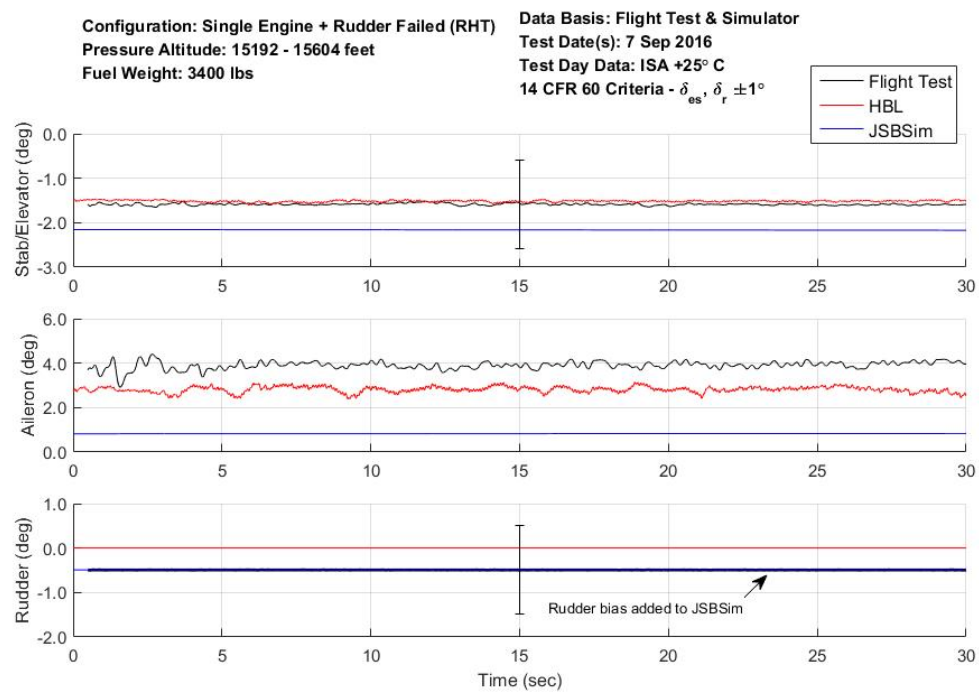


Figure F.40. Single Engine and Rudder Failed (RHT) at 1000 fpm: Control Deflections

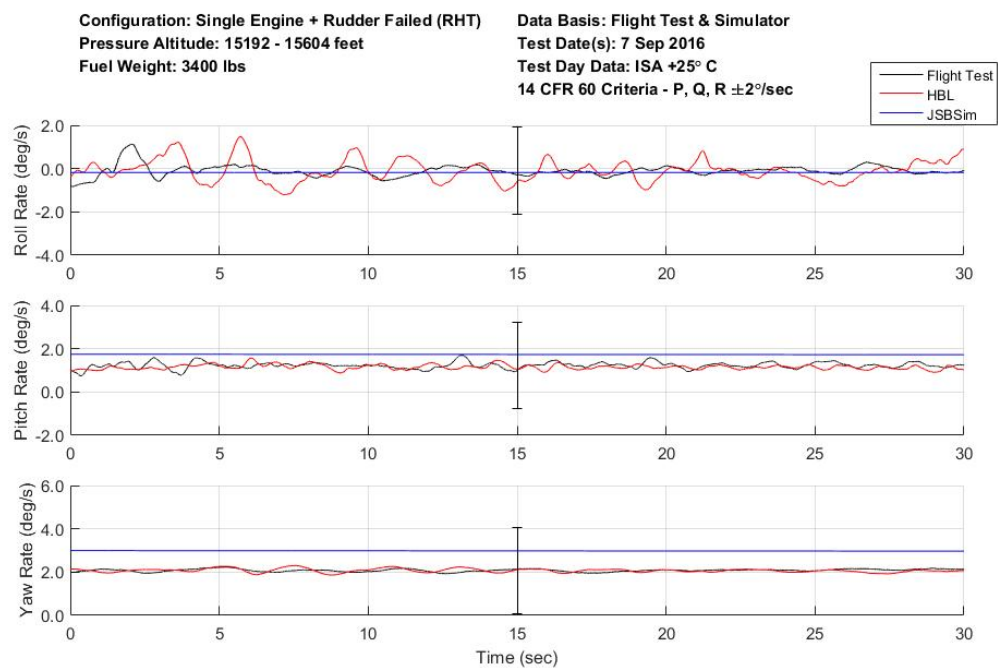


Figure F.41. Single Engine and Rudder Failed (RHT) at 1000 fpm: Rates

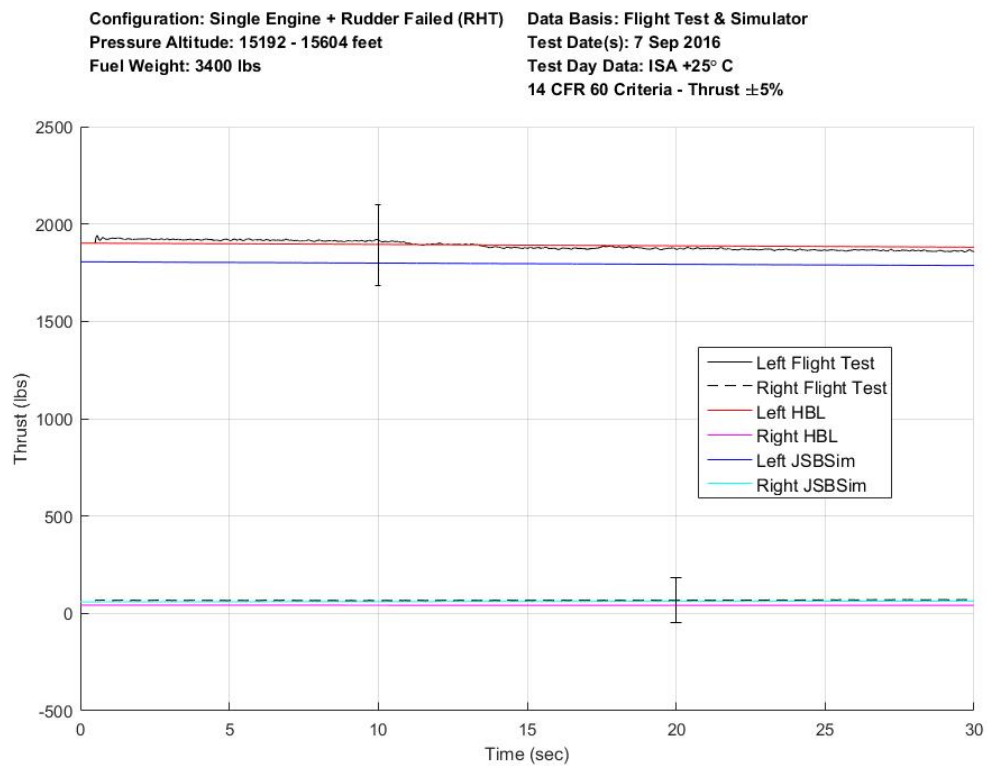


Figure F.42. Single Engine and Rudder Failed (RHT) at 1000 fpm: Split Thrust

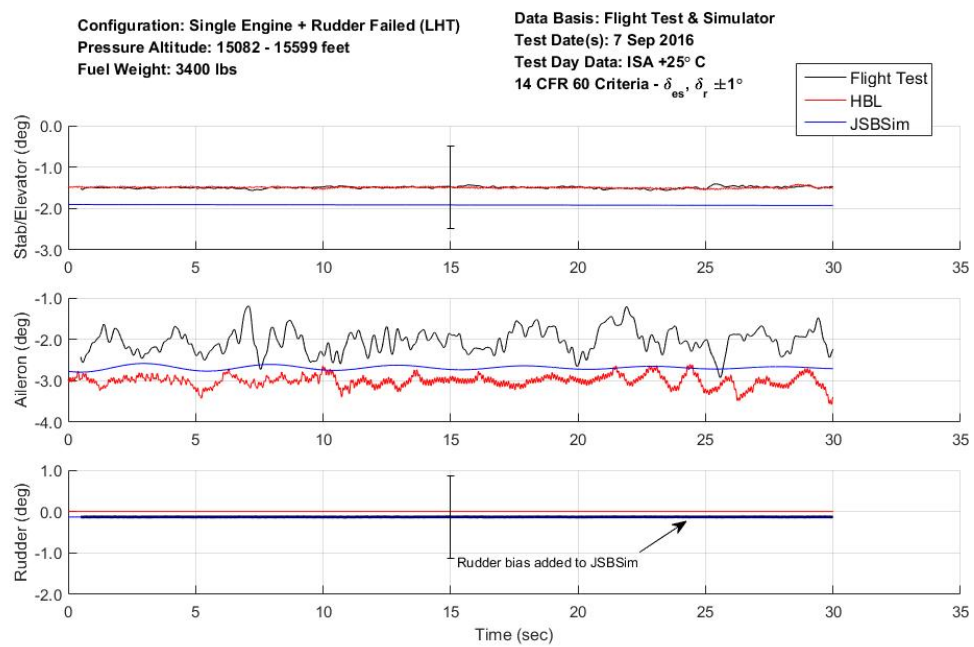


Figure F.43. Single Engine and Rudder Failed (LHT) at 1000 fpm: Control Deflections

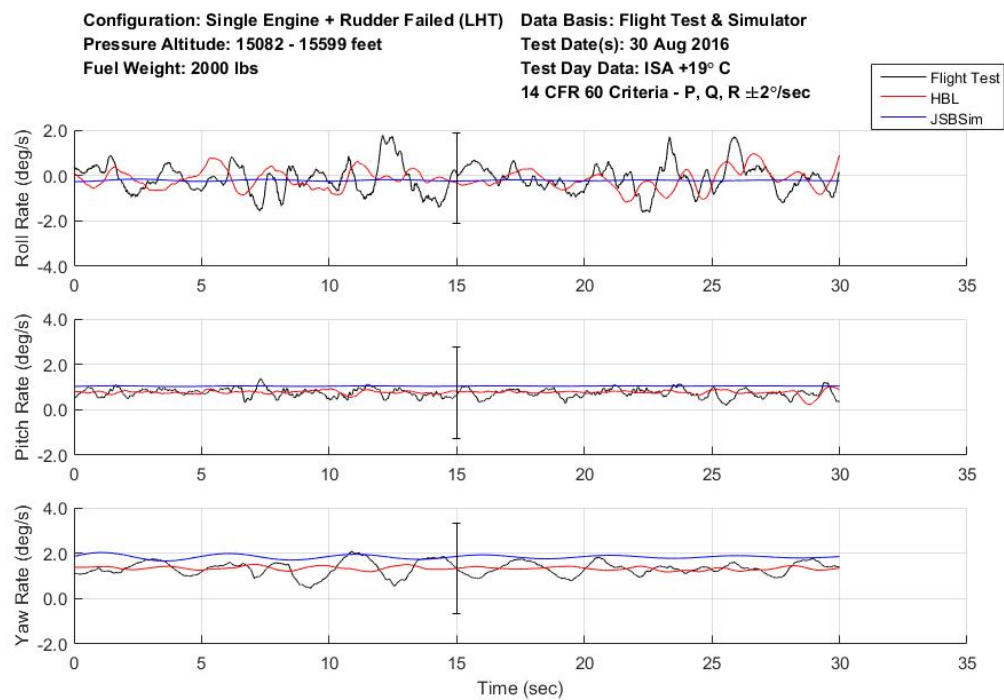


Figure F.44. Single Engine and Rudder Failed (LHT) at 1000 fpm: Rates

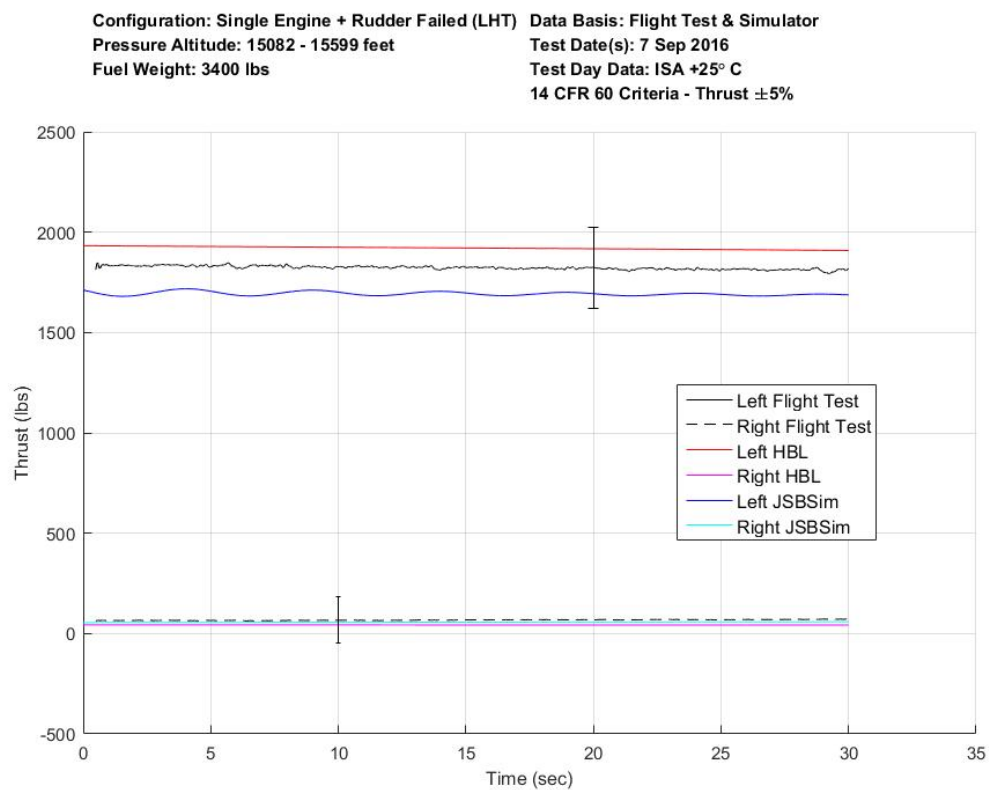


Figure F.45. Single Engine and Rudder Failed (LHT) at 1000 fpm: Split Thrust

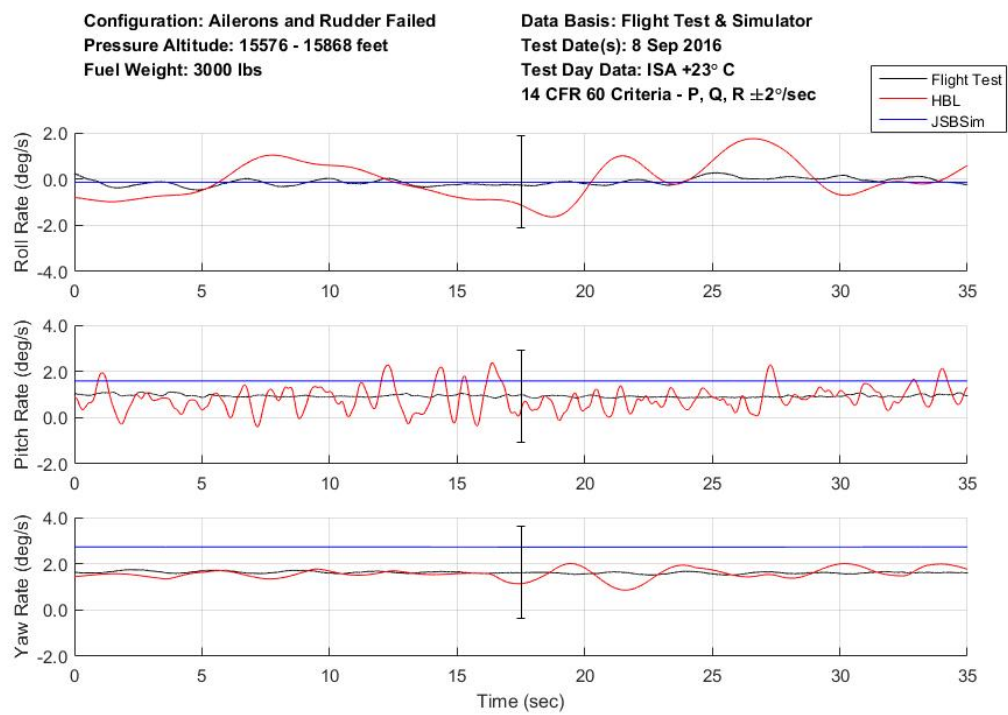


Figure F.46. Ailerons and Rudder Failed (500 fpm): Rates

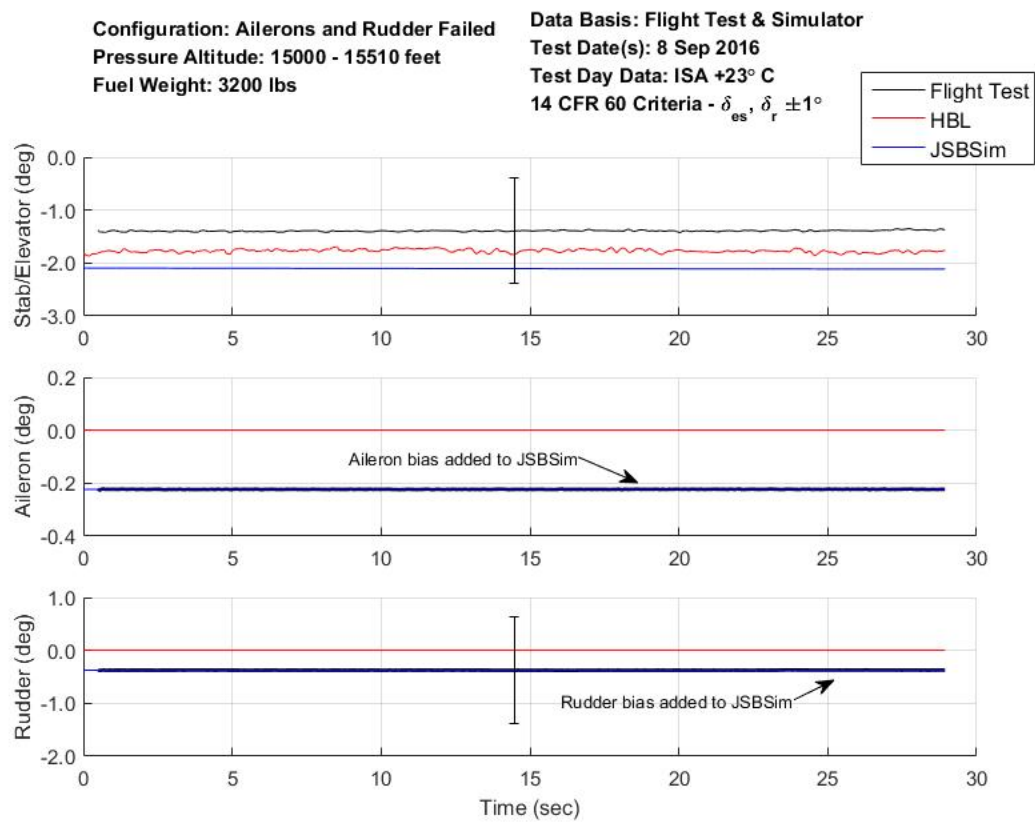


Figure F.47. Ailerons and Rudder Failed (1000 fpm): Control Deflections

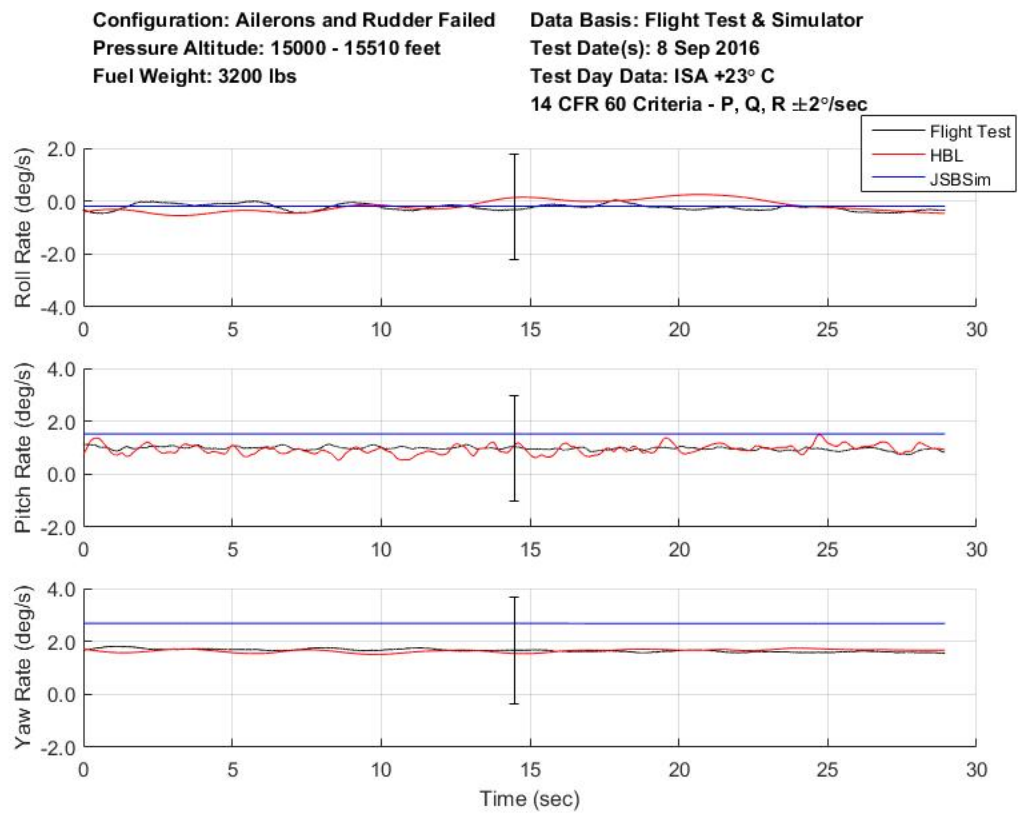


Figure F.48. Ailerons and Rudder Failed (1000 fpm): Rates

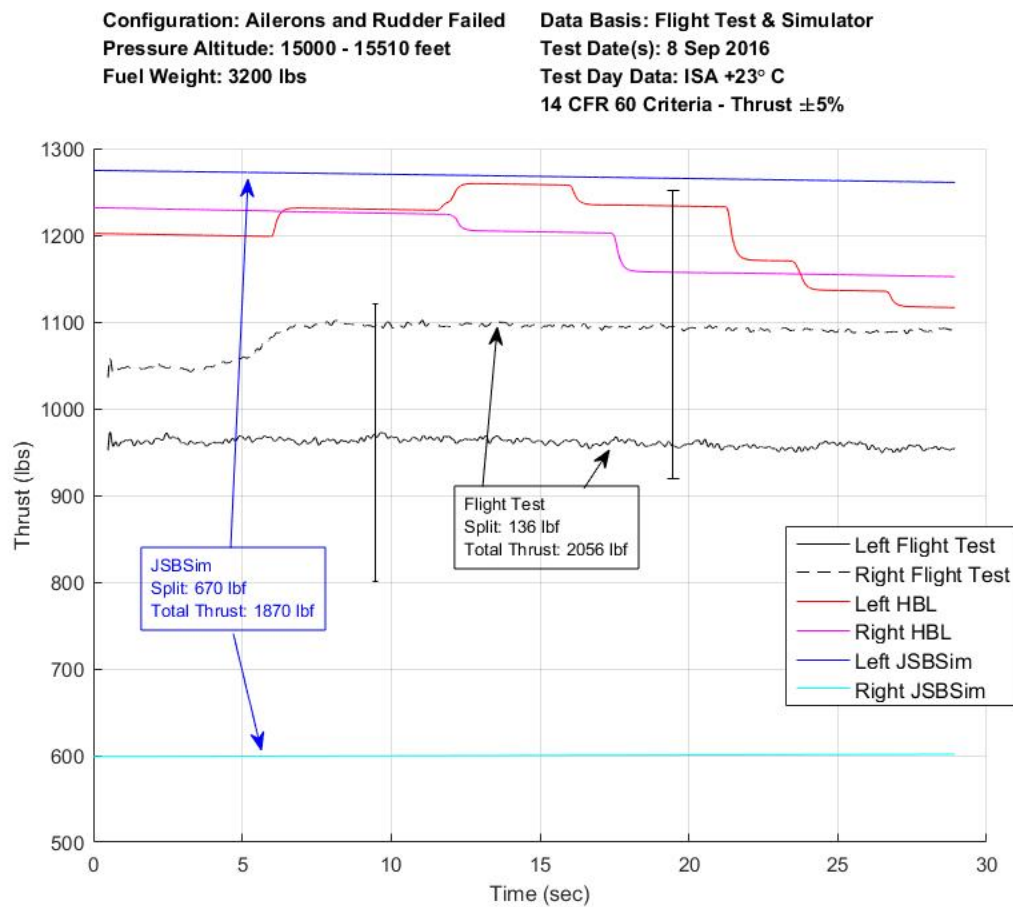


Figure F.49. Ailerons and Rudder Failed (1000 fpm): Split Thrust

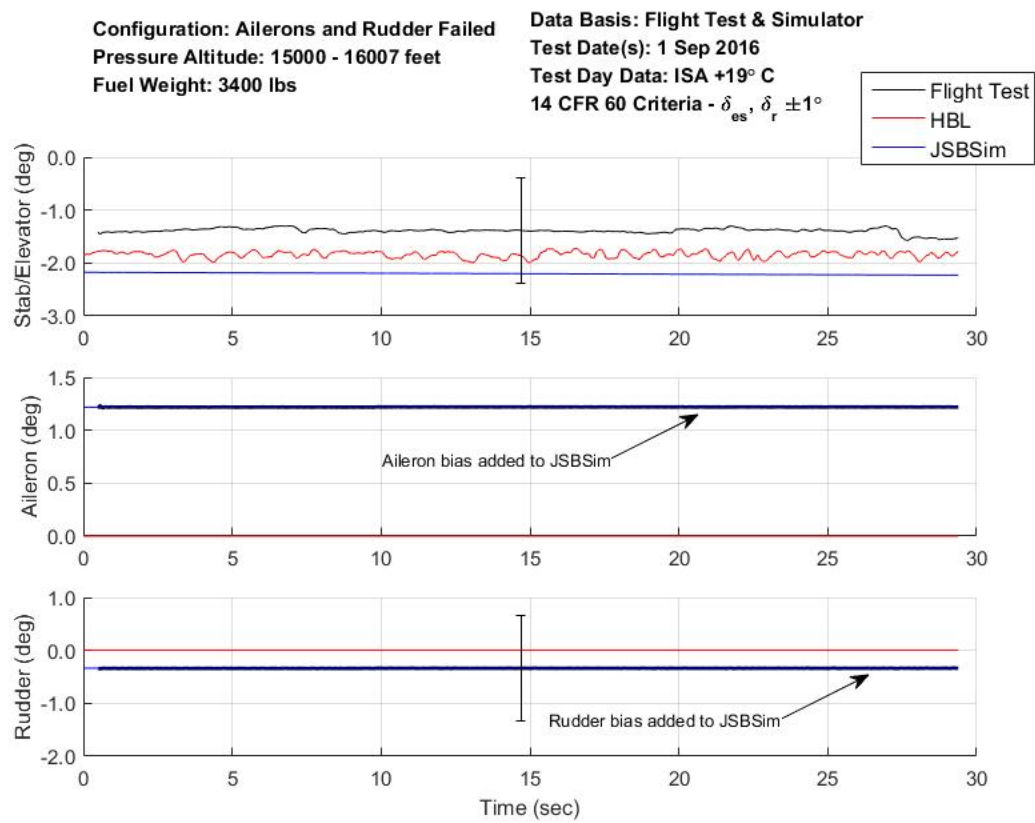


Figure F.50. Ailerons and Rudder Failed (2000 fpm): Control Deflections

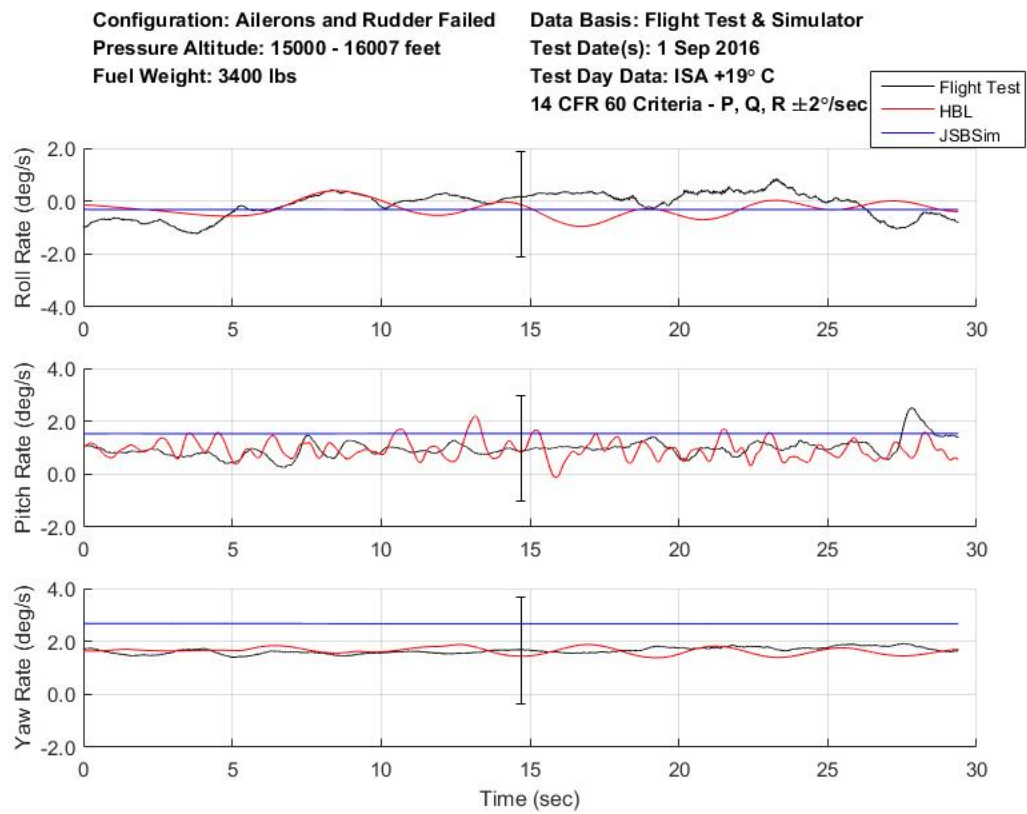


Figure F.51. Ailerons and Rudder Failed (2000 fpm): Rates

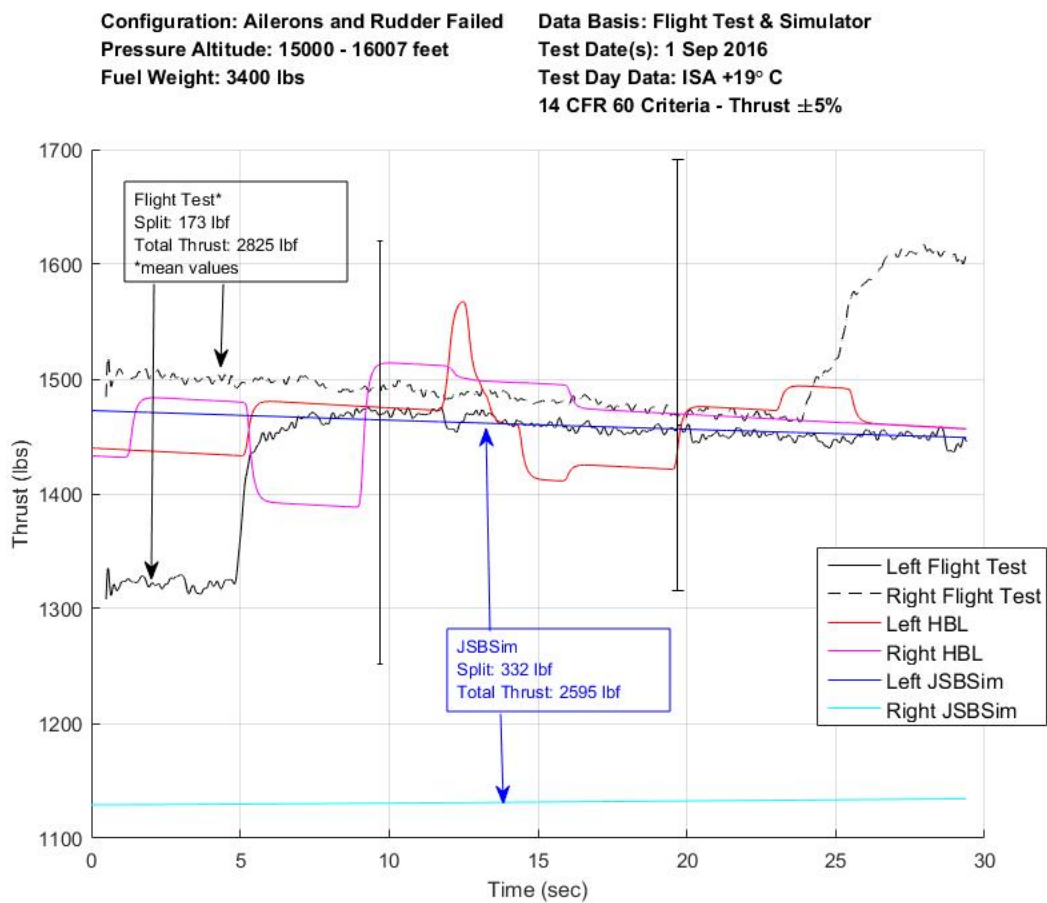


Figure F.52. Ailerons and Rudder Failed (2000 fpm): Split Thrust

Bibliography

1. Aerocross Systems, Inc. “Echo Hawk Affordable Multi-Mission UAV”. URL <http://www.aerocross.com/pdfs/ProductDatasheets/Aerocross-EchoHawk-Datasheet.pdf>. Accessed July, 2015.
2. Agte, J. *Multistate Analysis and Design: Case Studies in Aerospace Design and Long Endurance Systems*. Dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, Sept 2011.
3. Agte, J., N. Borer, and O. de Weck. “Multistate Design Approach to Analysis of Twin-Engine Aircraft Performance Robustness”. *Journal of Aircraft*, 49(3):781–793, 2012.
4. Agte, Jeremy, Nicholas Borer, and Olivia de Weck. “Design of Long-Endurance Systems with Inherent Robustness to Partial Failures During Operations”. *Journal of Mechanical Design*, 134, October 2012.
5. ArenaLogic. “ArenaLogic - Aircraft Simulation Software”. URL <http://www.arenalogic.com/>. Accessed July, 2015.
6. Atherton, K. “Google’s High-Flying Internet Drone Crashed In New Mexico Weeks Ago”. May. URL <http://www.popsci.com/googles-high-flying-internet-drone-crashes-outside-albuquerque>.
7. Barlow, R. and A. Wu. “Coherent Systems with Multistate Elements”. *Math. Operation Research*, 3:275.
8. Berndt, J. S. “JSBSim, An Open Source, Platform-Independent, Flight Dynamics Model in C++”. September 2011. URL <http://jsbsim.sourceforge.net/JSBSimReferenceManual.pdf>.
9. Berndt, Jon S. “JSBSim: An open source flight dynamics model in C++.” *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2004. URL <http://www.jsbsim.org/JSBSimReferenceManual.pdf>.
10. Berndt, Jon S. and Agostino De Marco. “Progress on the Usage of Open Source Flight Dynamics Model Software Library, JSBSim”. *AIAA Modeling and Simulation Technologies Conference*, 2009.
11. Betzina, M. D. and D. H. Brown. “Aerodynamic characteristics of an A-4B aircraft with simulated and actual gunfire damage to one wing”. 119, July 1976.
12. CAE-SimuFlite. “Learjet 24/25 Cockpit Reference Handbook”. August. URL <http://www.mrmoo.net/pilot/LR25/LR2520CockpitReferenceHandbook.pdf>.

13. Calspan-Corporation. "Learjet Inflight Simulator". 2015. URL <http://www.calspan.com/aerospace/flight-test-aircraft-operations/airborne-testbed/in-flight-simulators>.
14. Colantonio, R. "LEWICE". URL <http://icebox-esn.grc.nasa.gov/design/lewice.html>. Accessed August, 2015.
15. Cummings, R. M., J. R. Forsythe, S. A. Morton, and K. D. Squires. "Computational Challenges in High Angle of Attack Flow Prediction". *Progressive Aerospace Science*, 39(5):369, 2003.
16. DoD. "VV&A Recommended Practices Guide". November 1996. URL [https://acc.dau.mil/adl/en-US/649735/file/71953/Verification_validation_Accreditation\(VV_A\).pdf](https://acc.dau.mil/adl/en-US/649735/file/71953/Verification_validation_Accreditation(VV_A).pdf). Accessed, August 2015.
17. DoD. *MIL-STD-1797B Flying Qualities of Piloted Aircraft*. Department of Defense, 2004.
18. DoD. "DoDD 5000.59 - DoD Modeling and Simulation (MS) Management". August 2007. URL <http://www.dtic.mil/whs/directives/corres/pdf/500059p.pdf>. Accessed, August 2015.
19. Dorobantu, Andrei, Peter Seiler, and Gary Balas. "Validating Uncertain Aircraft Simulation Models Using Flight Test Data". *AIAA Atmospheric Flight Mechanics Conference*, August 2013.
20. Draper, N. and H. Smith. *Applied Regression Analysis*. Wiley, New York, NY, 2nd edition, 1981.
21. El-Neveih, E., F. Proschan, and J. Sethuraman. "Multi-state Coherent System". *Journal of Applied Probability*, 15:675.
22. Erb, R. *Pitot Statics and the Standard Atmosphere (2nd Ed.)*. USAF Test Pilot School, 2015.
23. Erm, L. P. "Survey on the Aerodynamics of Battle-Damaged Combat Aircraft". March 1999. URL <http://www.dtic.mil/dtic/tr/fulltext/u2/a360572.pdf>.
24. FAA. "AC 120-40B - Airplane Simulator Qualification". July 1991. URL https://www.faa.gov/regulations_policies/advisory_circulars/index.cfm/go/document.information/documentID/22762. Accessed, August 2015.
25. FAA. "14 CFR 60 - Flight Simulation Training Device Initial and Continuing Qualification and Use". January 2012. URL <http://www.gpo.gov/fdsys/>

- granule/CFR-2012-title14-vol2/CFR-2012-title14-vol2-part60.
Accessed, November 2015.
26. FAA. “Model: TCDS No. A10CE, Rev. No. 67, Learjet Inc., 25F”. February 2015. URL http://rgl.faa.gov/Regulatory_and_Guidance_Library/rgMakeModel.nsf/MainFrame?OpenFrameSet. Accessed, September 2015.
 27. Fiorino, F. “Southwest Adds Winglets to 737-700s to Boost Performance”. June. URL <http://aviationweek.com/awin/southwest-adds-winglets-737-700s-boost-performance>.
 28. Flight-Gear. “FlightGear Introduction”. URL <http://www.flightgear.org/about/>. Accessed July, 2015.
 29. Frost, W. “Flight in Low-Level Wind Shear”. *NASA Contractor Report 3678*, March 1983. URL <http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19830013440.pdf>. Accessed August, 2015.
 30. Goett, H. “SP-3300 Flight Research and Ames, 1940-1997”. 1998. URL Retrieved from: <https://history.nasa.gov/SP-3300/contents.htm>.
 31. Goppert, James, I. Hwang, and Agostino De Marco. “On Some Trim Strategies for Nonlinear Aircraft Flight Dynamics Models with the Open Source Software JSBSim”. *Proceedings of 3rd CEAS Air and Space Conference*, October 2011.
 32. Hayes, C. “Effects of simulated wing damage on the aerodynamic characteristics of a swept-wing model”. 1968.
 33. Hines, D. O. “Simulation in Support of Flight Testing”. *NATO Research and Technology Organization*, December 2000.
 34. duPont Inc. “The DP-1 UAV”. URL <http://www.dupontaero.com/DP-1/dp-1.html>. Accessed July, 2015.
 35. Inc., MathWorks. “What is an S-function?” 2016. URL <https://www.mathworks.com/help/simulink/sfg/what-is-an-s-function.html>. Accessed, January 2017.
 36. Instruments, National. “PID Theory Explained”. March 2011. URL <http://www.ni.com/white-paper/3782/en/>. Accessed, November 2015.
 37. Jategaonkar, R. and D. Fischenberg. “Aerodynamic Modeling and System Identification from Flight Data Recent Applications at DLR”. *Journal of Aircraft*, 21(4):681–694, 2004.
 38. JSBSim. “JSBSim S-Function Project”. URL <http://jsbsim.sourceforge.net/matlab.html>. Accessed July, 2015.

39. Kemper, Jay C. "Learjet Maneuvers". private email communication, November 2014.
40. Kirkpatrick, Jason. "Learjet Maneuvers". private email communication, November 2015.
41. Klein, V. and E. A. Morelli. *Aircraft System Identification: Theory and Practice*. AIAA Education Series, Reston, Virginia, 2006. ISBN 1-56347-832-3.
42. Lisnianski, A. and G. Levitin. *Multi-state System Reliability*. ISBN 9789812383068.
43. MathWorks, Inc. "System Identification Toolbox". 2015. URL <http://www.mathworks.com/products/sysid/>. Accessed August, 2015.
44. McFarland, R. E. "Finite Element Aircraft Simulation of Turbulence". *NASA Technical Memorandum 110437*, February 1997. URL <http://www.aviationsystemsdivision.arc.nasa.gov/publications/hit1/rtsim/Feast.pdf>. Accessed August, 2015.
45. McMahon, Ryan. "Learjet Maneuvers". private email communication, November 2015.
46. Milliken, W. F. Jr. "Progress in Stability and Control Research". *Journal of Aeronautical Sciences*, 14:494.
47. Moore, E. F. and C. E. Shannon. "Reliable Circuits Using Less Reliable Relays". URL https://en.wikipedia.org/wiki/Reliability_theory. Accessed July, 2015.
48. Morelli, Eugene. *System Identification Programs for Aircraft (SIDPAC)*. AIAA Atmospheric Flight Mechanics Conference, Monterey, CA, August 2002. URL <http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20030006686.pdf>. Accessed August, 2015.
49. Murchland, J. "Fundamental Concepts and Relations for Reliability Analysis of Multistate Systems. Reliability and Fault Tree Analysis". *Theoretical and Applied Aspects of System Reliability*.
50. Myers, R. *Classical and Modern Regression and Applications*. Duxbury Press, Boston, MA, 1986.
51. Nelder, J. A. and R. Mead. "Simplex Method for Function Minimization". *Journal of Computing*, 1965.
52. Newton, C. "Facebook Takes Flight". November. URL <http://www.theverge.com/a/mark-zuckerberg-future-of-facebook/aquila-drone-internet>.

53. Norris, G. "USAF Test Pilot School Vets Safer Adaptive Flight Controller". May. URL <http://aviationweek.com/technology/usaf-test-pilots-school-vets-safer-adaptive-flight-controller>.
54. Northrup-Grumman. "RQ-4 Block 40 Global Hawk". 2011. URL http://www.northropgrumman.com/Capabilities/GlobalHawkLogisticsServices/Documents/pageDocs/globalhawk_block40.pdf.
55. Ogata, Katsuhiko. *Modern Control Engineering*. Prentice Hall, 5th edition, 2009. ISBN 9780136156734.
56. OpenCourseWare, MIT. "Natural Frequency and Damping". 2010. URL https://ocw.mit.edu/courses/mathematics/18-03-differential-equations-spring-2010/readings/supp_notes/MIT18_03S10_chapter_13.pdf. Accessed, January 2017.
57. OpenEagles. "OpenEagles Simulation Framework". URL <http://www.openeaagles.org/wiki/doku.php>. Accessed July, 2015.
58. Park, M., L. Green, R. Montgomery, and D. Raney. "Determination of Stability and Control Derivatives Using Computational Fluid Dynamics and Automatic Differentiation". *AIAA Paper 99-3136*, 1999.
59. Pomerleau, M. "Future of Unmanned Capabilities: MALE vs HALE". May. URL <http://defensesystems.com/articles/2015/05/27/uas-male-vs-hale-debate.aspx>.
60. Powers, M. "Cray Supercomputers Play Key Role in Designing Boeing Dreamliner". URL <http://investors.cray.com/phoenix.zhtml?c=98390&p=irol-newsArticle&ID=1022899>. Accessed July, 2015.
61. Rausand, Marvin and Arnljot Hoyland. "System Reliability Theory: Models, Statistical Methods, and Applications".
62. Rogers, S. "Drones By Country: Who Has All The UAVs?" August. URL <http://www.theguardian.com/news/datablog/2012/aug/03/drone-stocks-by-country>.
63. Roskam, Jay. *Airplane Flight Dynamics and Automatic Control, Vol. 1*. Wiley-VCH, Lawrence, Kansas, 1994. ISBN 1884885179.
64. Ross, S. "Mutivalued State Element Systems". *Annals of Probability*, 7:379.
65. Schumacher, L. E. "A Method for Evaluating Aircraft Stability Parameters from Flight Test Data". 1952. URL <http://www.dtic.mil/dtic/tr/fulltext/u2/a075866.pdf>. Accessed August, 2015.

66. Science, Aurora Flight. “Medium Altitude Long-Endurance Unmanned Aircraft”. 2015. URL <http://www.aurora.aero/orion/>.
67. Seltzer, R. M. and G. R. Rhodeside. “Fundamentals and Methods of High Angle-of-Attack Flying Qualities Research”. January 1988.
68. Spearman, M. L. “Wind-tunnel studies of the effects of simulated damage on the aerodynamic characteristics of airplanes and missiles”. December 1982.
69. Steinberg, M. “History of Reconfigurable Flight Control”. *Journal of Aerospace Engineering*, 219:263.
70. Stevens, Brian L. and Frank L. Lewis. *Aircraft Control and Simulation*. John Wiley Sons, Inc, Hoboken, New Jersey, 2nd edition, 2003. ISBN 9780471371458.
71. Theil, H. *Economic Forecasts and Policy*. North Holland Publishing Company, 1958.
72. Tischler, M. and R. Remple. *Aircraft and Rotorcraft System Identification: Engineering Methods with Flight Test Examples*. AIAA Education Series, 2nd edition, 2012. ISBN 978-1600868207.
73. VirtualAir. “Virtual Air”. URL <http://virtualair.sourceforge.net/>. Accessed July, 2015.
74. Weingarten, N. C. “History of In-Flight Simulation and Flying Qualities Research at Calspan”. *Journal of Aircraft*, 42(2), 2005.
75. Yechout, T. *Introduction to Aircraft Flight Mechanics*. AIAA Education Series, 2014.

Vita

Major Mark W. Vahle was born on April 26th, 1983, in Saint Louis, Missouri. He was graduated from Purdue University in 2005 with a Bachelors of Science in Aeronautical and Astronautical Engineering. He commissioned into the United States Air Force through the Reserve Officers' Training Corps (ROTC) and upon commissioning, attended Specialized Undergraduate Pilot Training (SUPT), Columbus AFB, Mississippi, where he was recognized as a distinguished graduate and was selected to fly the F-16. He served his first assignment at Shaw AFB, South Carolina, as electronic combat pilot (ECP) for the 77th Fighter Squadron. While at Shaw AFB, Major Vahle deployed twice, once to Balad, Iraq, in support of Operation Iraqi Freedom (OIF), and second to Aviano AB, Italy, in support of Operation Unified Protector (OUP). He served his second assignment as an F-16 instructor pilot and flight commander at the F-16 Fighter Training Unit (FTU), 308th Fighter Squadron, Luke AFB, Arizona. Major Vahle's third operational tour was at Kunsan AB, South Korea, where he served as chief of standardization and evaluation for the 8th Operations Group. In 2014, Major Vahle was selected for the joint AFIT/USAF Test Pilot School program. In August of 2014, he began his studies towards a Master of Science in Aerospace Engineering at the Air Force Institute of Technology (AFIT). Upon completion of his studies he was assigned to USAF Test Pilot School, Class 16A, at Edwards AFB, CA. At Test Pilot School Maj Vahle was recognized as a distinguished graduate and assigned to the 40th Flight Test Squadron at Eglin AFB, Florida.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE (DD-MM-YYYY) 02-01-2017			2. REPORT TYPE Master's Thesis		3. DATES COVERED (From — To) Sept 2014 — Mar 2017	
4. TITLE AND SUBTITLE Application and Validation of Off-Nominal Aircraft Performance Models					5a. CONTRACT NUMBER	
					5b. GRANT NUMBER	
					5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Vahle, Mark W., Maj, USAF					5d. PROJECT NUMBER	
					5e. TASK NUMBER	
					5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765					8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GAE/ENY/17-M-297	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department of Aeronautics and Astronautics 2950 Hobson Way WPAFB OH 45433-7765 DSN 271-0690, COMM 937-255-3636 Email: mark.vahle@us.af.mil					10. SPONSOR/MONITOR'S ACRONYM(S)	
					11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION / AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.						
13. SUPPLEMENTARY NOTES						
14. ABSTRACT When faced with on-board system failures (i.e. control surface failures), recent developments in multistate design have shown great promise in improving system reliability through early stage design consideration of off-nominal performance. Accurately modeling the response to a control input typically requires a high-fidelity aerodynamic model and simulation routine with the ability to predict off-nominal aircraft performance. Presented in this research is a methodology to verify and validate off-nominal aircraft performance utilizing a low-cost open-source flight dynamics simulator (JSBSim). The central objective being a determination of the degree of accuracy in which one can model off-nominal performance in order to make early stage design decisions required by the multistate design process. Results indicate that nominal and off-nominal P&FQ could be predicted inside FAA simulator qualification criteria at low-cost and with a reasonable amount of effort.						
15. SUBJECT TERMS Off-nominal states, multistate aircraft design, simulation, modeling, validation, degraded states, aircraft reliability, robust aircraft design, JSBSim						
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON	
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. R. Cobb, AFIT/ENY	
U	U	U	U	259	19b. TELEPHONE NUMBER (include area code) (937) 255-3636, x4559; richard.cobb@afit.edu	