2009

Computational environment for the development of an FAA compliant level 6 flight training device

Steven Mullins
West Virginia University

Computational Environment for the Development of an FAA Compliant Level 6 Flight Training Device

Steven Mullins

Thesis submitted to the

College of Engineering and Mineral Resources

at West Virginia University

in partial fulfillment of the requirements

for the degree of

Master of Science

in Aerospace Engineering

Committee:

Dr. Larry Banta

Dr. Marcello Napolitano

Dr. Mario Perhinschi, Chair

Department of Mechanical and Aerospace Engineering

Morgantown, WV

2009

ABSTRACT

Computational Environment for the Development of an FAA Compliant Level 6 Flight Training Device

Steve Mullins

A flight simulator can successfully achieve its purpose only if equipped with adequate mathematical models of the aircraft, its sub-systems, and the environment. The US Federal Aviation Administration (FAA) has instituted stringent regulations to ensure that flight simulators used for pilot training reach desirable levels of accuracy and fidelity. The purpose of this thesis is to present the development and application of a design strategy and the computational environment associated to it for building an aircraft simulation model that meets the FAA regulations for flight simulator performance. The proposed methodology is based on using flight test data in combination with analytical modeling tools and heuristics.

The Simulink simulation environment within Matlab was selected due to its recognized capabilities, flexibility, and portability. Several interactive computational tools have been developed to support the development. Flight test data of a business class jet was used for the purpose of this research effort. An important part of the proposed strategy consists of selecting the flight data and converting them into a usable format for Matlab/Simulink. Parameter identification techniques must then be applied at specific points in the flight envelope of the aircraft in order to create an accurate flight dynamics model. Two such modeling techniques, in time and frequency domain, were used within this project. Lookup tables for the stability and control derivatives were built based on dynamic pressure. Tuning of the aerodynamic model is required to meet all FAA criteria. Once the FAA objective tests were completed, another more organic set of tests were conducted by pilots. The outcomes of these subjective tests were analyzed and additional tuning of the aerodynamic and dynamic model were performed accordingly. Eventually, compliance with both FAA objective and subjective tests is ensured through several tuning iterations and demonstrated.

Acknowledgments

I would like to thank Dr. Mario Perhinschi, Dr. Marcello Napolitano, and Phil Evans for their support on this project.

Table of Contents

List of Figures

List of Tables

Nomenclature

Acronyms

FAA	Federal Aviation Administration
PID	Parameter Identification

Stability and Control Derivatives

C_l	Rolling Moment Coefficient
C_{l0}	Initial Rolling Moment Coefficient
$C_{l\beta}$	Variation of Airplane Rolling Moment Coefficient with Angle of Sideslip
$C_{l\delta a}$	Variation of Airplane Rolling Moment Coefficient with Aileron Deflection Angle
$C_{l\delta r}$	Variation of Airplane Rolling Moment Coefficient with Rudder Deflection Angle
C_{lp}	Variation of Airplane Rolling Moment Coefficient with Change of Roll Rate
C_{lr}	Variation of Airplane Rolling Moment Coefficient with Change of Yaw Rate
C_m	Pitching Moment Coefficient
C_{m0}	Initial Pitching Moment Coefficient
$C_{m\alpha}$	Variation of Airplane Pitching Moment Coefficient with Change of Angle of Attack
C_{mq}	Variation of Airplane Pitching Moment Coefficient with Change of Rate of Pitch
C_n	Yawing Moment Coefficient
C_{n0}	Initial Yawing Moment Coefficient
$C_{n\beta}$	Variation of Airplane Yawing Moment Coefficient with Change of Sideslip Angle
$C_{n\delta a}$	Variation of Airplane Yawing Moment Coefficient with Change of Aileron Deflection
$C_{n\delta r}$	Variation of Airplane Yawing Moment Coefficient with Change of Rudder Deflection
C_{np}	Variation of Airplane Yawing Moment Coefficient with Change of Roll Rate
C_{nr}	Variation of Airplane Yawing Moment Coefficient with Change of Yaw Rate
C_x	Force Coefficient along the X-axis
C_{x0}	Initial Force Coefficient in the X-axis
$C_{x\alpha}$	Variation of airplane X-axis Force Coefficient with Angle of Attack
$C_{x\delta e}$	Variation of airplane X-axis Force Coefficient with Change of Elevator Deflection

C_{xq}	Variation of airplane X-axis Force Coefficient with Change of Pitch Rate
C_y	Force Coefficient along the Y-axis
$C_{y\delta r}$	Variation of airplane Y-axis Force Coefficient with Change of Rudder Deflection
C_{y0}	Initial Force Coefficient in the Y-axis
$C_{y\beta}$	Variation of airplane Y-axis Force Coefficient with Change of Sideslip Angle
$C_{y\delta a}$	Variation of airplane Y-axis Force Coefficient with Change of Aileron Deflection
C_{yp}	Variation of airplane Y-axis Force Coefficient with Change of Roll Rate
C_{yr}	Variation of airplane Y-axis Force Coefficient with Change of Yaw Rate
C_z	Force Coefficient along the Z-axis
C_{z0}	Initial Force Coefficient in the Z-axis
$C_{z\alpha}$	Variation of airplane Z-axis Force Coefficient with Change of Angle of Attack
$C_{z\delta e}$	Variation of airplane Z-axis Force Coefficient with Change of Elevator Deflection
C_{zq}	Variation of airplane Z-axis Force Coefficient with Change of Pitch Rate

Variables

α	Angle of Attack
b	Wing Span
β	Sideslip Angle
c_bar	Mean Aerodynamic Chord
C1	Left Bound
C2	Right Bound
δa	Aileron Deflection
Δ	Change in
δe	Elevator Deflection
δr	Rudder Deflection
H	Known Matrix
I	Moments of Inertia
J(θ)	Cost Function
ω	Frequency
p	Roll Rate
q	Pitch Rate
$\bar{q}$	Dynamic Pressure
r	Yaw Rate
R^{-1}	Positive Definite Weighting Matrix

R1	Upper Bound
R2	Lower Bound
t	Time
θ	Unknown Matrix
θ_hat	Parameter Vector
v	Noise Measurement
V	Velocity
X	Matrix of Vectors and Regressors
x_hat	Fourier Transform for Frequency Domain
z	Measurement
†	Conjugate Transpose

1. Introduction

Flight simulation is the inexpensive and effective way for pilots to gain proficiency in specific aircraft. With this type of training, the pilot can be more focused on the specific task at hand without exposure to inherent risks of actual flight. Flight simulation has always been an important part of flight training in the brief history of aviation, which shows that the flight simulation process has been in the minds of many flight instructors since near the beginning of mechanical flight. Instructors needed a tool that was safe, yet effective at teaching beginner pilots the intricacies of flying. The beginning of flight simulation can be identified with the Penguin System in the paper by Page [1]. This was then followed by the Sanders Teacher, Link Trainer, analog computers, and then the first digital simulator, the Link Mark 1. After this point, the digital flight simulation process was expanded on in many ways with advances in technology. Today, sophisticated flight simulators are used on a large scale for pilot training, but not only. Flight simulators have become an indispensable tool for the design of new aircraft contributing substantially to reducing duration and cost of development.

A flight simulator can successfully achieve its purpose only if equipped with adequate mathematical models of the aircraft, its sub-systems, and the environment. Although the necessary mathematics, physics knowledge and tools had been available for more than a century, how exactly to apply them to the problem of flight is a relatively recent discovery. Flight dynamics and aerodynamics, as a framework for flight modeling and simulation have achieved fast progress shortly before and during World War II. As a result, the first classic textbooks were published by Perkins (1949) [2], followed by Etkin (1959) [3], and Miele (1962) [4]. Other early contributions that are still very useful, include the books by Seckel (1964) [5], McCormick (1979) [6], and Babister (1980) [7]. Currently, numerous textbooks are available offering new

perspectives and methodologies for flight simulation and development of aircraft sub-system models [8 – 13]. Modeling of aircraft has benefited tremendously from the development of advanced methodologies and algorithms that allow the determination of the mathematical model of a dynamic system from series of experimental tests that explicitly relate known inputs to measured resulting outputs. These parameter identification (PID) techniques rely on a pre-defined structure of the model, which depends on a set of constant parameters that are unknown and must be determined. The experimental data points allow a system of equations to be built whose solution is the set of unknown parameters. Two large classes of methods have been developed, usually referred to as "time domain methods" [14-15] and "frequency domain methods" [14-16].

Worldwide, the engineering communities and governmental authorities have acknowledged the need for systematic and scientifically formulated requirements and criteria that could ensure the objective assessment of flight simulators performance level and accuracy. The US Federal Aviation Administration (FAA) has instituted, as early as 1973, regulations to this respect from the United States Advisory Circular 120-40B [17]. In the United States FAR regulations Part 60, a section was devoted to giving each type of flight training device a classification. In order for a simulator to be in a classification it must meet certain requirements. There a total of 7 different levels of flight training devices. Level 1 is the lowest flight training devices recognized by the FAA. Level 2 and 3 flight training devices are used for simulating a range of aircraft and have a generic cockpit setup. Level 4 through 7 flight training devices are made for a specific aircraft and have a replica of the cockpit for the aircraft. As the level of the flight training device increases the requirements set by the FAA become more stringent due to the complex nature of the flight training devices at the higher levels [18].

Compliance with such regulations requires that the development of the mathematical model and, in general, the design of the flight simulator be conducted following an ad-hoc strategy. The purpose of this thesis is to present the development and application of a design process and the computational environment associated to it for building an aircraft simulation model that meets the FAA regulations for flight simulator performance. The proposed methodology is based on using flight test data in combination with analytical modeling tools and heuristics to create a model that is accurate enough to pass the very stringent FAA requirements.

The Simulink simulation environment in Matlab was selected due to its recognized capabilities, flexibility, and portability. Matlab is a widely used software for simulation and the outcomes of this project may be easily accessible to wide categories of users throughout the aircraft modeling and simulation community.

Flight test data of a business class jet was used for the purpose of this research effort. An important part of the strategy of designing this aircraft model was to convert the flight test data into a usable format for Matlab/Simulink. Two Parameter Identification techniques were applied at specific points in the flight envelope of the jet aircraft in order to create an accurate flight dynamics model. Lookup tables for the stability derivatives at these flight conditions were built dependent on dynamic pressure. The first PID technique is the least squares estimator approach in the time domain found in [14]. The other technique is the recursive Fourier transform approach in the frequency domain found in [14].

Once a preliminary model of the aircraft was determined and put into the simulation, tests were then run using the inputs of the flight data. The flight test data was then compared to the output of the aircraft model as part of the objective FAA requirements for a Level 6 training

device. Tuning of the aerodynamic model was performed to meet all FAA criteria. Once the FAA objective tests were completed, another more organic set of tests were conducted by pilots. These subjective tests were not checked with flight test data; instead, pilot subjective evaluation was used. All issues that came up from the subjective tests were analyzed and solved by changing the aerodynamic and dynamic model accordingly. Finally, another iteration was performed to ensure that the FAA objective tests are satisfied.

This thesis is organized as follows. After this brief introduction, in Chapter 2, some of the most important milestones in the development of flight simulators and performance evaluation criteria are outlined. The general structure of the computational environment for the development of an FAA compliant aircraft simulation model is presented in Chapter 3. The next three chapters are dedicated to each of the main components of the proposed design strategy. Chapter 4 presents the computational tools developed for processing of the flight test data. The aircraft parameter identification techniques used are discussed in Chapter 5. The simulation environment developed for FAA requirements compliance and its use for performing objective and subjective tests are presented in Chapter 6. Finally, some conclusions are drawn in Chapter 7 followed by a list of references. Several appendices are included with relevant additional information.

2. History of Flight Simulation and Certification Requirements

The start of flight simulation was not a true flight simulator per se. Most of the first pilots learned to fly by starting off with a machine that was not airworthy but acted like an aircraft while on the ground. After the pilot was comfortable with the handling on the ground they then graduated to an aircraft that would be able to do small "hops" that would teach the pilot how to take off and land safely. After the pilot was skilled with small hops, the pilot would then move on to large hops and then to an actual flight. A similar style of training was adopted just before World War I known as the Penguin system [1]. With the Penguin system, instead of providing the trainees with an aircraft incapable of flight, an aircraft would be taken and its wingspan reduced so it would remain ground borne. This technique was effective and was used through the First World War. Although this was not true ground borne simulation, these beginning ideas transformed flight simulation into what it is today.

After the initial "hopping", or Penguin system was used, a new approach was attempted for flight simulation. The technique essentially created an aircraft that was attached to the ground and used the wind to create actual aerodynamic forces and moments similar to aircraft. One of the first known devices using this technique was the Sander's Teacher [1]. The problem with this technique was that in order for it to be successful there had to be a constant wind to make the simulator work. This type of simulation was abandoned due to the unreliable nature of wind.

The next stage in simulation was developed in the mid-1920's and it involved using mechanical or electrical actuators to orient the trainer in the orientation that was given from the

pilot inputs. The most famous of these trainers was known as the Link Trainer (see Figure 1) that was developed in 1929.

Figure 1. Link Trainer (http://www.britannica.com)

The main issue with these first simulators is that it was not actually simulating aircraft dynamics. To get the response similar to the aircraft, pilots would use the simulator and tune the simulator to what the aircraft felt like. This was not the best scenario because although pilots would get the simulator close to the aircraft's abilities, the important modes of motion (the short period, phugoid, and dutch roll) could not be reproduced accurately. The Link trainers were used widely in World War II due to the high volume of pilots that were needed at the time to be trained.

During World War II, analog computers were developed along with electronics were able to lead flight simulation in the right direction. With analog computers, the aircraft equations of motion could be solved and therefore could then be simulated. This would replace the pilots "intuition" of response of the aircraft to a more realistic model of what the aircraft should do. After World War II the first full flight simulator was developed for Pan American Airways through Curtiss-Wright for the Boeing 377 Stratocruiser [1] and was the first full simulator to be owned by an airline. There was a major issue with analog flight training simulators, the simulators had a lot of down time due to maintenance associated with analog computers.

With the reliability of analog flight simulators very low and the invention of digital computers, it was only logical that flight simulators would be created using digital instead of analog computers. The evolution from analog to digital flight simulation was initiated by the United States Navy in 1950 with a research into creating a reliable flight simulator using a digital computer. At that time though, digital computers were not powerful enough and could not be used for real time simulation of aircraft dynamics. In the early 1960's digital computers were becoming more powerful and thus real time simulation became more feasible. The first well received real time digital flight simulator was the Link Mark 1[1]. It was a very good simulator for its day and was purchased by both the U.S. military and airlines and is still the same basic concept used today in modern flight simulators.

In the 1950s motion systems were nonexistent for and flight training devices were known as fixed base systems. The reasoning behind this was that the general consensus at the time was that motion was not needed because pilot did not fly "by the seat of their pants," or by feeling the forces exerted when doing maneuvers, anymore and thus it was not needed. This notion was

overturned at the end of the 1950s and in 1958 a basic pitch motion system was created for the Comet IV simulator as seen in Figure 2.

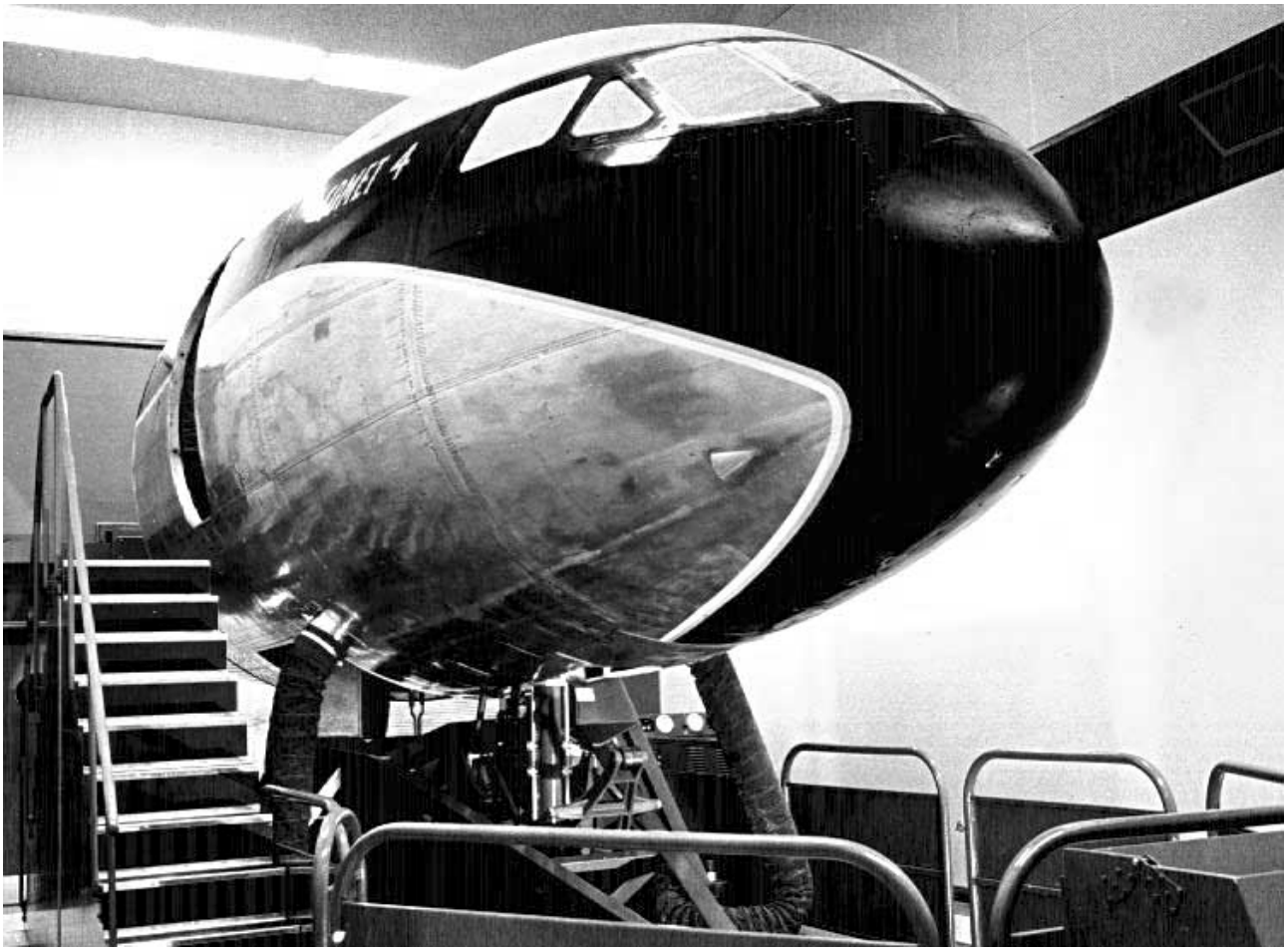


Figure 2. Comet IV Simulator with Basic Pitch Motion System.
(http://homepage.ntlworld.com/bleep/SimHist8.html)

With the pitch motion system of the Comet IV simulator, the industry saw the need for motion systems and two and three degree of freedom motion systems were then created. As the need for more complex motion systems arose, the six degree of freedom motion system was created. With the six degree of freedom motion system, translation and rotation can be achieved in three dimensions. An example of type of six degree motion system is seen in Figure 3 and is known as the Stewart platform and is a common platform used in the industry. For the motion

systems, hydraulics or electric motors are used to orient the simulator depending on the size or requirements of the flight training device.

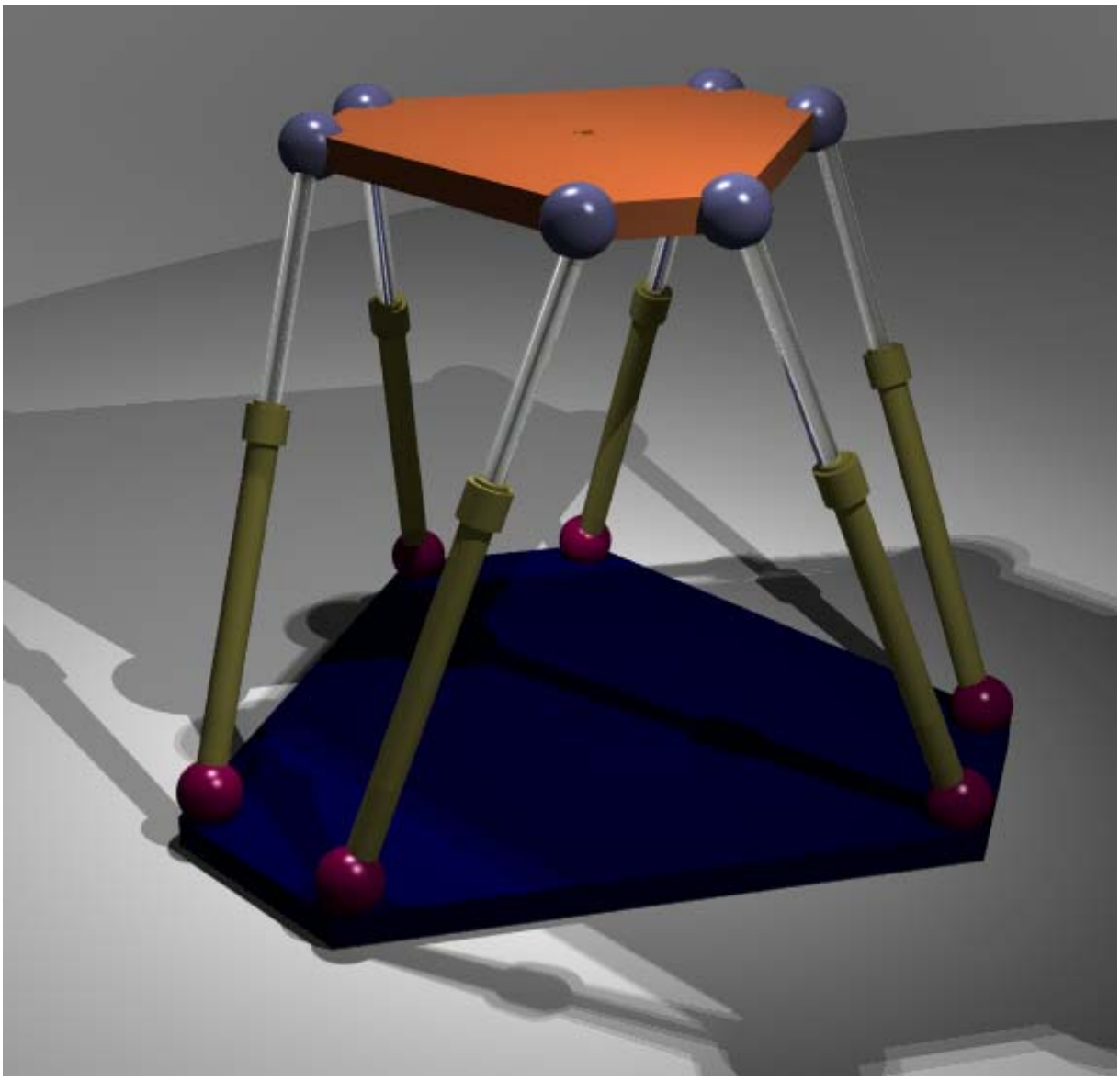


Figure 3. Six Degree of Freedom Stewart Platform.

Most early flight simulators did not use any sort of visual system as they were mainly used for instrument training. The first major visual system was point light source projection and was developed in the 1950s. The first modern visual system, i.e. computer video graphics was created by General Electric specifically for the space program. Modern visual aide systems

consist of computer video graphics which are run by the aircraft dynamics to simulate visually what the aircraft is doing. Thus, digital simulation along with modern motion and visual systems are still an effective method for implementing a flight training device.

Along with better modeling of aircraft, the Federal Aviation Administration started looking at flight simulators as a serious training method for pilots. One of the first regulations set by the FAA that was imposed on flight simulators occurred in 1973. This requirement was that the simulator had to handle like the aircraft it was simulating well enough that landing could be done in the simulator for pilots that needed to update their qualification of experience. Thus a very integral part of flight training, the landing, was finally able to be taught in a less stressful environment and therefore increased the knowledge retained by the pilot using the simulator.

The FAA decided that there should be more regulations on how a simulator was designed and implemented and in June of 1980 the Federal Aviation Regulations Part 121 (FAR) were amended and the FAA flight simulation plan was created [17]. With this amendment, flight simulators then had to pass more stringent requirements because a set of qualifications had to be met in order to be considered for use as flight training devices. As flight simulators become more advanced, more requirements have been added. Part of these requirements is to simulate the aircraft very accurately to get optimal training for the specified aircraft. Therefore the FAA saw a need to create objective requirements. The requirements of the simulator tend to be limits/thresholds on errors of the modal parameters for slow oscillations and/or limits on time histories of relevant parameters in specific maneuvers in comparison to flight test data of the given aircraft. Subjective requirements call for handling qualities assessment for specific maneuvers that are done by pilots and are analyzed by the pilots to get the right “feel” of the aircraft.

3. General Structure of a Computational Environment for the Development of an FAA Compliant Aircraft Simulation Model

A strategy had to be devised to take the flight data given by the manufacturer of the aircraft and create an aircraft dynamics model that would be able to pass FAA regulations. To begin the process, the flight data that was available had to be processed and converted from the file format that resulted from the data acquisition process to a format that was acceptable and convenient to be used within Matlab.

To process the data, the state variable and control variables of the flight test data were plotted. By analyzing the state and the control variables in these plots and considering the purpose of each flight test, (for example short period or dutch roll), a segment of the data would be selected for the FAA tests. Once the data segment was selected, Matlab would then convert the segment into .mat files with all of the desired state and control variables.

After the data has been processed, the next stage was to take the selected data for the FAA tests and then use parameter identification software to find the aircraft dynamics model. To do this, two approaches to parameter identification were used. These approaches were the least squares estimator in the time domain and the recursive Fourier transform in the frequency domain [14]. Using a combination of the results from both approaches and a background in aircraft dynamics (both analytical tools and heuristics), the model was created using the state space representation.

Once the basic aircraft dynamics were created, several subsystems had to be modeled and added to the flight simulation in order for the simulation to be used as a flight training device. The main subsystems in the model were the landing gear model and the engine dynamics model

that were developed separately from the aircraft dynamics model. After the subsystems were added, the simulation had to be run with the control inputs of the selected flight test data. The simulation was then initiated with the flight test data control inputs and the simulation data was compared with the flight test data and was checked for compliance with the FAA regulations. The only requirement by the FAA in regards to comparing the flight test data with the flight simulation data is that it has to be a snapshot of the response of the aircraft, meaning that the coefficients found using the parameter identification had to be compared with the original flight test data only.

If the aircraft model did not comply with FAA regulations one of two choices had to be made. The first was to use knowledge of aircraft dynamics to tweak the model parameters to meet the requirements. There were several possible causes of the model of the aircraft not matching the flight test data. Noise and sensor bias in the flight test data are large contributors to the problems with the model. The second choice was to alter the control inputs slightly. The second choice is most feasible for small changes to the response of the aircraft model. Changing the control inputs for the simulation is tolerable due to a possible bias in the control input sensors and is acknowledged by the FAA regulations. Tweaking the model is a more effective way of meeting the FAA regulations due to the stringent requirements set.

Once the model has been tweaked and complies with the FAA regulations, pilots that are qualified experts on the aircraft, then test the model subjectively by using the simulation with hardware control inputs. The pilots attempt to fly the aircraft as it should be flown and if it does not “feel” like the aircraft, the aircraft model is then altered to meet the subjective testing of the pilots and the model is retested to meet the objective FAA requirements.

Finally, after both the subjective tests of the pilots and the FAA test requirements have been achieved – possibly after several iterations – the model is completed and the final phase begins. In the final phase the FAA requires the data from the simulation to be in a certain format for ease of certification of the flight simulator. Figure 2 shows the general block diagram for the strategy of creating a flight simulation model from flight test data that can be evaluated for compliance with FAA flight simulator performance requirements.

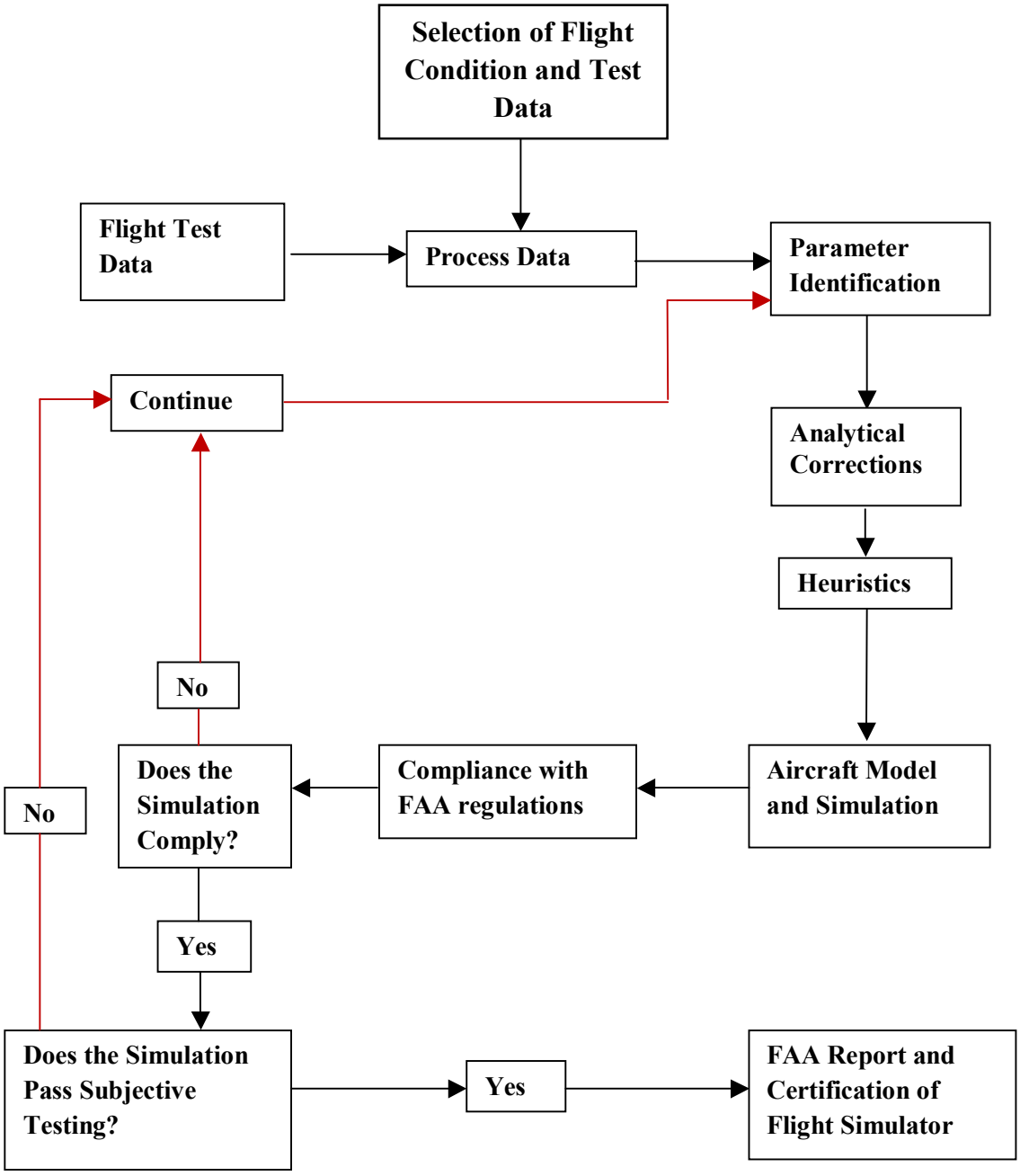


Figure 2. Block Diagram of Flight Simulation Design Strategy.

4. Computational Tools for Flight Data Processing

Flight data was given on a number of flights and for each flight, the data were compiled into one file that contained all of the parameters that were measured for that flight. The content and the general format of the flight test data file is typical for what is currently used in the industry. It should be noted that, throughout this thesis, all flight test data have been normalized and/or altered such that the proprietary rights of the manufacturer are protected.

The flight data were recorded in a comma separated variable file (.csv) and Matlab does not directly support the .csv file. Matlab has a function called csvread that takes .csv files and converts them to Matlab data files (.mat files). The main issue that came up was that the files were not just data. The data was accompanied with a heading that displayed the flight number and what each column of data represented, i.e. angle of attack, control deflections, engine parameters, etc. Matlab's csvread command would only read data and therefore the data could not be read just using the plain csvread command. Matlab's csvread did have an option that would read a certain range of data in the file. The range was set up in a [R1 C1 R2 C2] format where R1 and C1 were the upper-left bounds of the data and R2 and C2 were the lower-right bounds of the data. Once the range was set up, the data was then able to be read and reduced by Matlab.

Since there were many flight test data files and so many measured outputs in the .csv file, finding desired data would be difficult without the help of a program to reduce the time searching for desired data points. Another issue was that there are large amounts of data in each flight test that part of the program had to be designed to take desired data points, such as a non-cross controlled short period maneuver, and convert them into a matrix in a Matlab .mat file for

use in the parameter identification process. A program was created to do that and it will be referred to as the Plot Data program.

To start the Plot Data program, a menu was created to initialize all of the ranges of data for each of the flight test data files as seen in Figure 3. After the program has been initialized and the continue button is clicked, the program brings up another menu, which is shown in Figure 4. In this menu, each of the parameters required for FAA tests can be selected along with the flight data file. Figure 4 shows all of the parameters that are required for FAA compliance tests. Once the desired file and data are selected the continue button becomes active and when clicked loads the desired data and plots the data. The data is plotted on separate plots in Matlab for ease in finding data that can be used for parameter identification and FAA compliance.

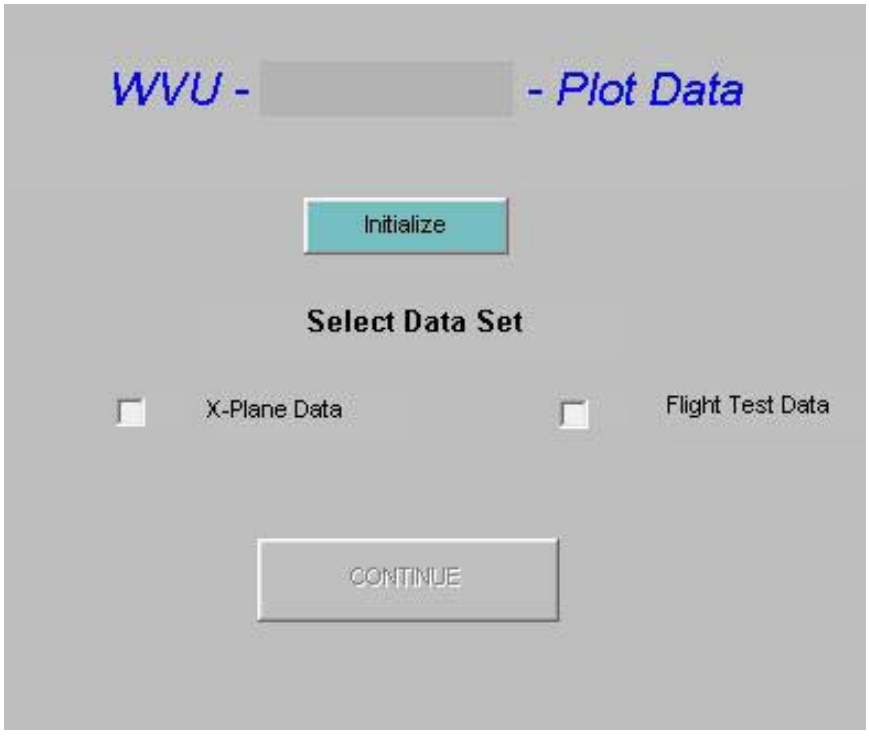


Figure 3. Initializing the Plot Data Program.

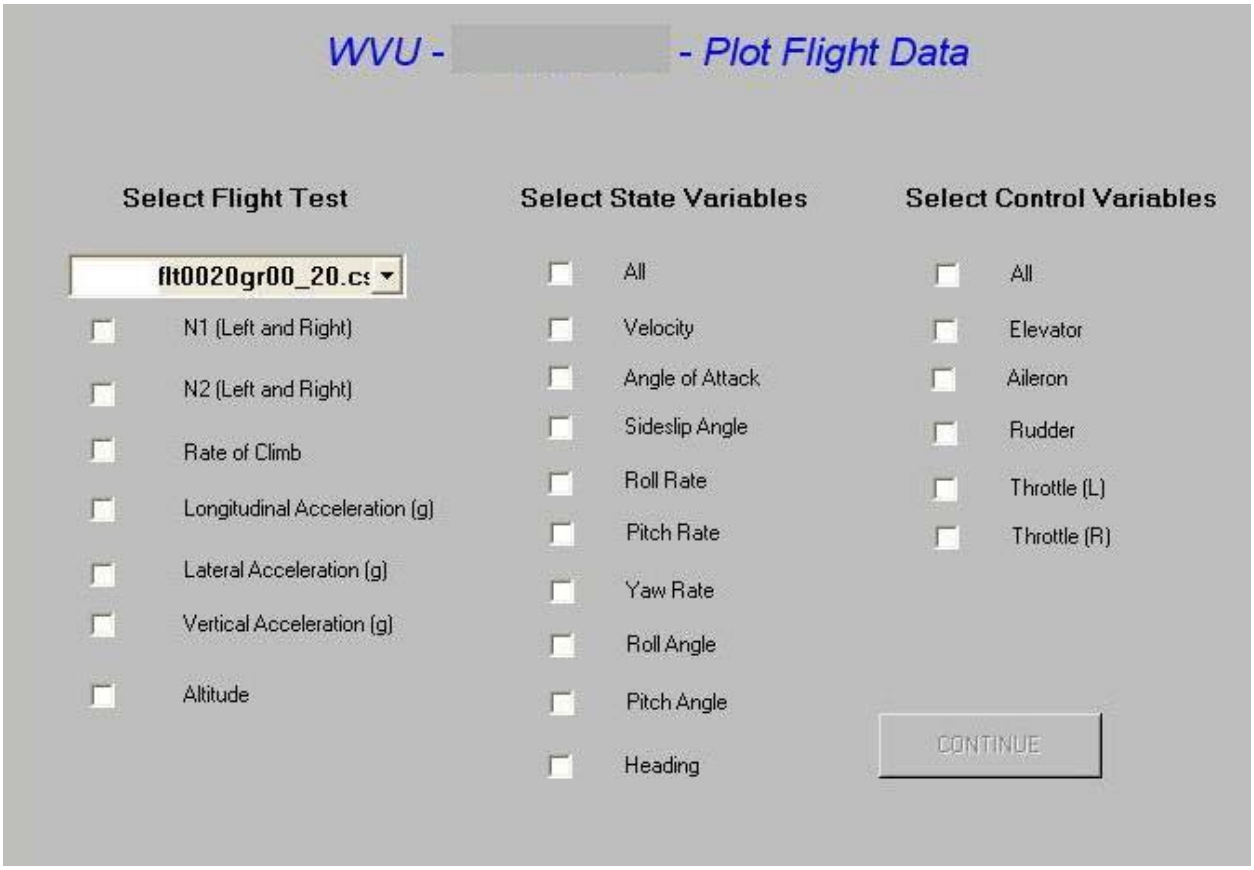


Figure 4. Selection of Flight Test Data Variables.

For example, Flight Test #1, which is altered flight test data by using a multiplier, was the flight test data for the short period response of the aircraft. Plotting the state and control variables shows the response of the aircraft very clearly. For this example, Figures 5 through 7 show the pitch angle, pitch rate, and elevator deflection respectively. The rest of the state and control variables for Flight Test #1 can be seen in Appendix A.

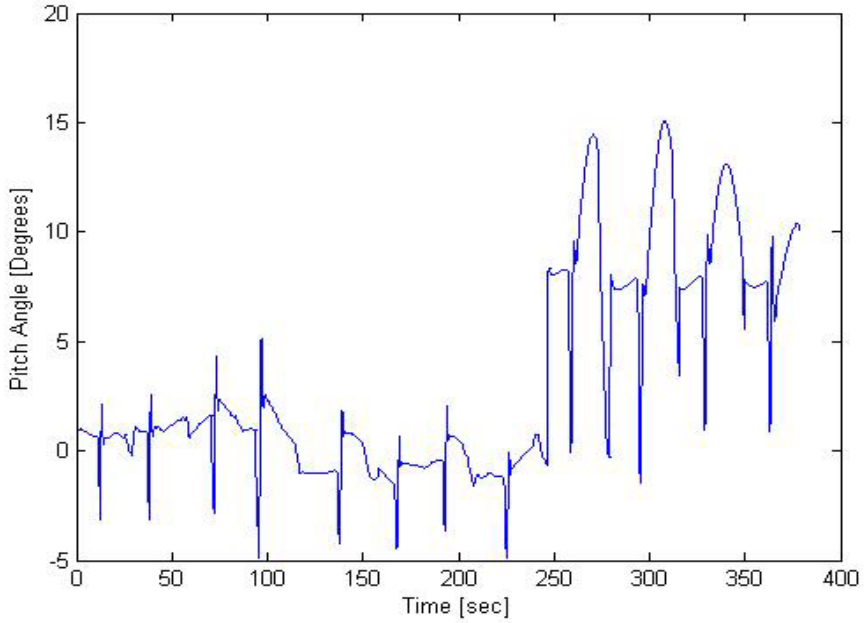


Figure 5. Pitch Angle for Altered Short Period Flight Data.

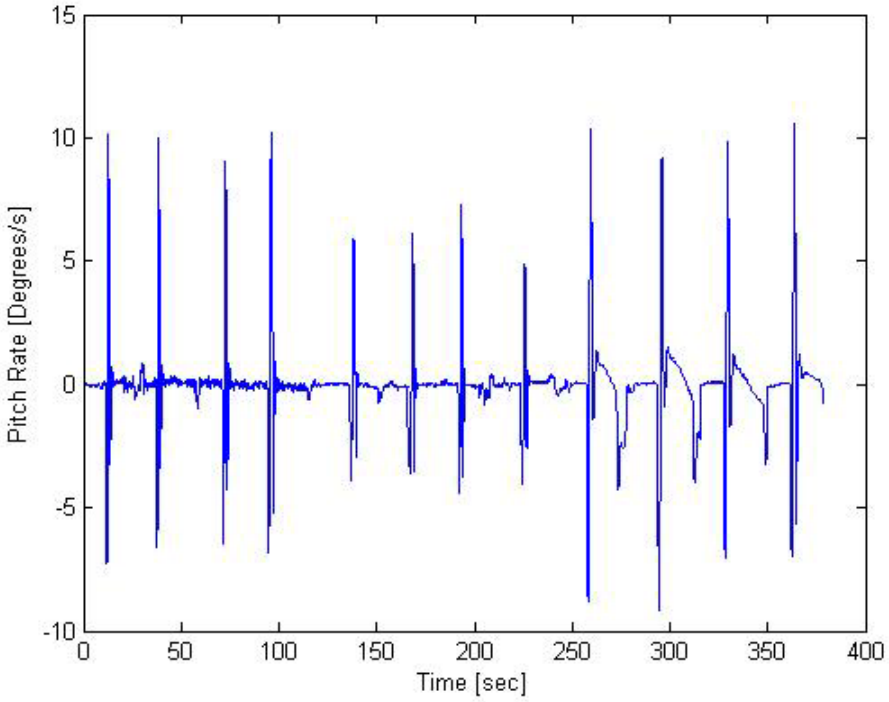


Figure 6. Pitch Rate for Altered Short Period Flight Data.

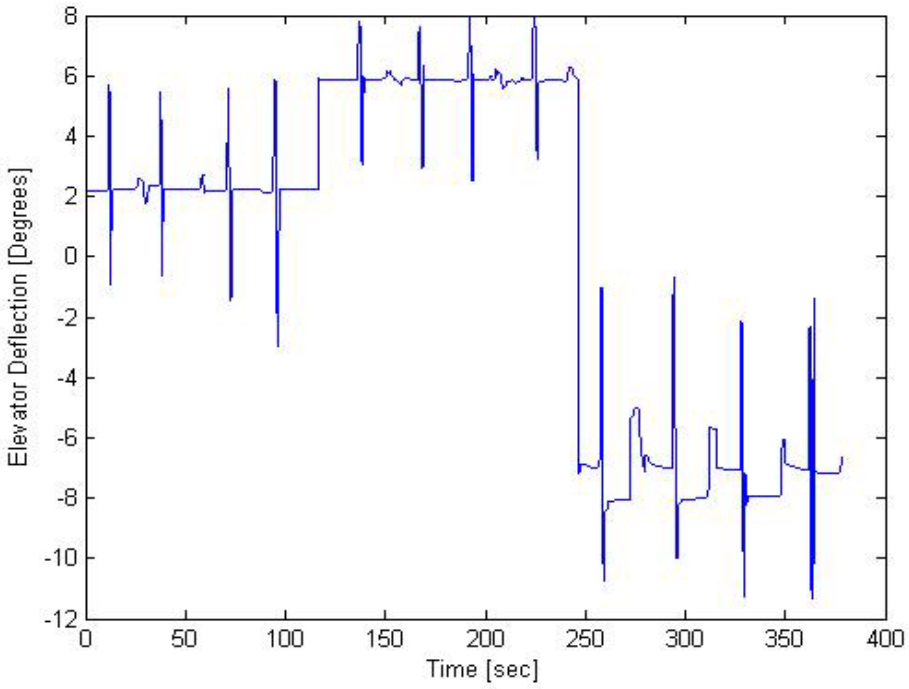


Figure 7. Elevator Deflection of Altered Short Period Flight Data.

After analyzing the flight test data, there are 12 short period maneuvers in the data set. For parameter identification it is best to select data that closest meet the flight condition. The flight conditions are cruise, approach, landing, and steady state sideslip. Constraints on the general conditions of the test have been considered for each particular FAA objective test. In this example pertinent to the short period evaluation, these constraints are:

- General consistency of the data with the baseline steady state horizontal, symmetric, rectilinear, uniform flight
- The existence of an initial and final (before and after the maneuver) steady state segment of adequate duration
- Absence of crossed inputs (Avoid data with multiple inputs, for example not having both aileron and rudder deflection in the desired data)
- Non-violation of linear domain boundaries

- Limited input sensor bias

After analyzing the whole flight data file, a maneuver within the flight data can be selected by specifying the start and end time of the maneuver, within the interactive menu shown in Figure 8. The desired variables that need to be saved can be selected by clicking on the checkboxes. Figure 8 has the same basic format as Figure 4 except for some slight variations. The first variation is the "Save as" column. The "Save as" column shows what the name of the vector will be in the .mat file. The next difference is the "Save in" column that creates the name of the .mat file. The following tables show what vectors are saved in each save data file.

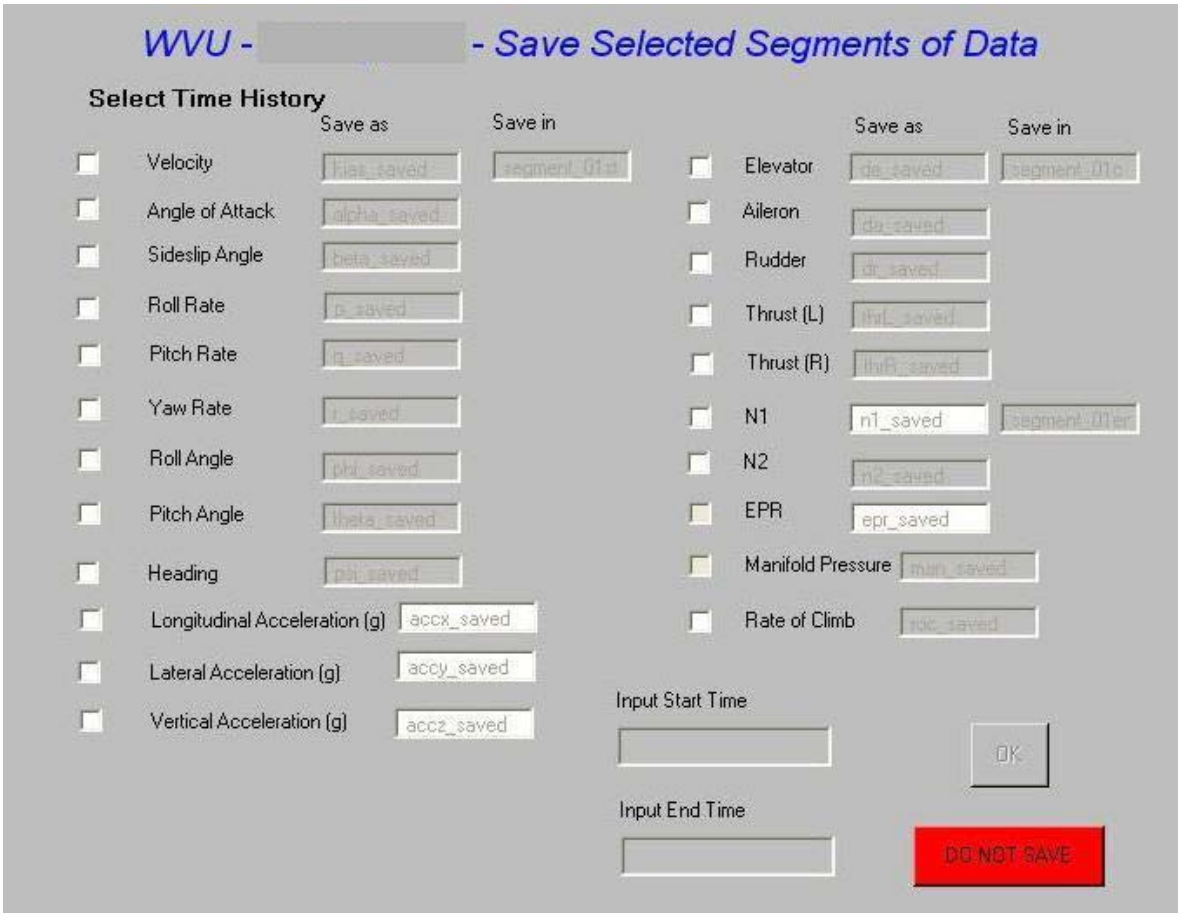


Figure 8. Reducing Flight Data to Desired Data Segments.

Table 1. State Variables Save File Vectors.

Segment_01st file	
Variable	Save as
Velocity	kias_saved
Angle of Attack	alpha_saved
Sideslip Angle	beta_saved
Roll Rate	p_saved
Pitch Rate	q_saved
Yaw Rate	r_saved
Roll Angle	phi_saved
Pitch Angle	theta_saved
Heading	psi_saved
Longitudinal Acceleration	accx_saved
Lateral Acceleration	accy_saved
Vertical Acceleration	accz-saved

Table 2. Control Variables Save File Vectors.

segment_01c file	
Variable	Save as
Elevator	de_saved
Aileron	da_saved
Rudder	dr_saved
Thrust(Throttle) Left	thrL_saved
Thrust(Throttle) Right	thrR_saved

Table 3. Engine Parameters and Rate of Climb File Vectors.

Segment_01eng file	
Variable	Save as
N1	N1_L_saved and N1_R_saved
N2	N1_L_saved and N1_R_saved
EPR	epr_saved
Manifold Pressure	man_saved
Rate of Climb	roc_saved

In Table 3, the rate of climb was added to the engine parameters because it was added to the program at the same time the engine parameters were added. As an example, from the altered flight data of Flight Test #1 shown in Figures 5 through 7, the time of 130 sec was selected as the start time and the end time was selected as 150 because it encapsulates one of the short periods in the flight test. This segment was selected because it contains a very clear short period maneuver complying with all the requirements formulated earlier. Figures 9 through 11 show the segment of data that were selected for pitch, pitch rate, and elevator deflection respectively of the selected short period segment and the rest of the altered flight tests data of the same segment can be seen in Appendix B. Several such segments were used in the PID process, however, the FAA regulations only require that compliance be demonstrated for one case.

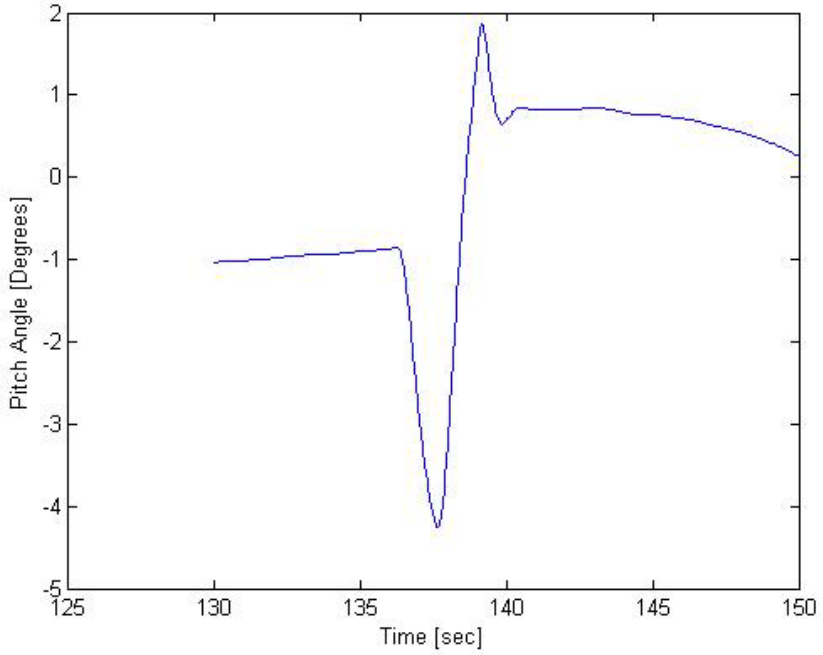


Figure 9. Pitch Angle Segment of Altered Short Period Flight.

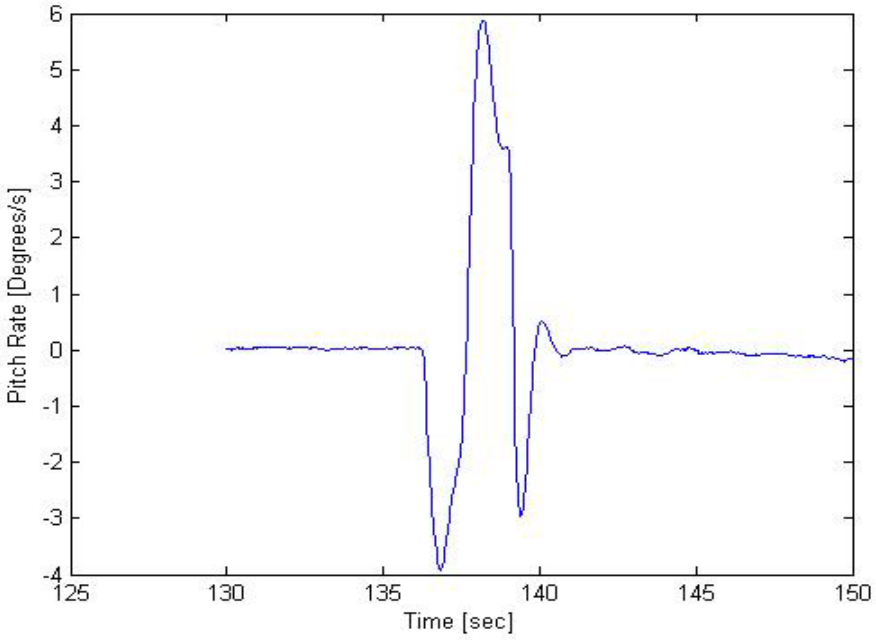


Figure 10. Pitch Rate Segment of Altered Short Period Flight.

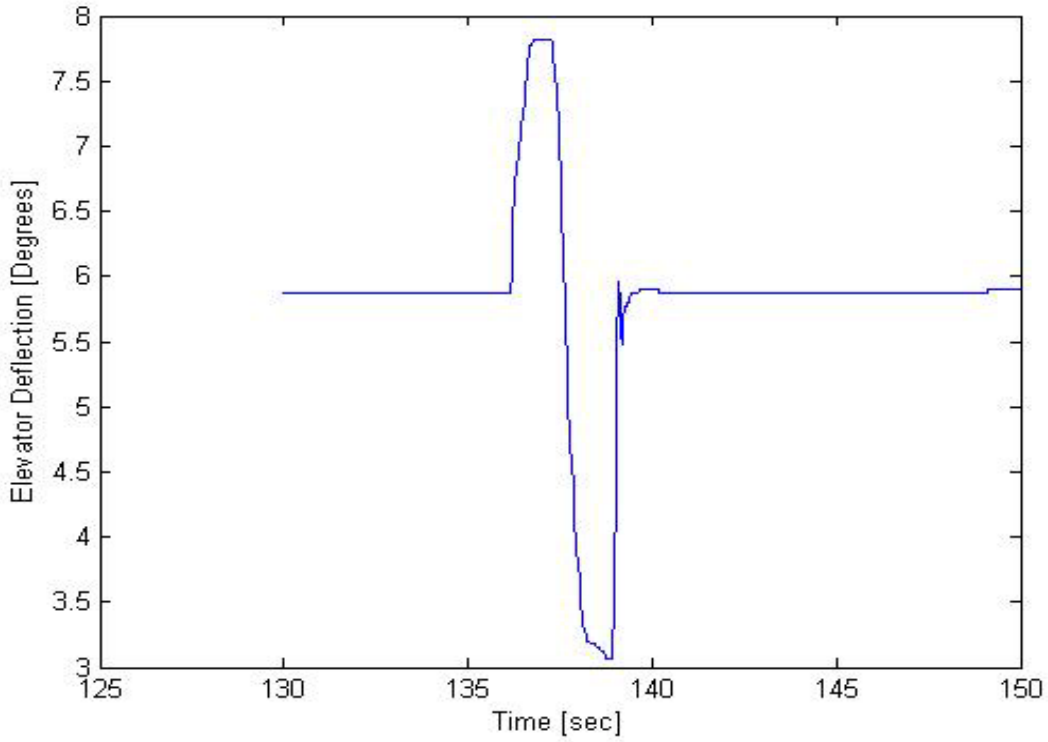


Figure 11. Elevator Deflection Segment of Altered Short Period Flight.

Figure 12 shows the final menu in the Plot Data program. When the first option is selected the program reloads all of the same flight test data and brings up the menu to select certain data from the flight test data. The second option brings up the menu to select a new flight test data at the menu from Figure 4. The third option closes all of the figures open including the menu and stops the program.

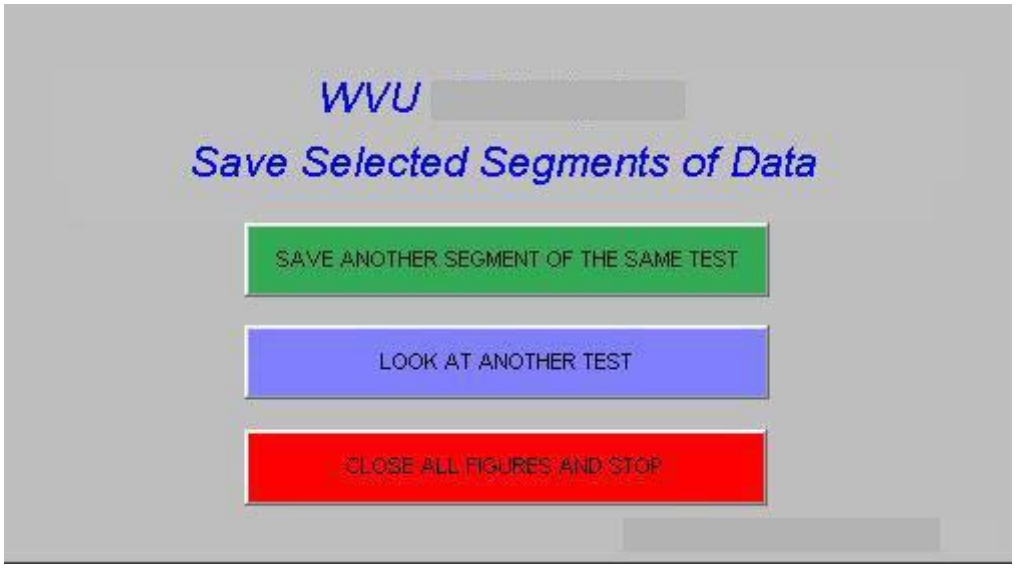


Figure 12. Final Choice Menu.

Once the program is finished, the selected flight test data for each test is processed and is now compatible with Matlab. The processed data can now be put through parameter identification software to regress the flight data into a mathematical model of the aircraft.

5. Parameter Identification

PID is a group of linear regression techniques that take a known output data, in this case flight test data of a business jet aircraft. Knowing how the system should respond mathematically, a set of unknown parameters, the stability and control derivatives, can be regressed to create a linear equation approximation of the data. A simple state variable model of aircraft dynamics can be seen in Appendix C. Given the flight test data of the aircraft and geometry, parameter identification was well suited to what was required. Two methods of parameter identification were used to determine the preliminary aircraft aerodynamic model. Both methods were used due to the reliability of both approaches and it gave the ability to check the results and give them The first is the least squares estimator. This approach takes the data and tries to do a best-fit linear regression for all data points with the assumption of a linear representation of the forces and moments as functions of state and control variables.

The other approach is called the recursive Fourier transform. The recursive Fourier transform takes the same approach as the least square estimator but applies it to the frequency domain. These approaches to parameter identification were selected because of the similar performance between the two techniques in accuracy and convergence [18]. Both the recursive Fourier transform and the least squares estimator both use the same basic equation to solve for the unknown parameters which is:

$$z = H\theta + v \tag{1}$$

5.1. Least Squares Estimator

Parameter estimation using the least squares estimator assumes that Θ is a vector of unknown constant parameters, H is a known matrix, v is the measurement noise, and z is the measurement. An assumption made in the parameter identification is that v = 0. The best way to estimate Θ is to minimize the weighted sum of squared differences between the measured outputs and the model outputs. This is accomplished by using the equation known as the cost function J(Θ):

$$J(\theta) = \frac{1}{2}(z - H\theta)^T R^{-1}(z - H\theta) \tag{2}$$

where R^{-1} is a positive definite weighting matrix (4). An assumption was made for this project where the difference will be equally weighted therefore R^{-1} =1 and the equation simplifies to:

$$J(\theta) = \frac{1}{2}(z - H\theta)^T(z - H\theta) \tag{3}$$

This is known as ordinary least squares. For ordinary least squares the H matrix is changed to the X matrix that is a matrix of vectors and regressors. To minimize the sum of square differences between the measurements and the model must satisfy.

$$\frac{\partial J}{\partial \theta} = -X^T z + X^T X\hat{\theta} = 0 \tag{4}$$

Solving for $\hat{\theta}$, the parameter vector, the equation reduces to:

$$\hat{\theta} = (X^T X)^{-1} X^T z \tag{5}$$

Using Matlab to find the parameters of the least square estimator the C_m values normalized values Table 4 and the normalized C_m are compared against the computed value of C_m from measured states and controls (flight data) in Figure 13.

Table 4. Normalized Pitching Moment Stability Coefficient and Control Derivatives Using Least Squares.

Cm	
0	0
V	-0.001
α	-0.01214
q	-1
α_dot	0.146866
δe	-0.00049

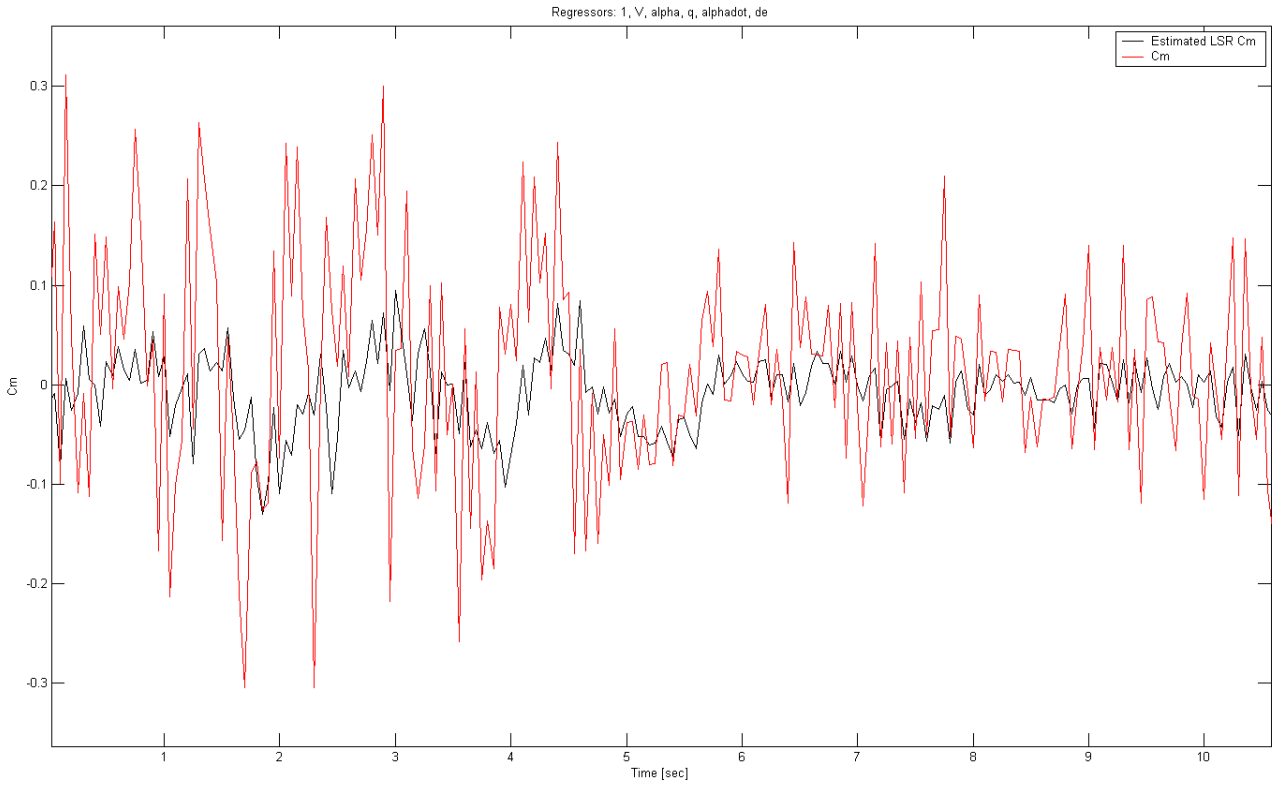


Figure 13. Comparison of Normalized Least Squares Estimated and Measured Pitching Moment Coefficient

5.2. Recursive Fourier Transform

The Fourier transform of a signal is defined by:

$$\bar{x}(\omega) = \int_0^T x(t)e^{-j\omega t}dt \qquad (6)$$

Transforming the Fourier transform to a discrete form that is required for data that has sampled yields the equation,

$$\tilde{x}(\omega) \equiv \Delta t \sum_{i=0}^{N-1} x(i) e^{-j\omega i \Delta t} \tag{7}$$

and defining,

$$X(\omega) \equiv \sum_{i=0}^{N-1} x(i) e^{-j\omega i \Delta t} \tag{8}$$

Therefore the finite Fourier transform approximation is,

$$\tilde{x} = \Delta t X(\omega) \tag{9}$$

For regression in the frequency domain, the least squares estimation equation must be changed due to complex numbers in the system, which the equation now becomes,

$$\tilde{z} = \tilde{X}\theta \tag{10}$$

Where $\tilde{z}$ is the measurement data in the frequency domain and $\tilde{X}$ is the matrix of finite Fourier transforms of the unknown parameter matrix. The cost function, J(Θ), for an ordinary least squares becomes,

$$J(\theta) = \frac{1}{2}(\tilde{z} - \tilde{X}\theta)^{\dagger}(z - \tilde{X}\theta) \tag{11}$$

Where † is the conjugate transpose of the matrix. Minimizing the sums of the cost function by taking the derivative of the cost function matrix and solving for the parameter vector, $\tilde{\theta}$ yields,

(12)

$$\hat{\theta} = Re[\hat{X}^{\dagger}\hat{X}]^{-1}Re[\hat{X}^{\dagger}\dot{z}]$$

Parameter Identification has one major issue in finding the stability derivatives. In order for parameter identification to work the stability derivatives much benefit from a direct cause and effect relationship between the aircraft dynamics and the control surface deflections. The issue is for certain coefficients, such as the drag coefficient, since the drag cannot be directly affected from control surface deflections. This is an issue that was resolved by using empirical methods to derive the drag coefficients of the aircraft.

Using PID software designed for aircraft parameter identification in Matlab and Simulink the stability derivatives can be found. To start the PID program, the initial conditions of all of the state variables must be set into an initial conditions file. Once the initial conditions file is completed and the data processed from the flight test data is loaded, the least squares estimator is run in Matlab. Once the least squares estimator has been run in Matlab, the recursive Fourier transform approach is then run in Simulink as seen in Figure 14. The recursive Fourier transform is run in real time, therefore as the data is analyzed it can be plotted and the normalized values at which the stability and control derivatives converge can be seen in Figure 15.

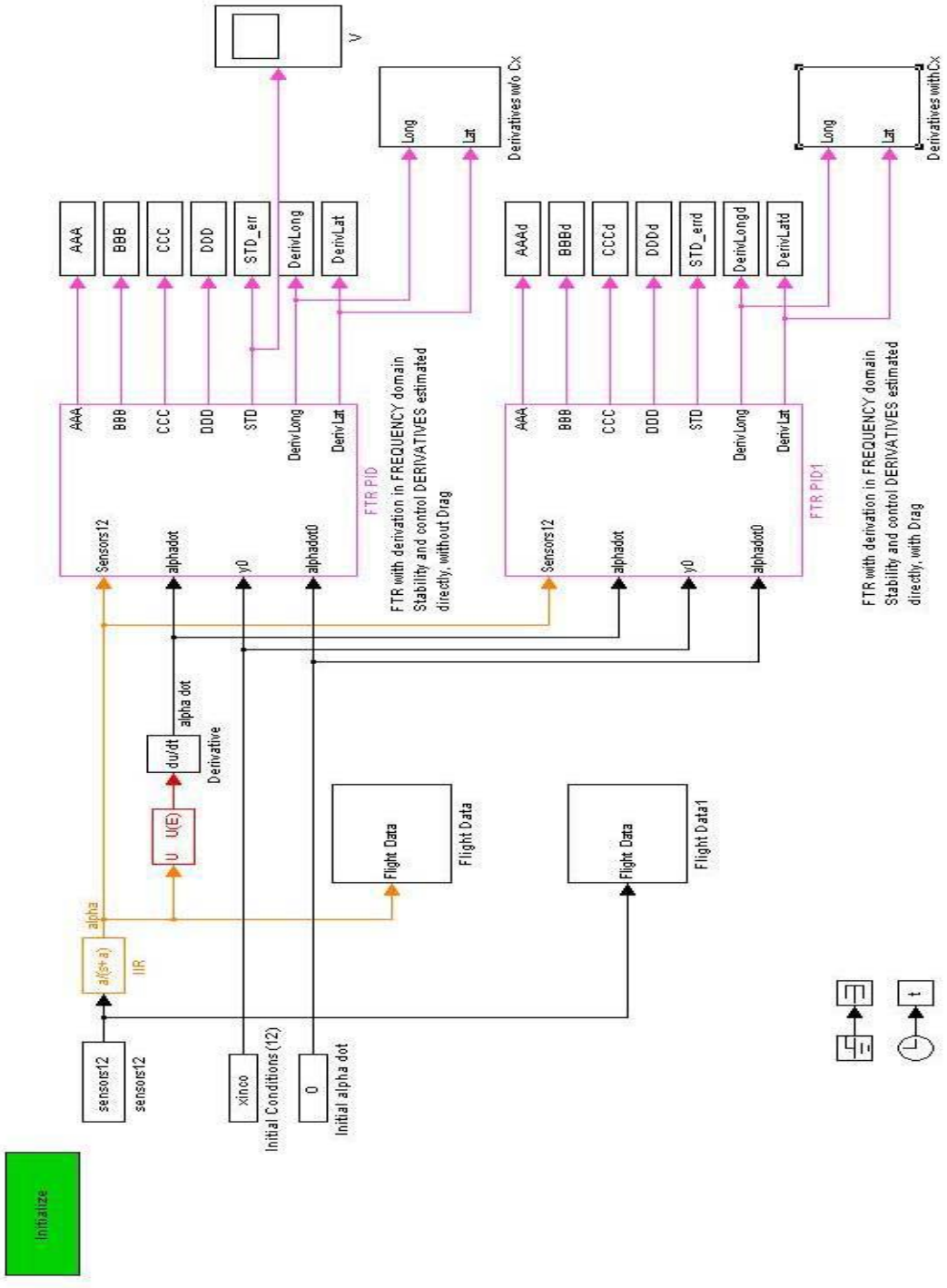

Initialize
sensors12
sensors12
Initial Conditions (12)
xinco
Initial alpha dot
0
a/(s+a)
IIR
alpha
U
U(E)
du/dt
alpha dot
Derivative
Flight Data
Flight Data
Flight Data1
Flight Data
AAA
BBB
CCC
DDD
STD_err
DerivLong
DerivLat
Sensors12
alphadot
y0
alphadot0
FTR PID
FTR with derivation in FREQUENCY domain
Stability and control DERIVATIVES estimated
directly, without Drag
V
Long
Lat
Derivatives w/o Cx
AAAd
BBBd
CCCd
DDDd
STD_errd
DerivLongd
DerivLatd
FTR PID1
FTR with derivation in FREQUENCY domain
Stability and control DERIVATIVES estimated
directly, with Drag
Derivatives withCx
t

Figure 14. Recursive Fourier Transforms in Simulink

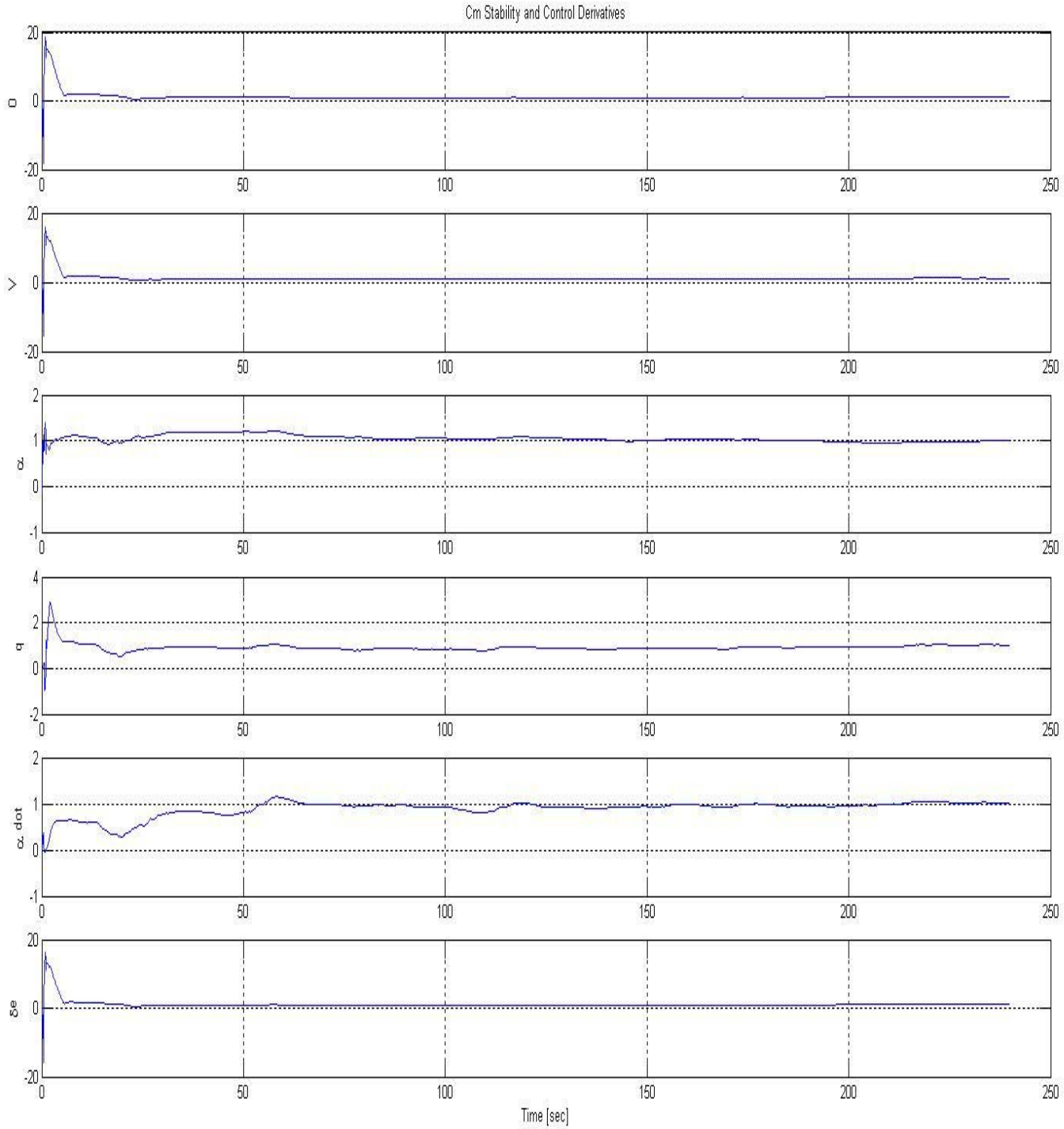


Figure 15. Normalized Pitching Moment Stability Coefficient and Control Derivatives Using Recursive Fourier Transforms.

5.3. Aircraft Parameter Identification

For aircraft parameter identification there are six main equations of motion that use parameter identification to find the stability and control derivatives. The six equations of motion are:

$$C_X = C_{X_\alpha}\Delta\alpha + C_{X_{\dot\alpha}}\Delta\dot\alpha + C_{X_q}\frac{\Delta q\bar{c}}{2V} + C_{X_\delta}\Delta\delta_e + C_{XV}\frac{\Delta V}{V} + C_{X_0} \tag{13}$$

$$C_Z = C_{Z_\alpha}\Delta\alpha + C_{Z_{\dot\alpha}}\Delta\dot\alpha + C_{Z_q}\frac{\Delta q\bar{c}}{2V} + C_{Z_\delta}\Delta\delta_e + C_{Z_0} \tag{14}$$

$$C_m = C_{m_\alpha}\Delta\alpha + C_{m_{\dot\alpha}}\Delta\dot\alpha + C_{m_q}\frac{\Delta q\bar{c}}{2V} + C_{m_\delta}\Delta\delta_e + C_{m_0} \tag{15}$$

$$C_Y = C_{Y_\beta}\Delta\beta + C_{Y_p}\frac{\Delta pb}{2V} + C_{Y_r}\frac{\Delta rb}{2V} + C_{Y_{\delta_a}}\Delta\delta_a + C_{Y_{\delta_r}}\Delta\delta_r + C_{Y_0} \tag{16}$$

$$C_l = C_{l_\beta}\Delta\beta + C_{l_p}\frac{\Delta pb}{2V} + C_{l_r}\frac{\Delta rb}{2V} + C_{l_{\delta_a}}\Delta\delta_a + C_{l_{\delta_r}}\Delta\delta_r + C_{l_0} \tag{17}$$

$$C_n = C_{n_\beta}\Delta\beta + C_{n_p}\frac{\Delta pb}{2V} + C_{n_r}\frac{\Delta rb}{2V} + C_{n_{\delta_a}}\Delta\delta_a + C_{n_{\delta_r}}\Delta\delta_r + C_{n_0} \tag{18}$$

Assumptions were made in the parameter identification process that helped simplify the process of parameter identification. A very common assumption is to decouple the longitudinal and lateral-directional dynamics. Another was that the stability derivatives that affect short

period and phugoid are not the same derivatives and therefore the assumption of decoupling the short period and phugoid can be made. Although PID is very accurate for some of the equations of motion, some of the equations derivatives cannot be accurately found. An example is represented by the derivatives of the longitudinal force component coefficient, C_x, which is typically not accurate because the drag of the aircraft cannot be accurately represented in the parameter identification process. Therefore, from experience, the drag of the aircraft was calculated and was applied as known in the parameter identification process and then the rest of the derivatives values could be found using PID. To get the parameters to converge in many of the tests; the selected flight test data for the test was concatenated several times to get the stability and control derivatives to converge to the correct values. After the stability and control derivatives were found, there were some values of the derivatives which appeared to be identified incorrectly, an assumption was made that these values were wrong and they were removed from the PID. The PID process was repeated with reduced number of parameters and it was noticed that the initially incorrectly determined parameters had little effect on the identification of the others.

After the values had been identified, a weighted average was taken of all of the values and a preliminary set of stability and control derivatives were created for the model. These preliminary values were then put into the model of the aircraft and the response was compared to the flight test data. If the model did not meet the FAA requirements, experience was used to alter the stability derivatives to improve the model's modes by using sensitivity analysis. The main derivatives that affect the longitudinal modes are C_{mq}, $C_{m\alpha}$, and C_{mV}. Derivatives that affect the longitudinal modes not as strongly as the main derivative are known as the secondary longitudinal derivatives which are: $C_{Z\alpha dot}$, C_{ZV}, and C_{XV}. Table 5 shows how the derivatives

affect each of the modal parameters of the short period and phugoid. For the lateral-directional stability derivatives, the most important are C_{lp}, $C_{n\beta}$, and $C_{l\beta}$. A secondary derivative for the lateral-directional derivative is C_{nr}. Table 6 shows how the derivatives affect the lateral-directional derivatives affect on the modal parameters.

Table 5. Longitudinal Stability Derivatives Sensitivity.

Stability Derivative		**ω_{nSP}**	**ζ_{SP}**	**ω_{nP}**	**ζ_{P}**
$\lvert C_{mq} \rvert$	↑	-	↑	-	-
$\lvert C_{m\alpha} \rvert$	↑	↑	-	-	-
$C_{m\alpha\_dot}$	↑	-	↓	↑	↑
$\lvert C_{mV} \rvert$	↑	-	-	↑	↓
$C_{z\alpha\_dot}$	↑	↑	↓	-	-
C_{ZV}	↑	-	-	↑	-
C_{XV}	↑	-	-	-	↑

Table 6. Lateral-Directional Derivative Sensitivity.

Stability Derivative (Increase)	**T_r**	**ω_{nDR}**	**ζ_{DR}**
C_{lp}	↓	-	-
$C_{n\beta}$	-	↑	↓
$\lvert C_{l\beta} \rvert$	-	-	↓
C_{nr}	-	↑	-

6. Development of a Simulation Environment for FAA Compliance with Objective and Subjective Tests

6.1. General Architecture

The results of the FAA compliance tests have been normalized due to the proprietary nature of the data given by the manufacturer. Also, any of the FAA compliance tests that require numerical evaluation of the data, for example damping or period, can only be expressed as percent difference or differences from the flight data values. The results are only for the aerodynamics of the FAA requirements.

All of the stability derivative coefficients that were found using the parameter identification software at different points in the flight envelope were put into look up tables with respect to dynamic pressure. The model was then generated in Simulink that could take recorded inputs from the flight test data, pilot commands from a joystick, or a mixture of the two. Two different implementations were developed including the same dynamic model. One was meant to be used on a regular desktop computer and was interfaced with the Aviator Visual Design Simulator (AVDS) [19] a commercial visualization software compatible with Matlab/Simulink. Figure 16 shows the general architecture of this aircraft model implementation including the engine model and landing gear model subsystem. Figure 17 shows the visual interface including AVDS. The second implementation was meant to be used with the WVU 6 degrees of freedom motion-based flight simulator. The Simulink model was customized to interact with X-Plane [20], the software that produces the visual cues within the motion-based flight simulator. This model is shown in Figure 18 [21].

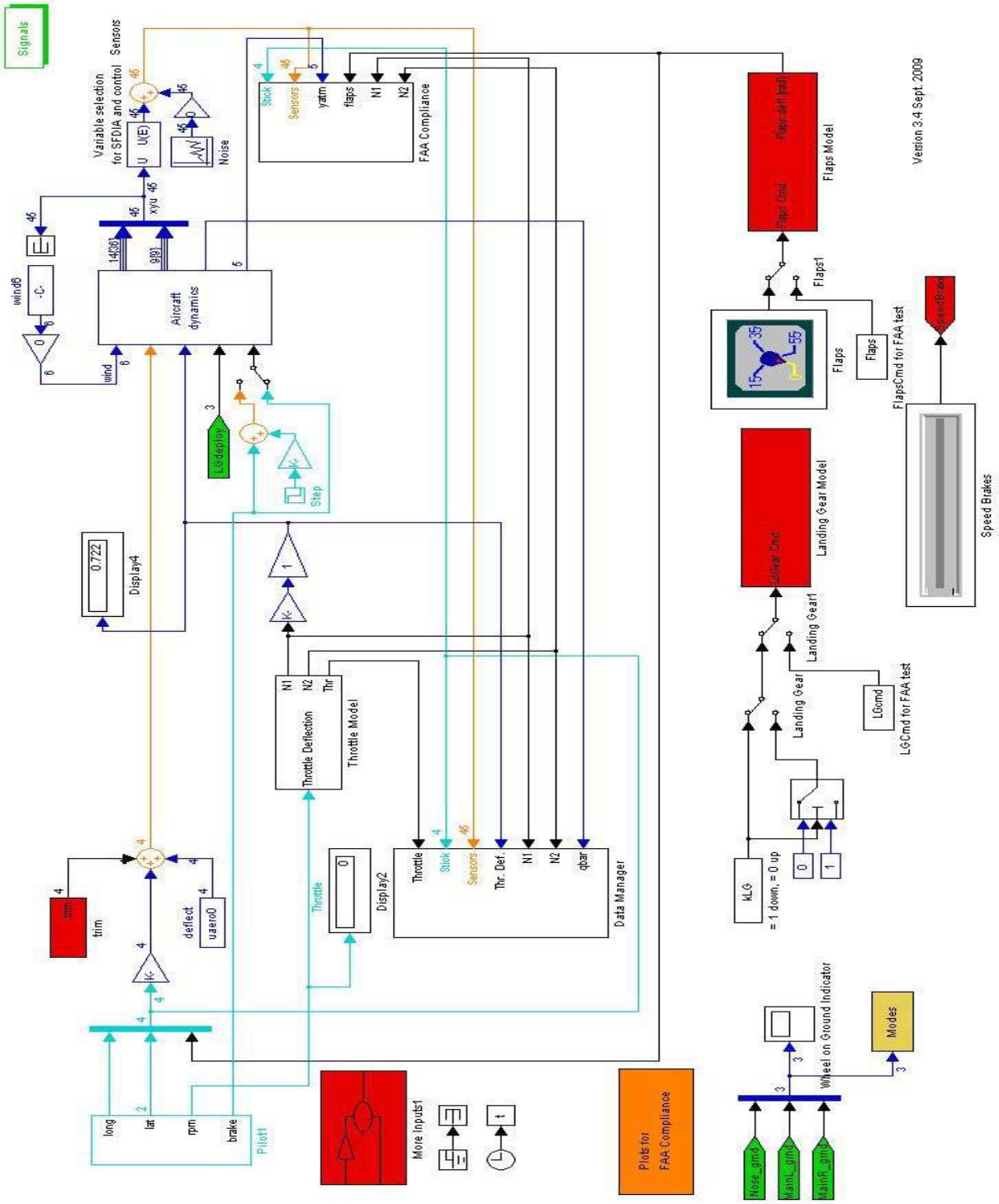

Signals
Variable selection for SFDIA and control
Sensors
Noise
Aircraft dynamics
wind6
wind
xyu
FAA Compliance
ytrim
flaps
N1
N2
Stick
Sensors
LGdeploy
Step
Display4
0.722
Throttle Deflection
Throttle Model
Thr
Display2
Throttle
Data Manager
Thr. Def.
qbar
trim
deflect
uaero0
long
lat
rpm
brake
Pilot1
More Inputs1
Plots for FAA Compliance
Flaps Model
Flaps Def feed
Flaps Cmd
Flaps1
Flaps
FlapsCmd for FAA test
Speed Brakes
Landing Gear Model
LGear Cmd
Landing Gear1
Landing Gear
LGcmd
LGCmd for FAA test
kLG
= 1 down, = 0 up
Wheel on Ground Indicator
Modes
Nose_grnd
MainL_grnd
MainR_grnd
Version 3.4 Sept 2009

Figure 16. Desktop Computer Implementation.

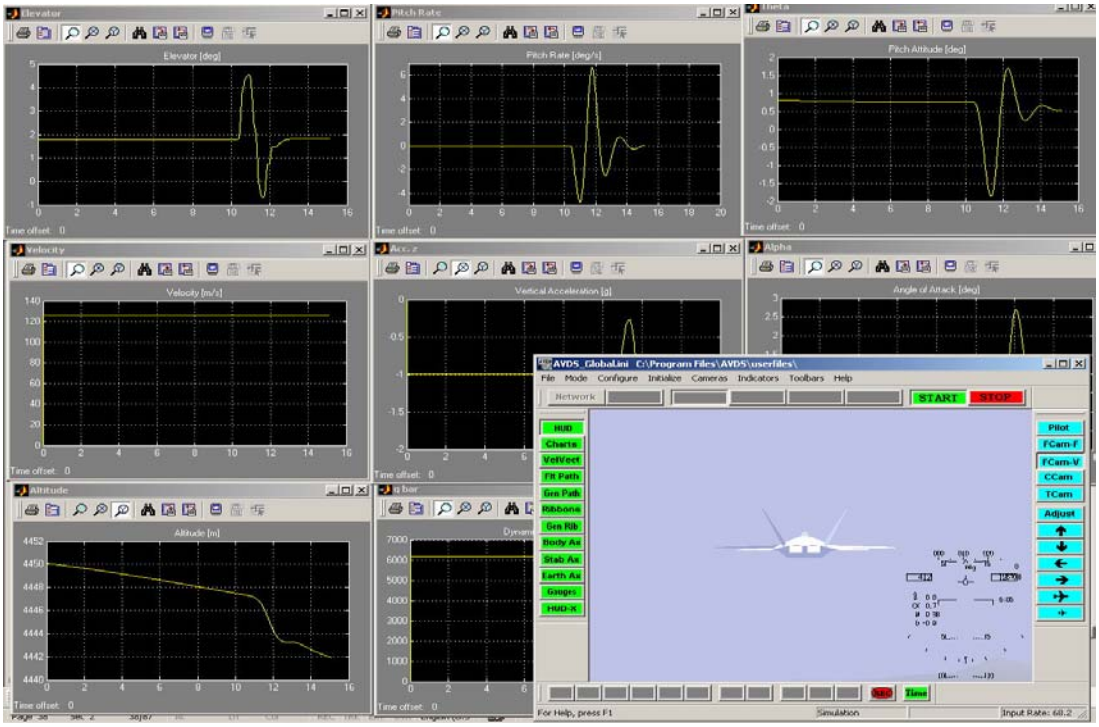


Figure 17. Desktop Computer Implementation – Interactive Visualization.

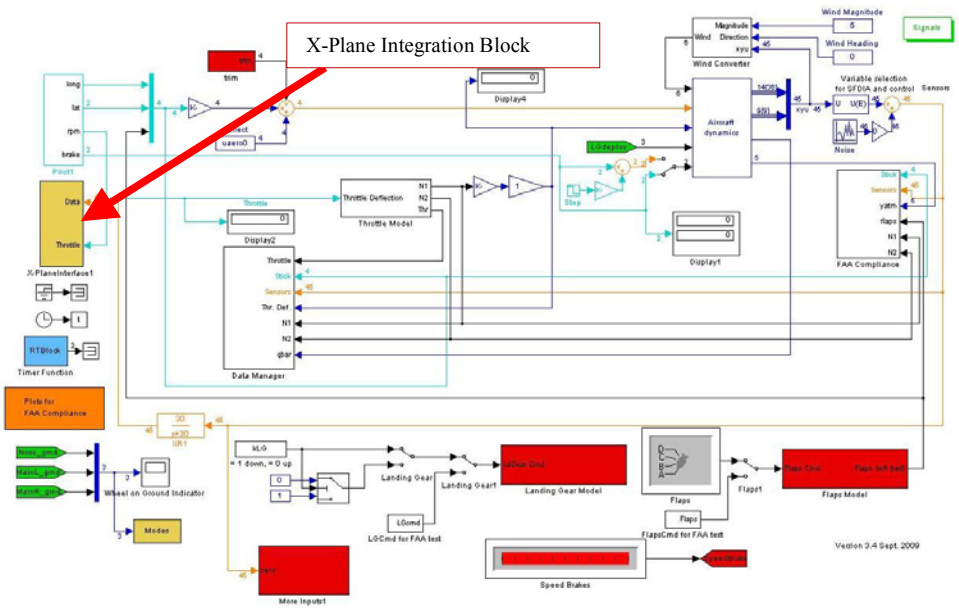


Figure 18. Motion-Based Flight Simulator Implementation [21].

To run the simulation, interactive menus were created to help load the initial conditions and the piloted input or FAA compliance inputs and flight test data. The menus and a brief explanation of how they are used are presented in this section and the user manual of the flight simulation can be seen in Appendix D. Figure 19 shows the initial menu for the flight simulator. In this menu, a choice of the FAA Compliance or piloted input is selected. The FAA compliance loads all of the flight data and the inputs from the flight data and the flight model. The piloted input choice loads all the pertinent information to the flight model only.

Figure 19. Initial Simulation Menu.

If the FAA Compliance is selected, the next menu(see Figure 20) selects the desired flight condition, i.e. cruise, approach, climb, steady state sideslip, or longitudinal trim.

Figure 20. Flight Conditon Menu.

Once the flight condition has been selected or the piloted input from Figure 19 is selected the next menu (see Figure 21) selects the desired inputs.

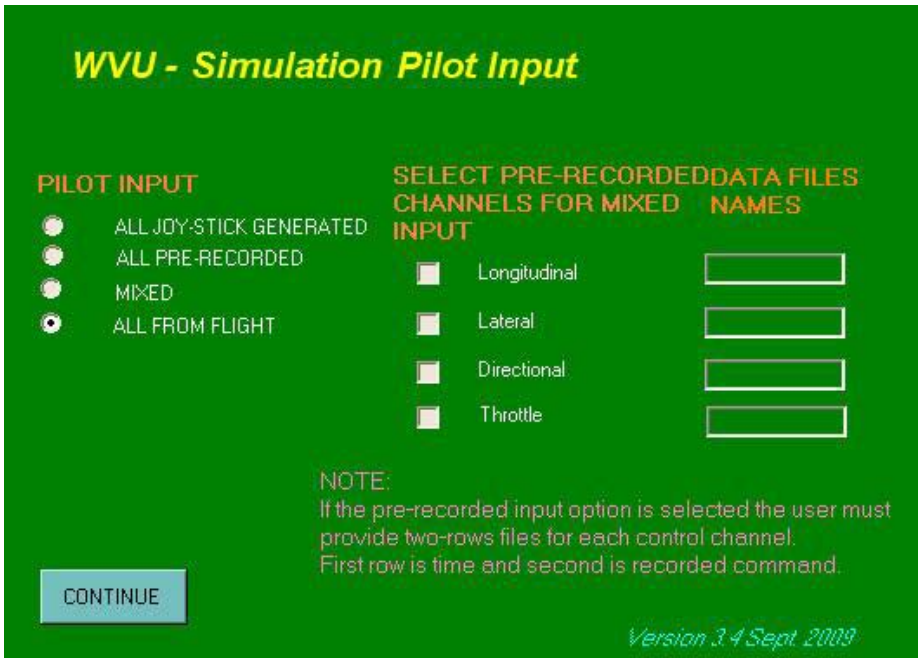


Figure 21. Input menu.

If the FAA compliance was chose this menu will require the all from flight option, while the piloted input will require the joy-stick generated choice. This menu can also have a mixture of pre-recorded inputs with joystick inputs. After the continue button is selected, another menu is opened. For the FAA compliance, the menu shown in see Figure 22 is opened. In this menu, the desired flight test requirement is selected for the flight conditon. The piloted input menu is the same except for the desired flight condtion is selected instead of the flight test requirement.

Figure 22. Flight Test Selection Menu.

6.2. Computation of Dynamic Characteristics of Slow Oscillatory Modes

FAA regulations require that the difference between the modal characteristics of the actual aircraft and the simulation model be within strict limits. In particular, there are criteria on the damping and natural frequency of both the phugoid and the dutch roll modes. These parameters must be computed for both the flight data and the simulation data. The so-called Peak-to-Valley method [22] was implemented for this purpose. Next, the Peak-to-Valley is

presented for the dutch roll mode; however, the algorithm is applied identically for the Phugoid, except that the variable to be considered is the pitch attitude angle instead of the sideslip angle.

The dutch roll mode can be approximated by a second order transfer function. The contribution of the dutch roll mode to the time history of the sideslip angle β can be expressed as:

$$\beta(t) = Ae^{-\zeta_{DR}\omega_{nDR}t}\sin(\omega_{dDR}t + \mu) \tag{19}$$

where ω_{nDR} is the "undamped" or "natural" Dutch Roll frequency, ω_{dDR} is the "damped" dutch roll frequency, ζ_{DR} is the dutch roll damping coefficient, A is the amplitude, and μ is the phase angle. The values of both A and μ depend on the initial conditions. The well known relationship between the "damped" and "undamped" frequencies is given by:

$$\omega_d = \omega_n\sqrt{1-\zeta^2} \tag{20}$$

Assume that a 'peak' and 'valley' of the sideslip angle time history succeed each other at times t_1 and t_2. They are one half cycle apart as shown in Figure 23. Using equation (19) to express DA_1 and DA_2, one can determine the *transient peak ratio* (TPR) to be equal to:

$$TPR = \frac{DA_2}{DA_1} = e^{-\zeta_{DR}\omega_{nDR}T_P/2} \tag{21}$$

To mitigate measurements error and improve accuracy, the value for *TPR* is actually computed by evaluating the average over several consecutive peak/valley pairs. From the expression for *TPR*, the associated logarithmic decrement can be evaluated using:

$$\delta = \ln(TPR) = -\zeta_{DR}\omega_{nDR}\left(\frac{\pi}{\omega_{dDR}}\right) \tag{22}$$

Finally, the damping of the dutch roll mode can be obtained using:

$$\zeta_{DR} = \frac{\left|\ln(TPR)\right|}{\sqrt{\pi^2 + \ln^2(TPR)}} \tag{23}$$

and the natural frequency of the dutch roll mode can be obtained using:

$$\omega_{nDR} = \frac{2\pi}{T_p\sqrt{1-\zeta_{DR}^2}} \tag{24}$$

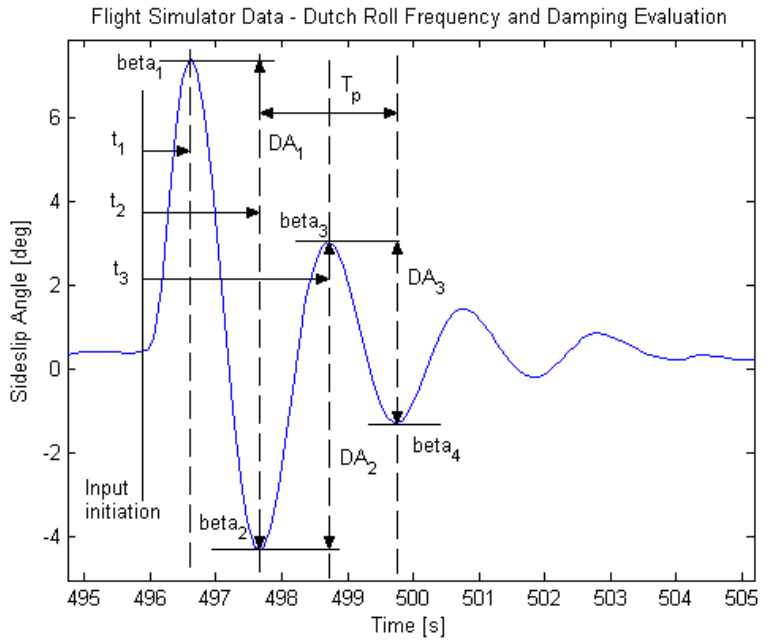


Figure 23. Parameters of the Peak-to-Valley Method to Determine the Characteristics of Slow Oscillatory Modes (Dutch Roll) [22]

6.3. Objective Tests

For the aerodynamic modeling, four flight conditions were necessary to pass the tests. These conditions were cruise, approach, climb, and steady state sideslip on approach. For the cruise condition the configuration of the aircraft was clean, i.e. landing gear up and flaps retracted to zero. For the approach condition the configuration of the aircraft was given as approach flaps at low altitudes. The climb condition had the same clean configuration of the

aircraft as the cruise condition. The steady state sideslip on approach is its own condition due to the fact that the aircraft has cross controlled inputs as the initial conditions. For all of the compliance tests done in this paper, the results were normalized due to the proprietary nature of the flight test data.

For the cruise condition there were five separate tests that the simulator had to comply with. The first test was the short period test. The requirements of the short period test require that the pitch angle must be within +/- 1.5 degrees of the flight data. The pitch rate must be within +/- 2 degrees per second of the flight data. The vertical acceleration must be within +/- .1 g of the flight data. As seen in Figure 24, the flight simulator conforms to the Federal Aviation Regulations Part 121.

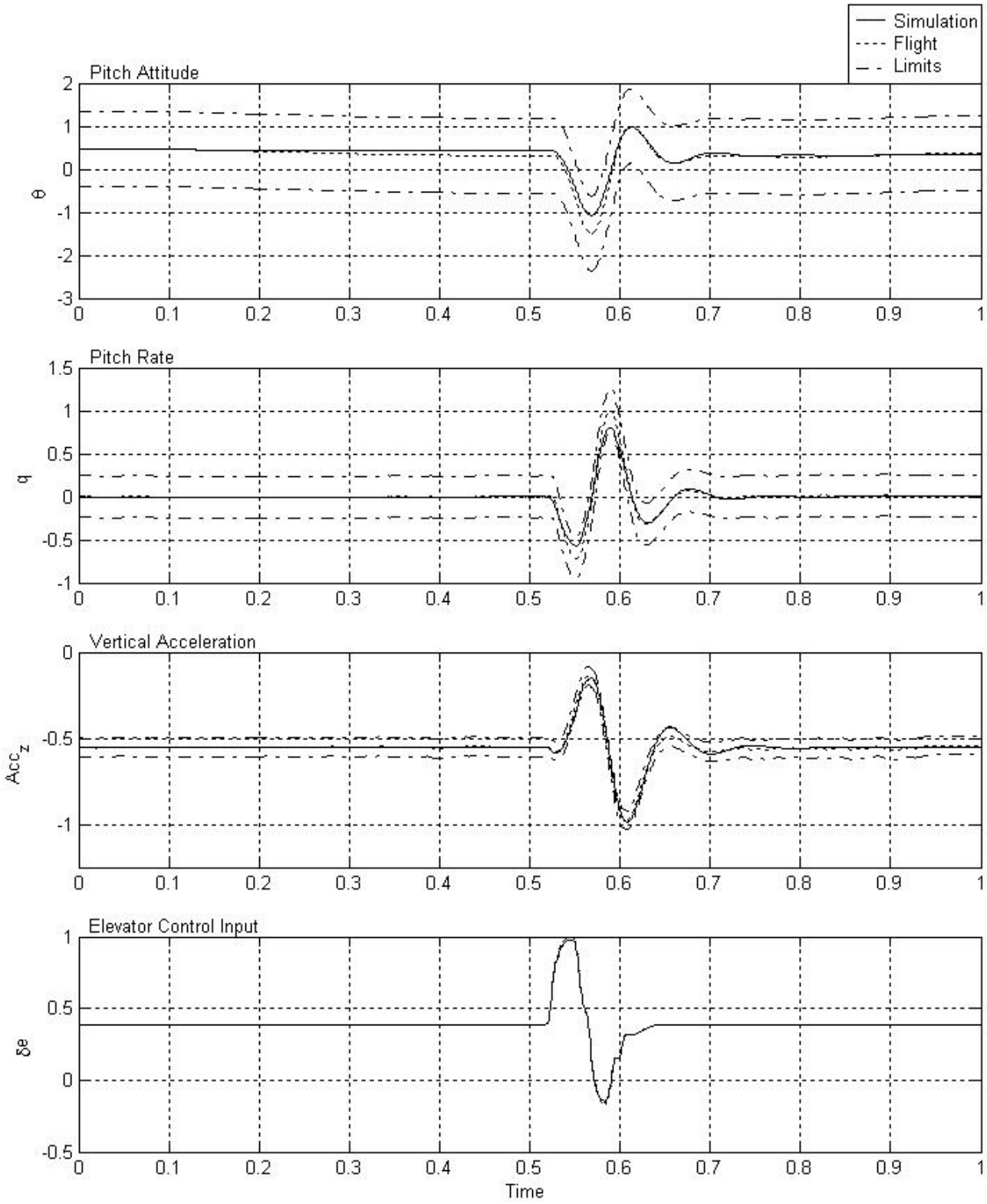


Figure 24. Normalized FAA Compliance Test 2.c.10. Short Period at Cruise Condition.

Figure 25 is the phugoid test at the cruise condition. The first requirement for the test is that the period be within +/- 10% of the flight data. The Peak-to-Valley method presented in section 6.2 was used to determine the phugoid modal parameters and the dutch roll modal parameters in later test requirements. The period cannot be directly seen in the figured do the normalization of the data, but does comply with the requirement. The next requirement was the damping be within plus/minus .02 of the flight data and does comply with the requirement.

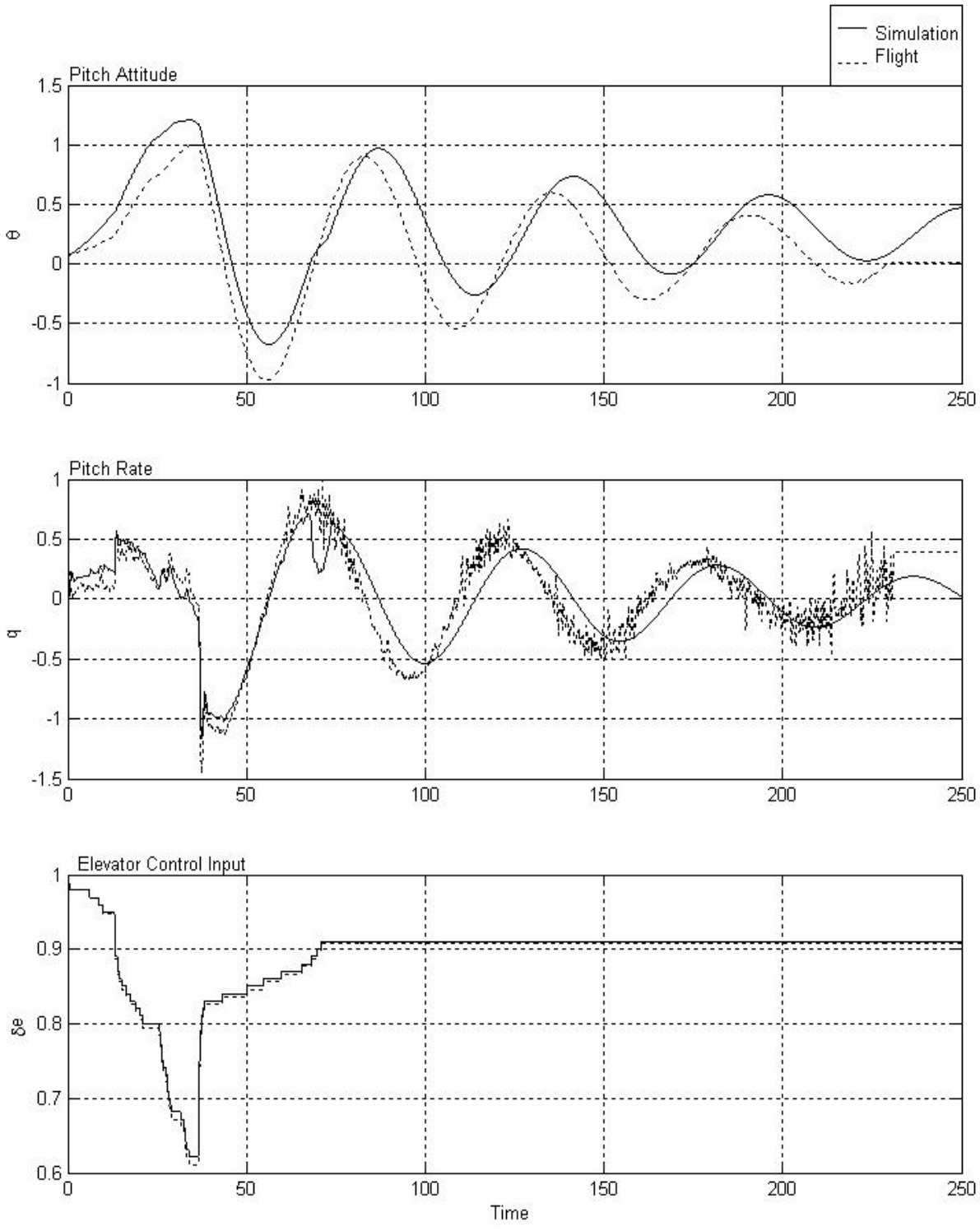


Figure 25. Normalized FAA Compliance Test 2.c.9. Phugoid at Cruise Condition.

Figure 26 shows the dutch roll test at the cruise condition. This test required a period of +/- 10% and a damping ratio of .02. The flight simulator does comply with both of the requirements for this test. The Peak-to-Valley method presented in section 6.2 was used to determine the dutch roll modal parameters.

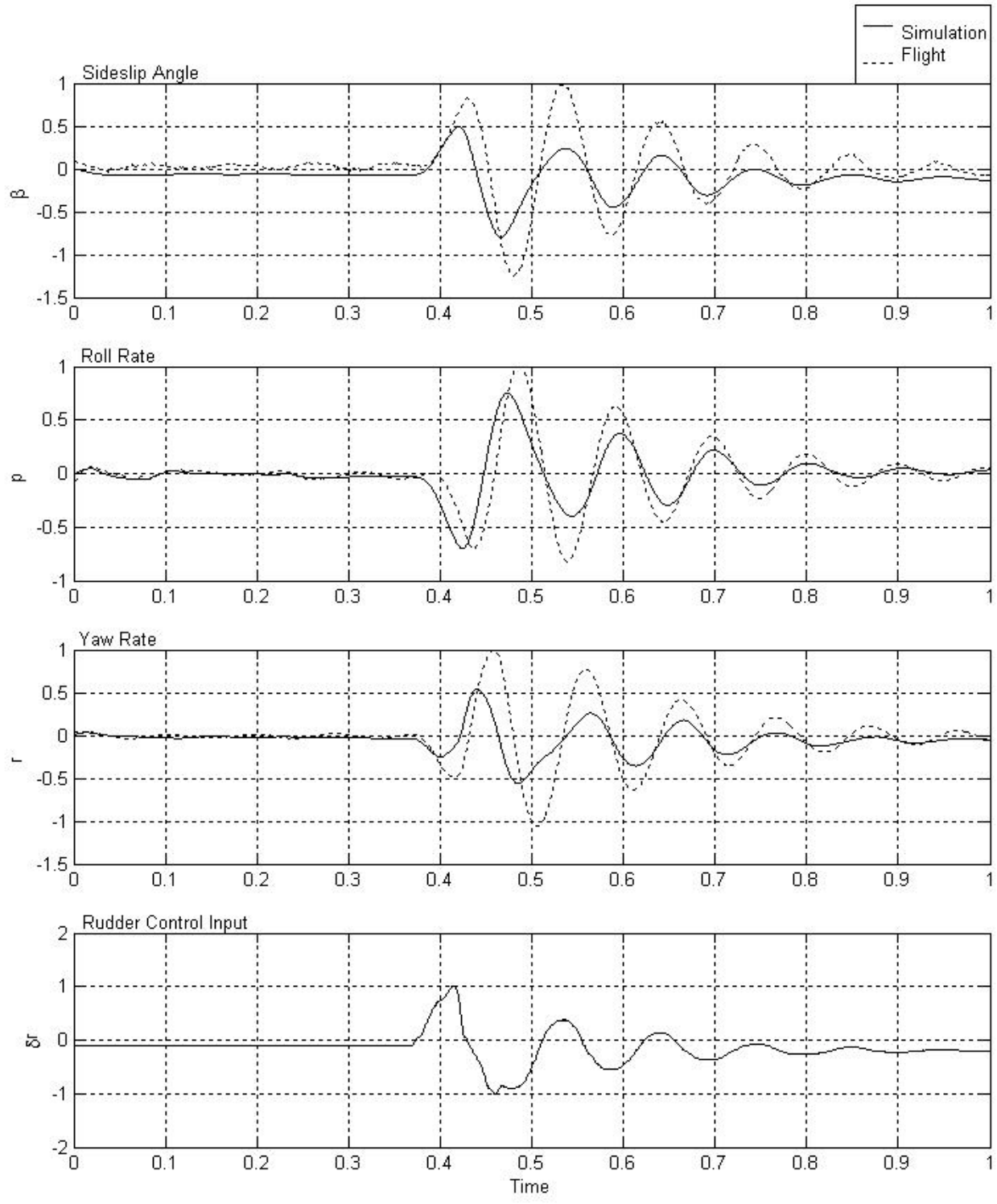


Figure 26. Normalized FAA Compliance Test 2.d.7. Dutch Roll At Cruise Condition.

Figure 27 is the roll response test at cruise condition. For this requirement, the simulator must be within +/- 10% of the flight data in the roll rate. Figure 27 shows the compliance with this test.

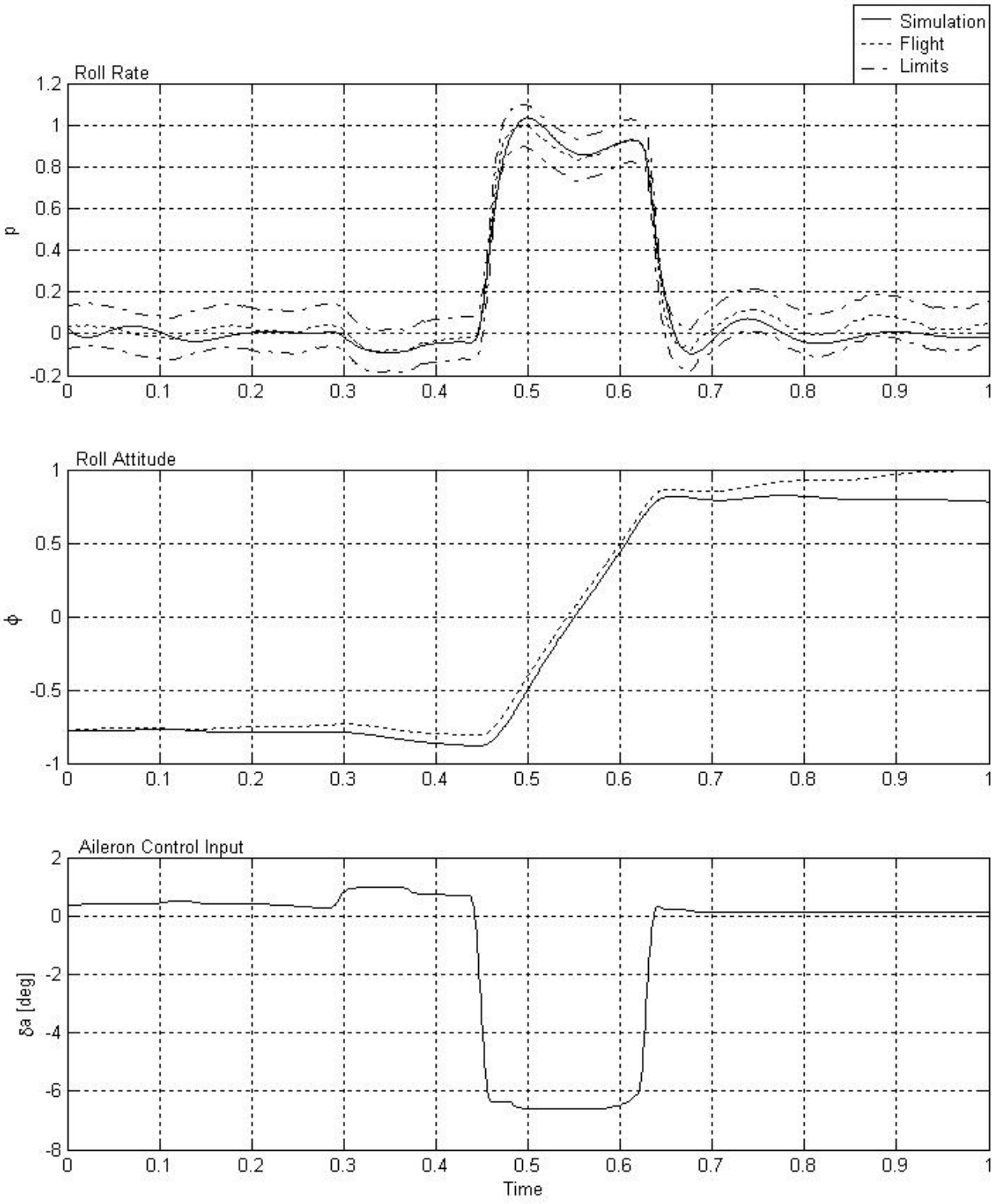


Figure 27. Normalized FAA Compliance Test 2.d.2. Roll Response Test at Cruise Condition.

Figure 28 is the FAA compliance test for spiral stability at cruise. For this test aircraft must show the correct trend in the bank angle, and the bank angle must be within +/- 10% in 30 seconds. As seen in the figure, the trend is the same as the flight data and complies with the +/- 10% requirement for the roll angle.

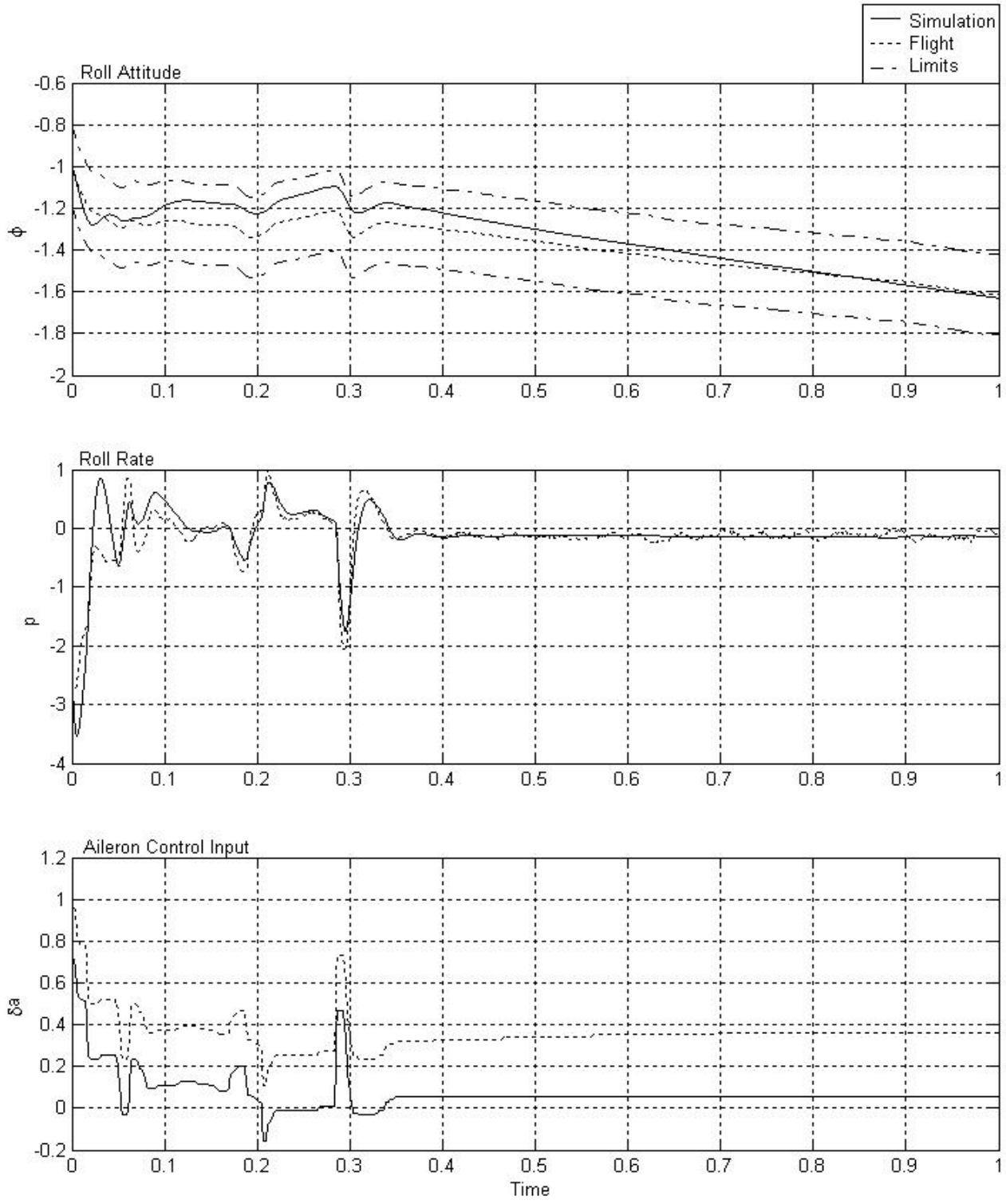


Figure 28. Normalized FAA Compliance Test 2.d.4. Spiral Stability at Cruise Condition.

The next flight condition was the approach condition. At this condition there are two tests. The first test is for the dutch roll on approach, in Figure 29. This test required a period of +/- 10% and a damping ratio of .02. Also note that the aileron deflections are not the same. The reason for this is a bias in the sensors of this flight test. The FAA allows the deflection of the control surfaces to be up to a few degrees different because of these biases. Our simulator aileron deflections are well within the bounds of the bias.

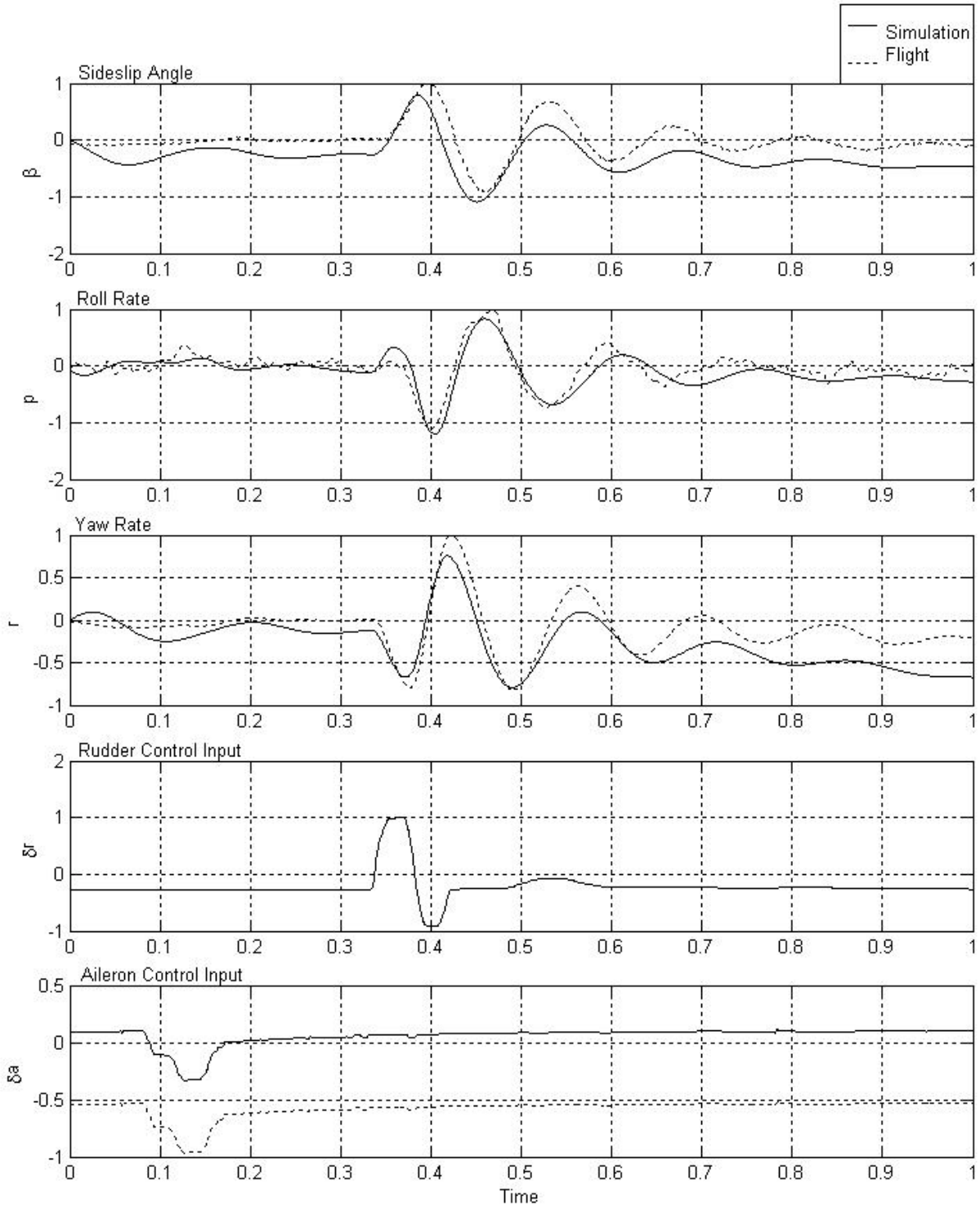


Figure 29. Normalized FAA Compliance Test 2.d.7. Dutch Roll at Approach Condition.

Figure 30 shows the roll response test at the approach condition. The requirements for this test are the same as the requirements for the roll response test at the cruise condition plus the roll angle has to be within +/- 2 degrees of the flight data within 10 seconds of neutral inputs. This test does comply with the FAA regulations but at the end of the test the roll angle slightly comes out of the bounds of the test. This still complies with the test because the requirements were for the bounds of the roll angle for +/- 2 degrees in 10 seconds, which the simulator achieves.

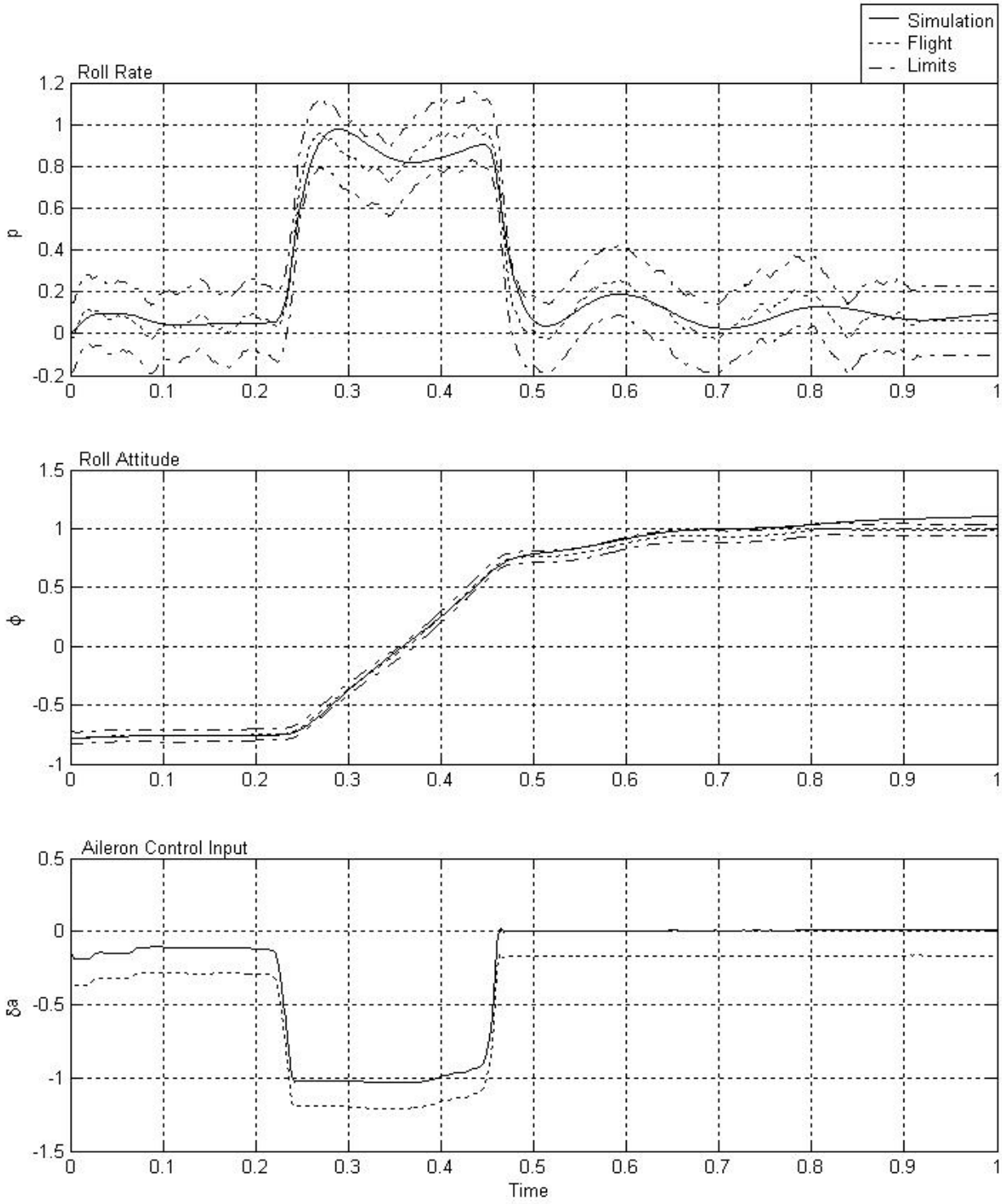


Figure 30. Normalized FAA Compliance Test 2.d.2 and 2.d.3. Roll Response at Approach Condition.

The next flight condition is the climb condition. In this condition there is only one test required. This test requires that the airspeed be within +/- 3 knots airspeed of the flight data and the rate of climb be within +/- 100 feet per minute for 1000 feet gained in altitude. Figure 32 is the climb test and at first glance does not pass the airspeed requirement of the FAA compliance test. It does pass the test, although there is a violation of the bounds at the end of the simulation, because the requirement is for 1000 feet of climb and that requirement was met before the simulator left the bounds of the flight data.

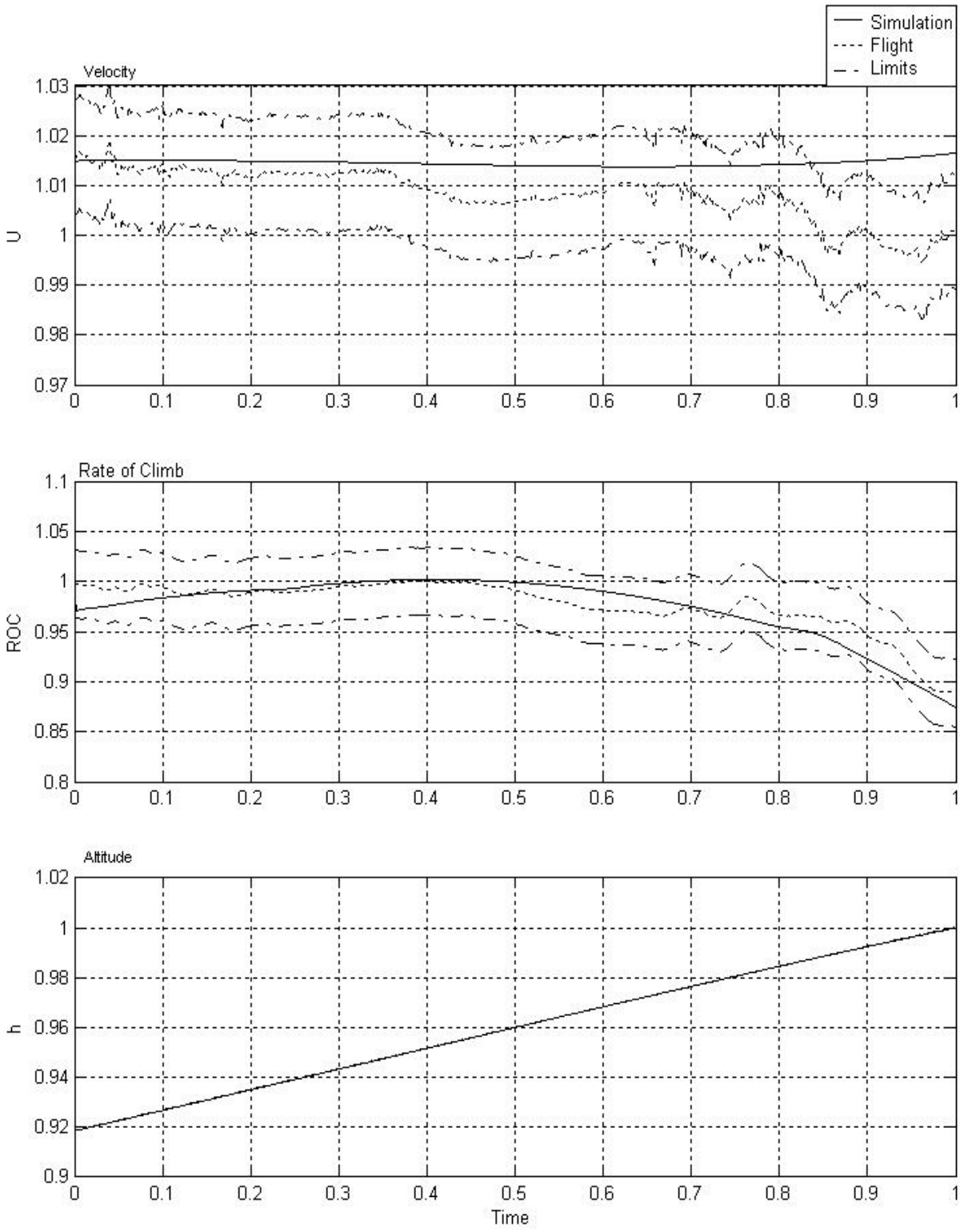


Figure 31. Normalized FAA Compliance Test 1.c.1. Normal Climb requirements.

The next test requirement is for the steady state sideslip on approach condition. For this test requirements are: roll angle must be within +/- 2 degrees of flight data; sideslip angle must be within +/- 1 degree of flight data; aileron deflection must be within +/- 10% of the flight data. As seen in Figure 32, the simulator matches the aircraft very well in the steady state sideslip condition.

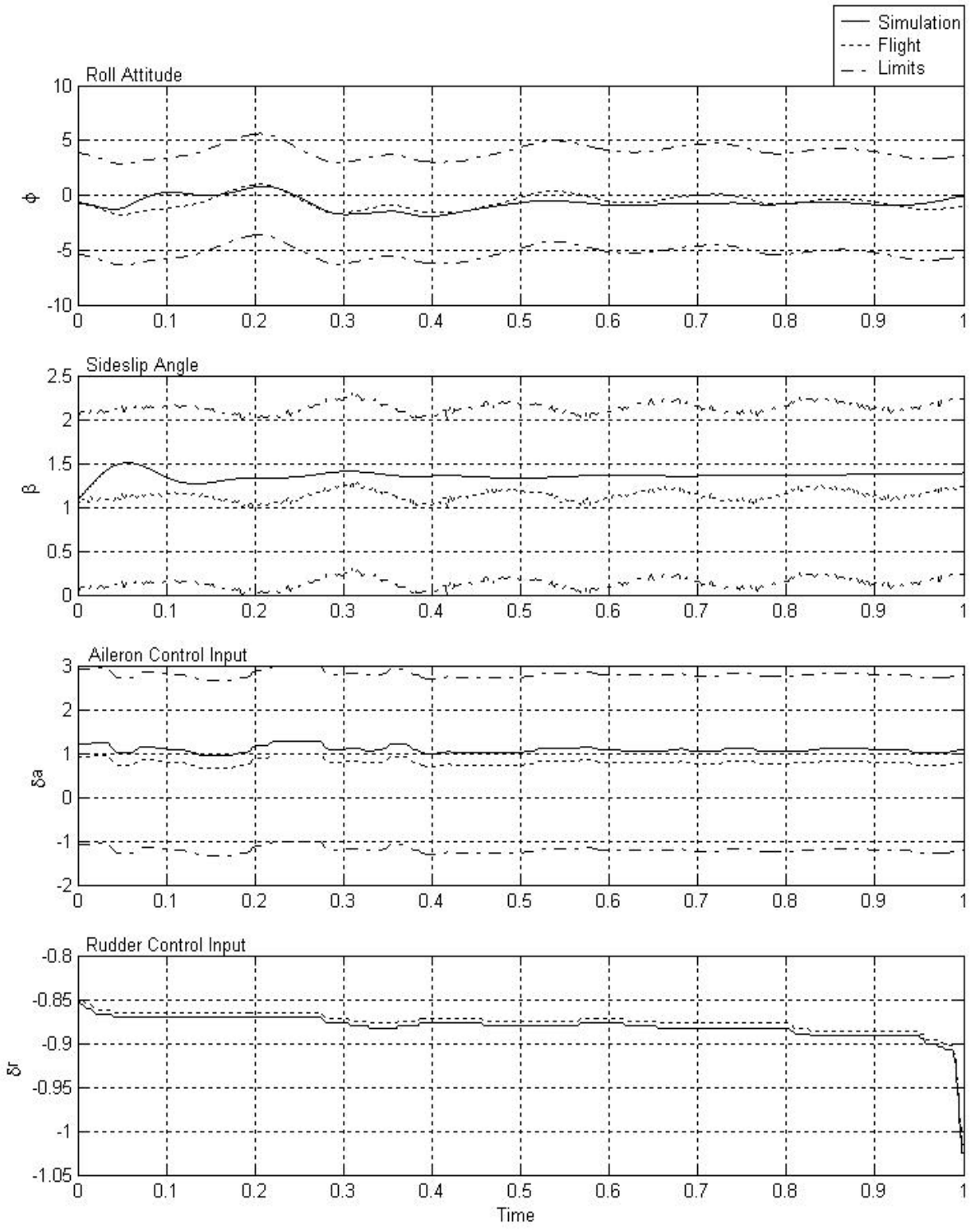


Figure 32. Normalized FAA Compliance Test 2.d.8. Steady State Sideslip at Approach Condition.

The last tests are the same test but at different flight conditions. The test is for the longitudinal trim. In this test, the elevator must be within +/- 1 degree of the flight, the pitch angle must be within +/- 1 degrees and the thrust must be within +/- 5% of the flight data. For the engine, the N1 variable of the engine directly correlates with the thrust of the engine and is therefore the variable compared in the flight data. The Figures 33, 34, and 35 are the longitudinal trim on cruise, approach, and landing respectively. In all three flight conditions, the simulation conforms to the requirements thus passing the test.

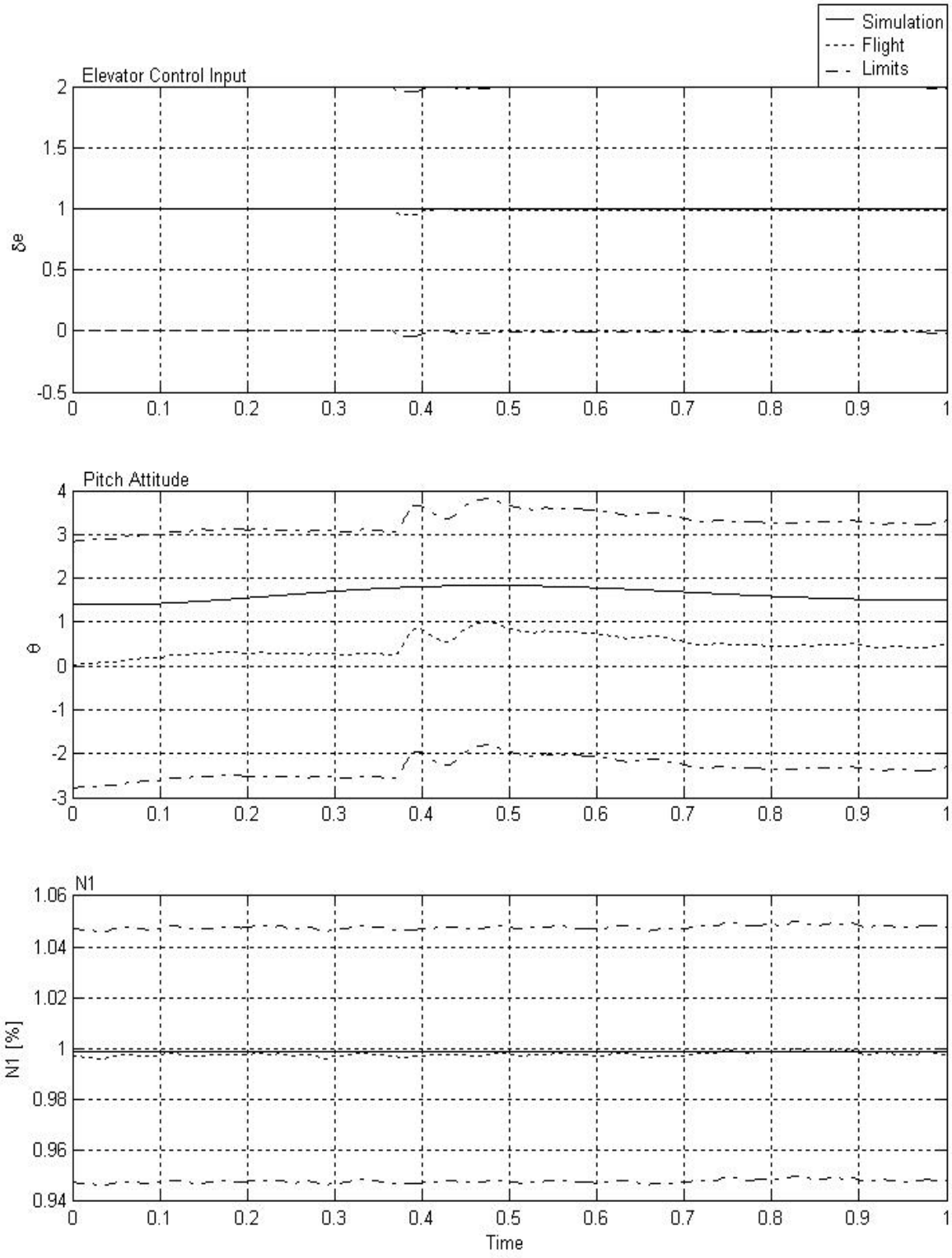


Figure 33. Normalized FAA Compliance Test 2.c.5. Longitudinal Trim at Cruise.

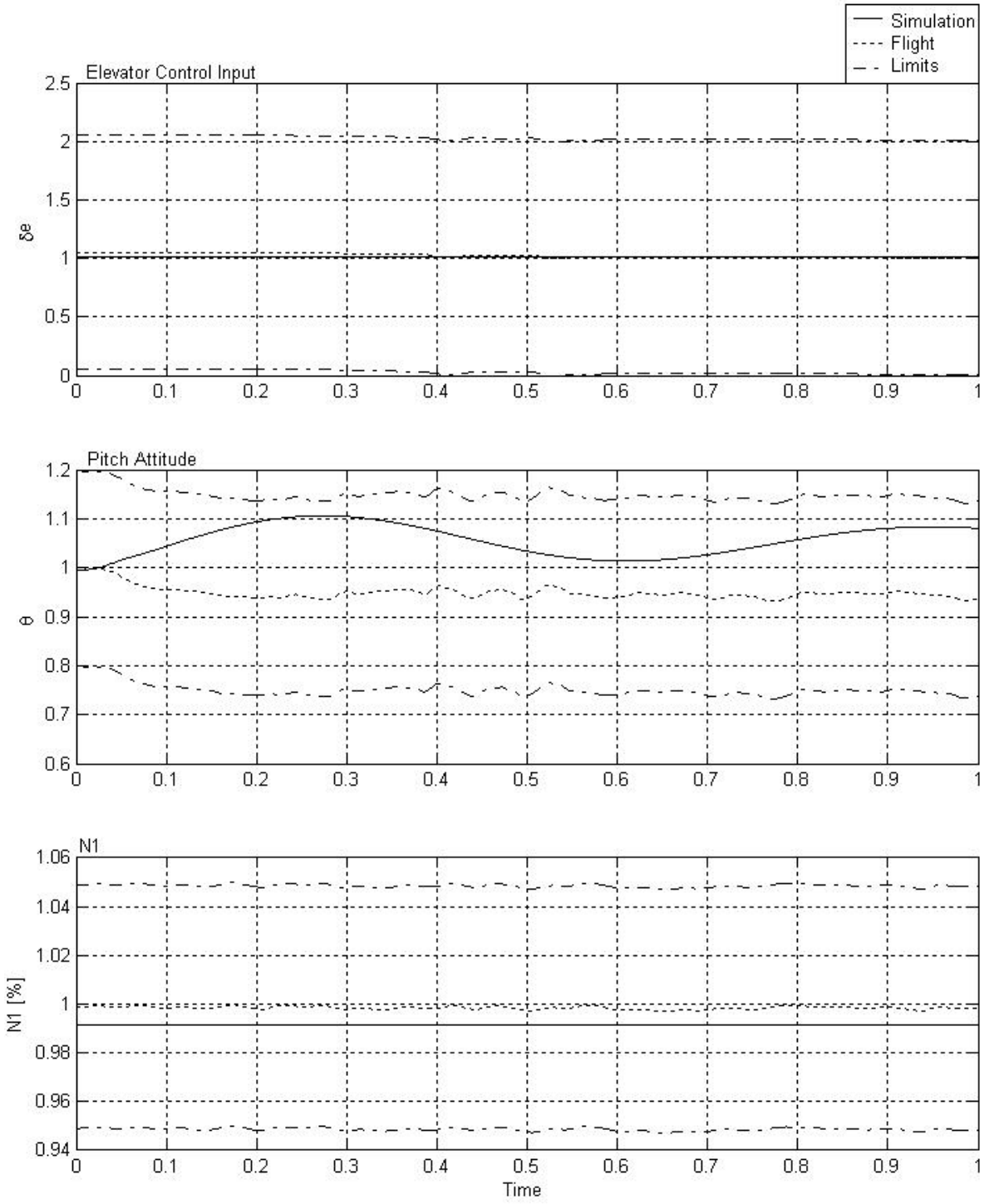


Figure 34. Normalized FAA Compliance Test 2.c.5. Longitudinal Trim at Approach.

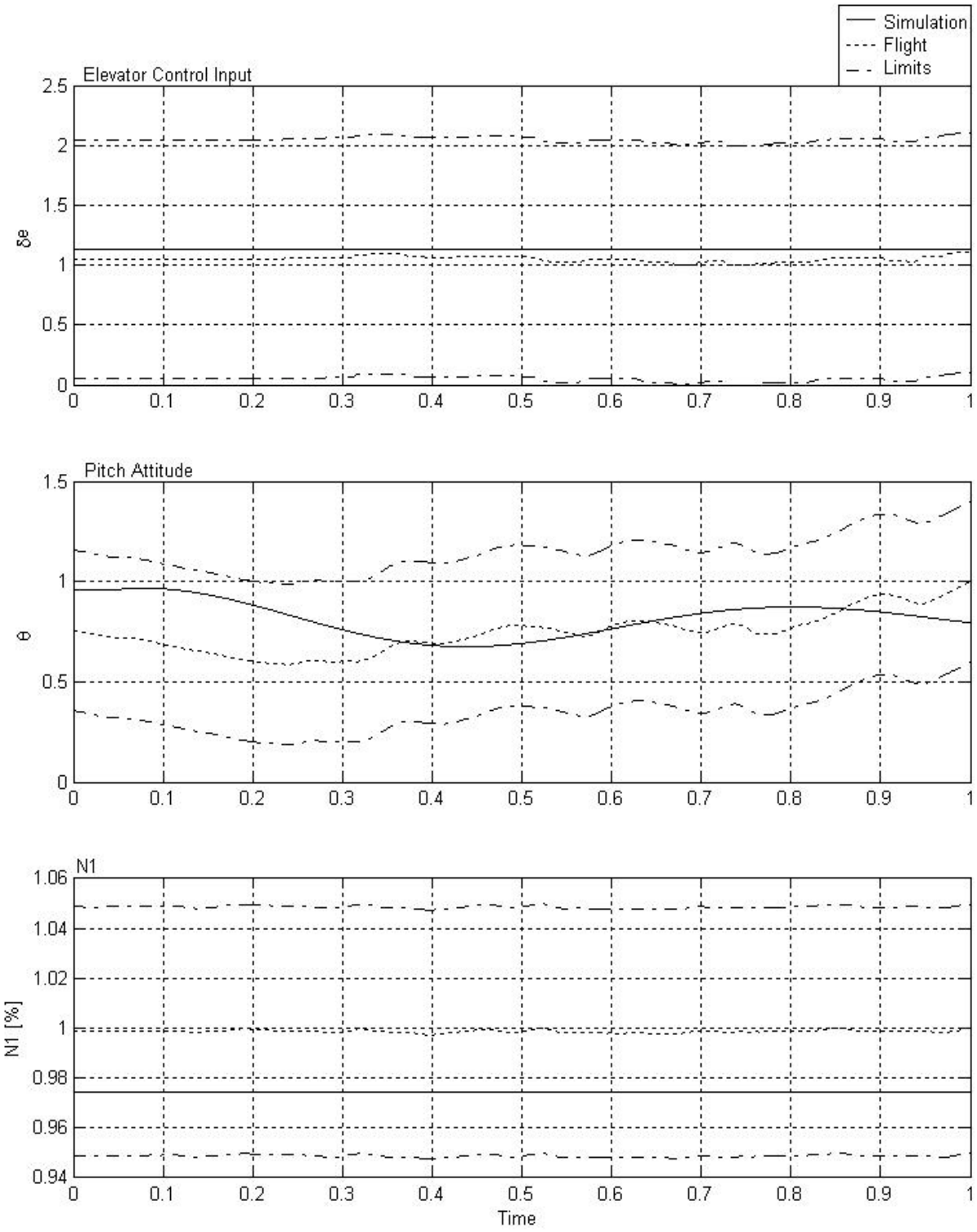


Figure 35. Normalized FAA Compliance Test 2.c.5. Longitudinal Trim at Landing.

To show how sensitive the response is to the alteration of the stability or control derivatives, the stability derivative C_{lp} was decreased by 3% and the FAA compliance test 2.d.2 and 2.d.3 were then attempted again with the new C_{lp} value. In Figure 36, the roll rate does fail the test at about .45 and .65 of the normalized time. It is interesting to see how the stability derivatives affect the response of the aircraft model. The small variation in stability derivative here shows how stringent the FAA compliance tests are. The FAA objective tests are an excellent standard in developing very accurate models for flight training devices.

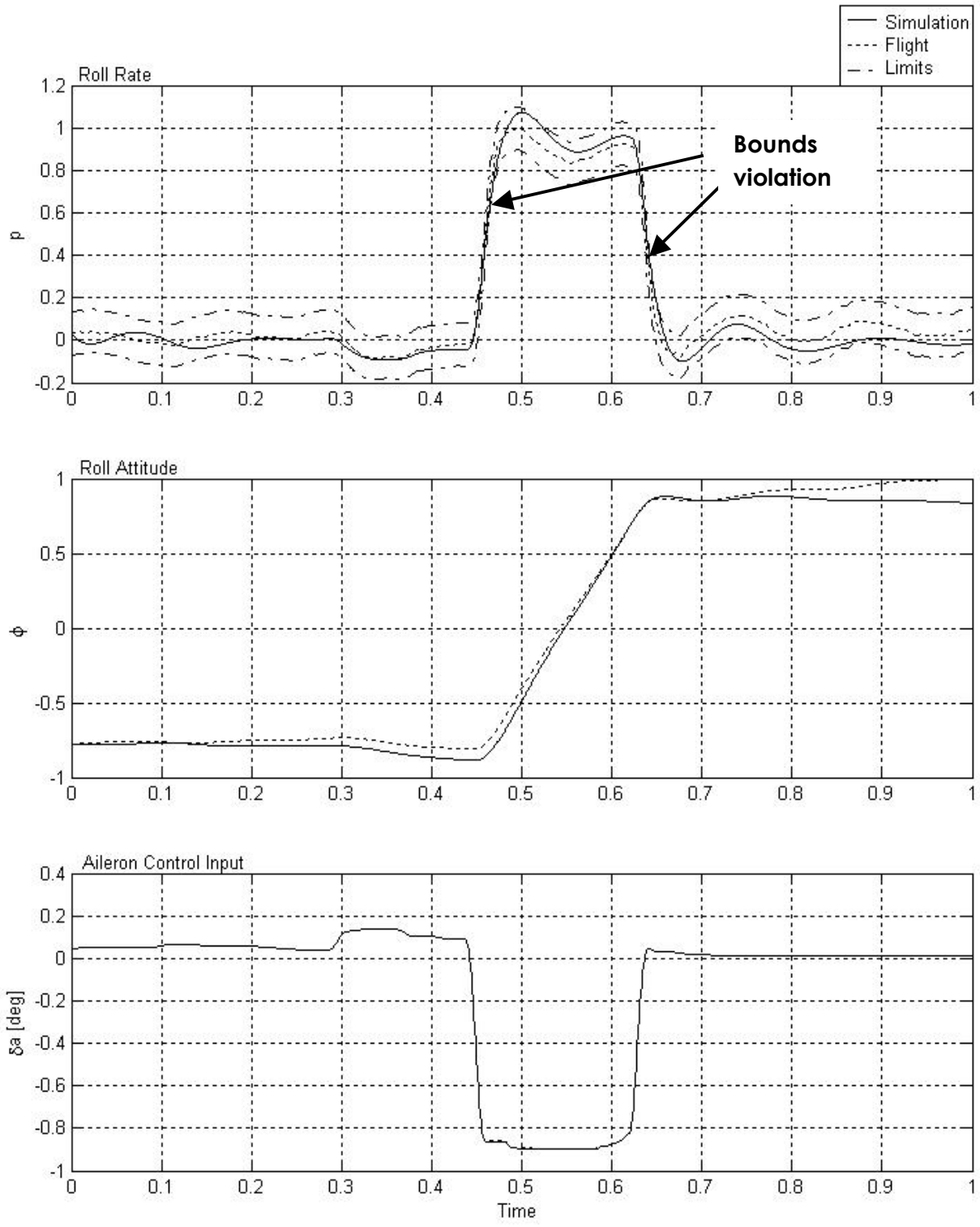


Figure 36. Normalized FAA Compliance Test 2.d.2 and 2.d.3. Roll Response at Cruise Condition with Altered C_{lp}.

6.4. Subjective Testing

Another requirement is that the flight simulator be able to be flown by hardware control inputs and give the same reaction the pilots would expect from the aircraft. For an example of this, a simulation was done with inputs from a joystick. There was a bias in the joystick on the rudder input of -1 degrees and causes the steady state to occur later in the simulation. After the aircraft reached steady state, a maximum deflection rudder doublet was done to excite the dutch roll and the response was plotted in Figure 37.

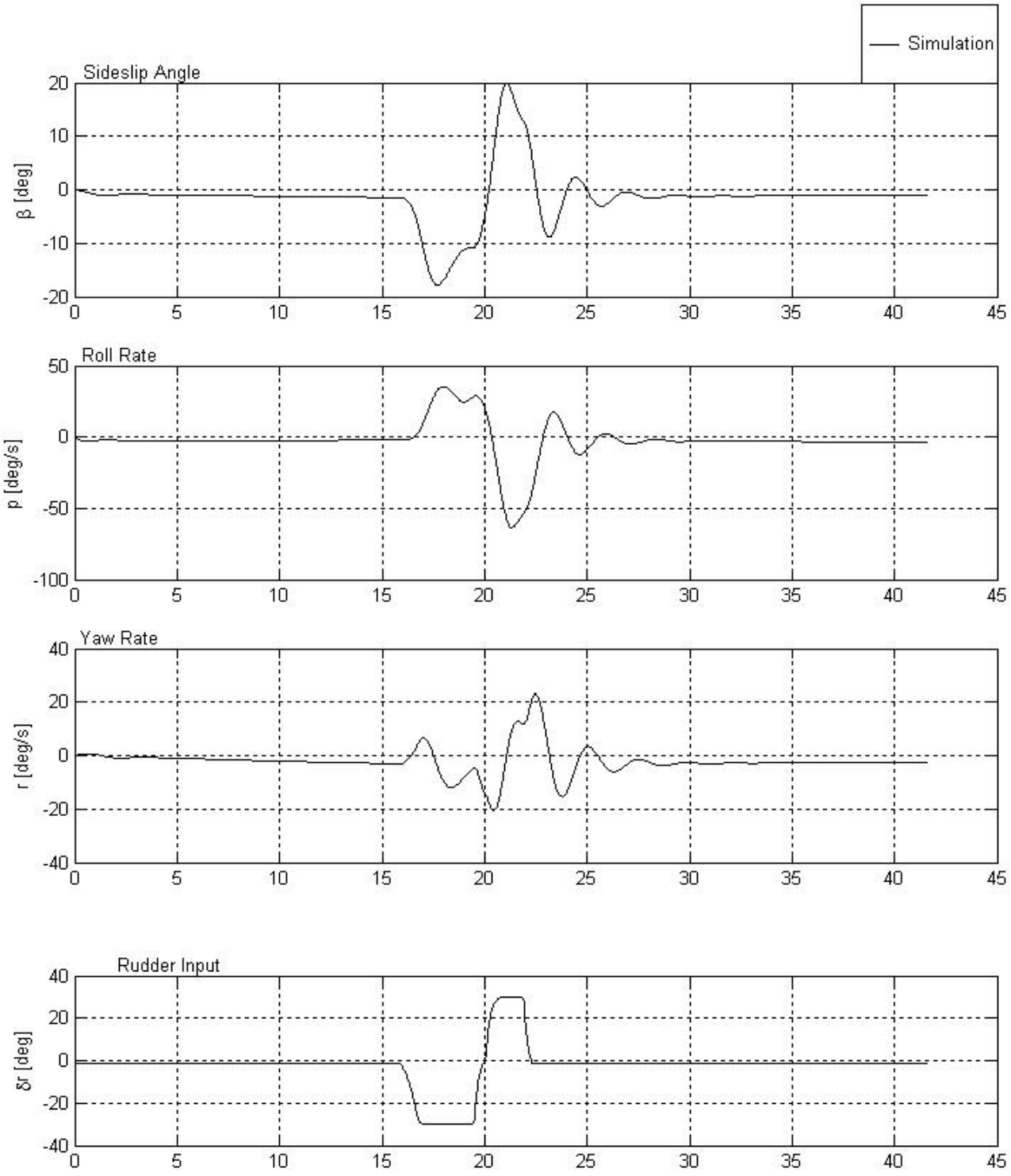


Figure 37. Piloted Input.

An issue that arose in subjective testing would have never been found in the FAA objective tests. The issue was for steady state sideslip. When the pilot would input a large positive rudder deflection, the heading angle should decrease (the sign convention is that yaw angle and rate are negative when the nose of the aircraft moves to the left). In this case, the aircraft would start with a negative heading change and then have a positive heading change which was intuitively wrong. Looking at the stability derivatives the first derivative that came to mind that could possibly affect the heading due to rudder input was $C_{n\delta r}$. Changing $C_{n\delta r}$ by about -5% had very little effect on the heading angle. $C_{n\delta r}$ was then changed by -20% and it still had a mediocre effect on the heading angle issue, but then the aircraft model failed the FAA requirements. $C_{n\delta r}$ was changed back to its original value and then another approach was taken. A steady state sideslip was then attempted. With a steady state sideslip, a large rudder deflection is countered with an aileron deflection to keep a steady heading angle. If the yaw rate was negative, therefore a decreasing heading angle, it would have to be countered with a positive roll angle to keep a steady heading. A piloted test was done and there was a problem. To keep a steady state sideslip the aircraft had to have a negative roll angle with a negative yaw rate which was wrong. This led to looking at the lateral force coefficient. The problem was that the $C_{y\delta r}$ was too large causing that term to be larger than $C_{y\beta}$, thus causing the steady state sideslip roll angle to be negative instead of positive. To solve this problem, pilot experience gave us the information for the steady state roll angle. Once the roll angle was known, the lateral force equation was set to zero and solved for this desired roll angle and sideslip angle to get the desired $C_{y\delta r}$. The pilot then tested this maneuver again and was satisfied with the new $C_{y\delta r}$. After finding the new $C_{y\delta r}$, the FAA objective tests were affected only minimally and still passed.

After the subjective and objective tests were completed, the model was implemented on the simulator hardware and will go through the FAA certification process. To help speed this process up, the objective tests were plotted as they were in the figures above and any requirements that required a % difference or difference, mainly the damping and natural frequency constants, will be put into a table.

7. Conclusions

The strategy developed for building an FAA approved Level 6 Flight Training Device software was an overall success. With this strategy, similar Level 6 flight simulators can be created. Using the computational tool to analyze the flight test data to find the best data points for the tests and parameter identification greatly reduced the time it would have taken to look at the data individually for every file. The computational tool was also a very effective tool to take the desired data and processing it into a format to be used with the simulation program of Matlab and Simulink. The parameter identification techniques are a very efficient way to find the stability and control derivatives. However, it should be noted that the type of excitation present in the flight data and the specific parameters that need to be determined are correlated for successful PID. Tweaking the model though, requires an understanding of flight dynamics and how the stability derivatives affect the response of the aircraft.

References

1. Page, Ray L. "Brief History of Flight Simulation."
2. Perkins C. D.; Hage R. E. "Airplane Performance, Stability, and Control," Wiley, New York, 1949
3. Etkin B. "Dynamics of Flight, Stability, and Control," Wiley, New York, 1959
4. Miele A. "Flight Mechanics: Theory of Flight Paths," Addison-Wesley Publishing Co., 1962
5. Seckel E. "Stability and Control of Airplanes and Helicopters," Academic Press, 1964
6. McCormick B. "Aerodynamics, Aeronautics, and Flight Mechanics," J. Wiley & Sons, 1979
7. Babister A. "Aircraft Dynamic Stability and Response," Pergamon Press, 1980.
8. Etkin, B., and Reid, L., *Dynamics of Flight: Stability and Control*, J. Wiley & Sons, 1996
9. Zipfel P. H. "Modeling and Simulation of Aerospace Vehicle Dynamics," AIAA Education Series, AIAA, Inc. Reston, Virginia, 2000
10. Stevens B. L. Lewis F. L., "Aircraft Control and Simulation," John Wiley and Sons , Inc., Second Edition, Honoken, NJ, 2003
11. Tewari A. "Atmospheric and Space Flight Dynamics – Modeling and Simulation with MATLAB and Simulink," Birkhaeuser, Boston, 2007
12. Dreier M. E. "Introduction to Helicopter and Tiltrotor Flight Simulation," AIAA Education Series, AIAA, Inc. Reston, Virginia, 2007
13. Roskam J. "Airplane Flight Dynamics and Automatic Flight Controls," Roskam Aviation and Engineering Corporation, Ottawa, KS, 1982
14. Klein, V. and Morelli, E.A., "Aircraft System Identification – Theory and Practice," AIAA Education Series, AIAA, Reston, VA, 2006.

15. Debusk, W.; Chowdhary, G.; Johnson , E. N. "Real-Time System Identification of a Small Multi-Engine Aircraft," Georgia Institute of Technology, Atlanta, GA, 2009
16. Morelli, E. A., "Real Time Parameter Estimation in the Frequency Domain," AIAA Journal of Guidance, Control and Dynamics, Vol.23, no.5.
17. United States Advisory Circular 120-40B, *Airplane Simulator Qualification*.
18. United States Advisory Circular 120-45A, *Airplane Flight Training Device Qualification*.
19. Yongkyu, S., Campa G., Napolitano M.R., Seanor B., Perhinschi, M.G. "Comparison of On-Line Parameter Estimation Techniques Within a Fault Tolerant Flight Control System," AIAA Journal of Guidance, Control and Dynamics, May 2001
20. ***, "*Aviator Visual Design Simulator (AVDS) - User Manual*," Rassmussen Simulation Technologies, Ltd., Oct. 2000.
21. Meyer A. Van Kampen H., "X-Plane On-Line Instruction Manual," Laminar Research Inc., Columbia SC, 8th edition, 2002 (www.X-Plane.com).
22. Sagoo G. K., Gururajan S., Napolitano M. R., Perhinschi M. G., Gu Y., Seanor B., Campa G., "Pilot-in-the-Loop Assessment of Neurally Augmented Dynamic Inversion Based Fault Tolerant Control Laws in a Motion-Based Flight Simulator," *Proceedings of the AIAA Modeling and Simulation Technologies Conference,* August 2008, Honolulu, HW
23. Perhinschi M. G., Napolitano M.R. "Integration of Computer Simulation for Flight Dynamics and Control Education," *Computers in Education Journal*, vol. XVIII, no. 1, pp 13-22, Jan.-Mar. 2008

Appendix A – Altered Flight Test Data Example

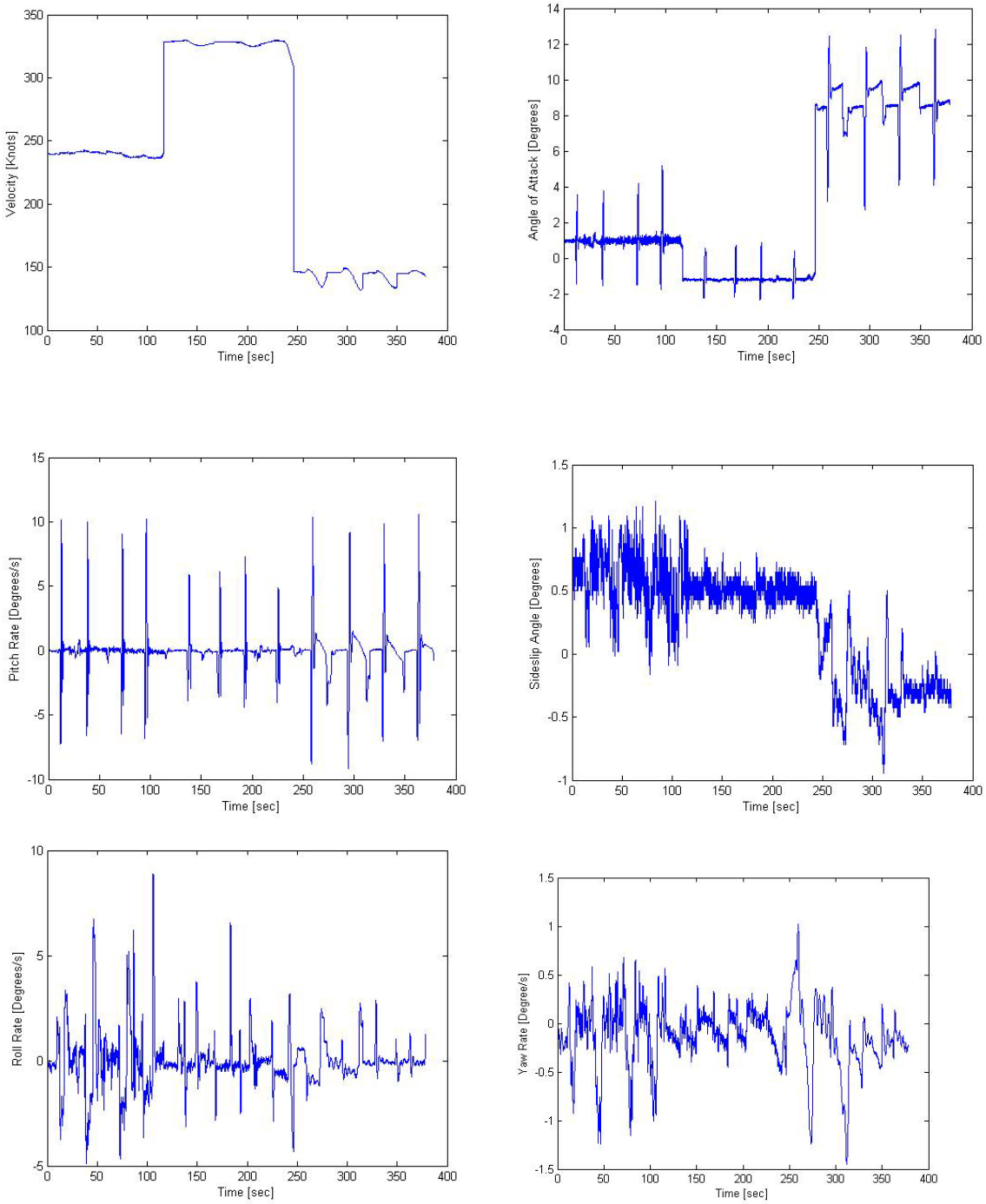


Figure A1. State Variables of Altered Flight Data for a Short Period Test

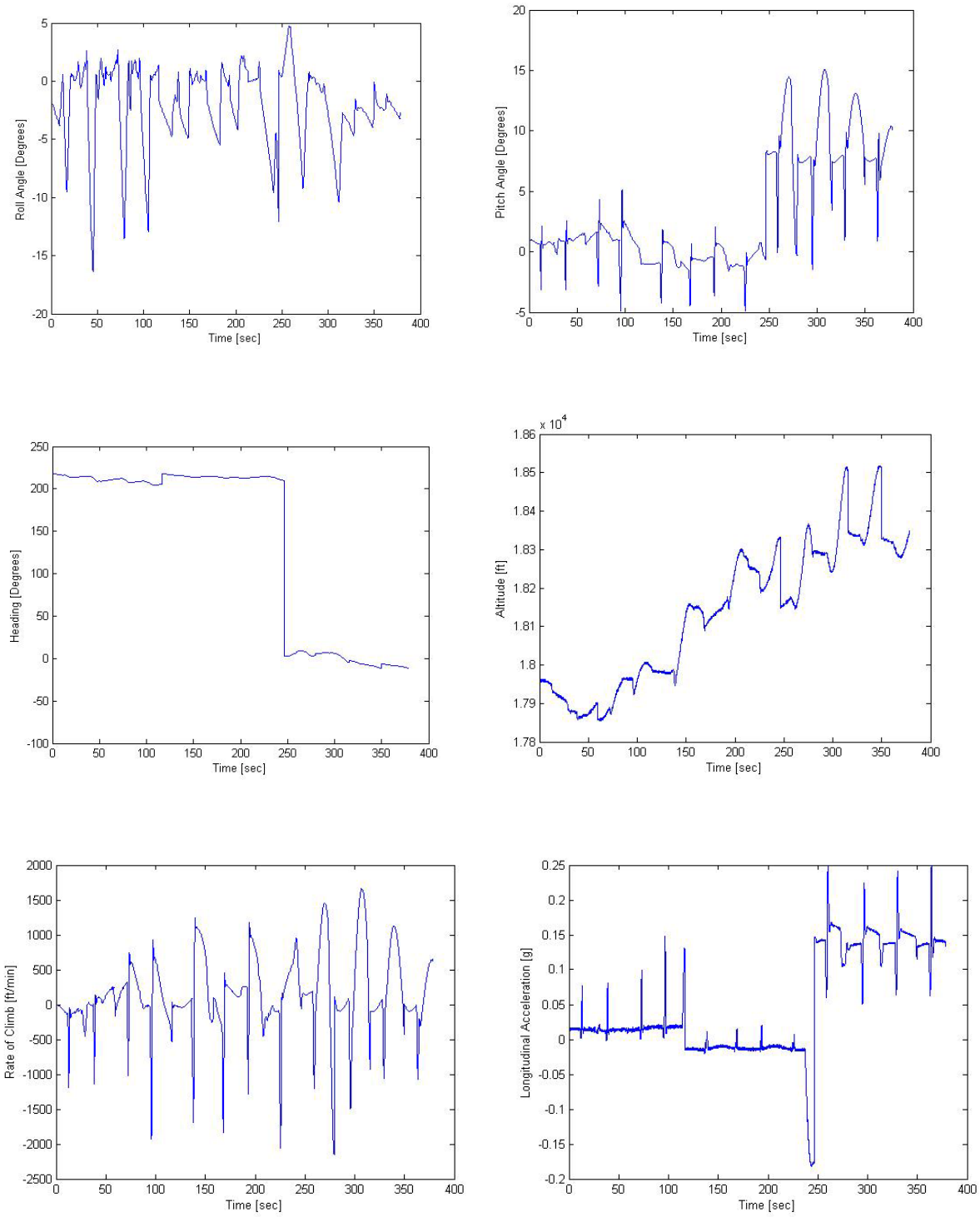


Figure A2. State Variables of Altered Flight Data for a Short Period Test.

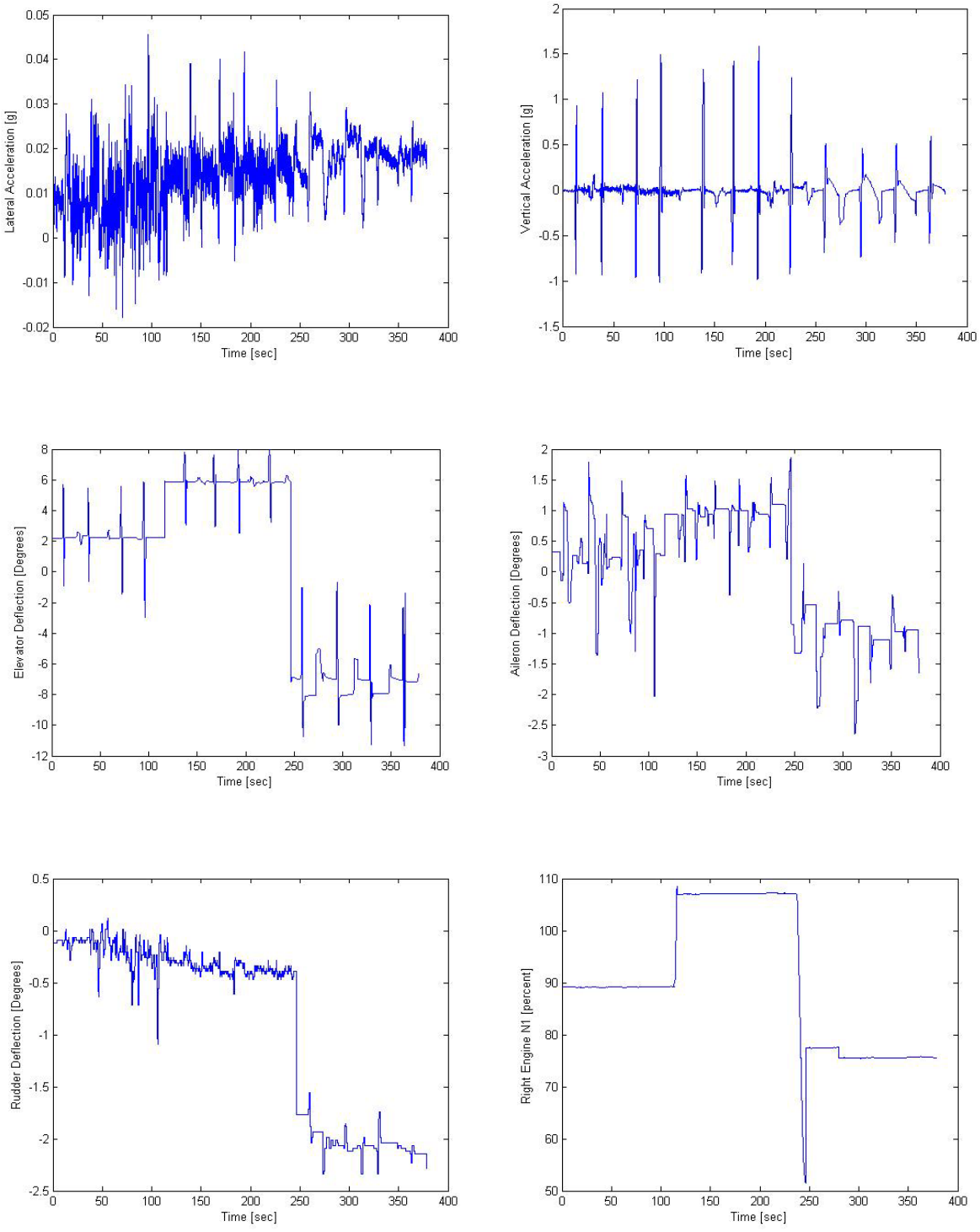


Figure A3. State and Control Variables of Altered Flight Data for a Short Period Test.

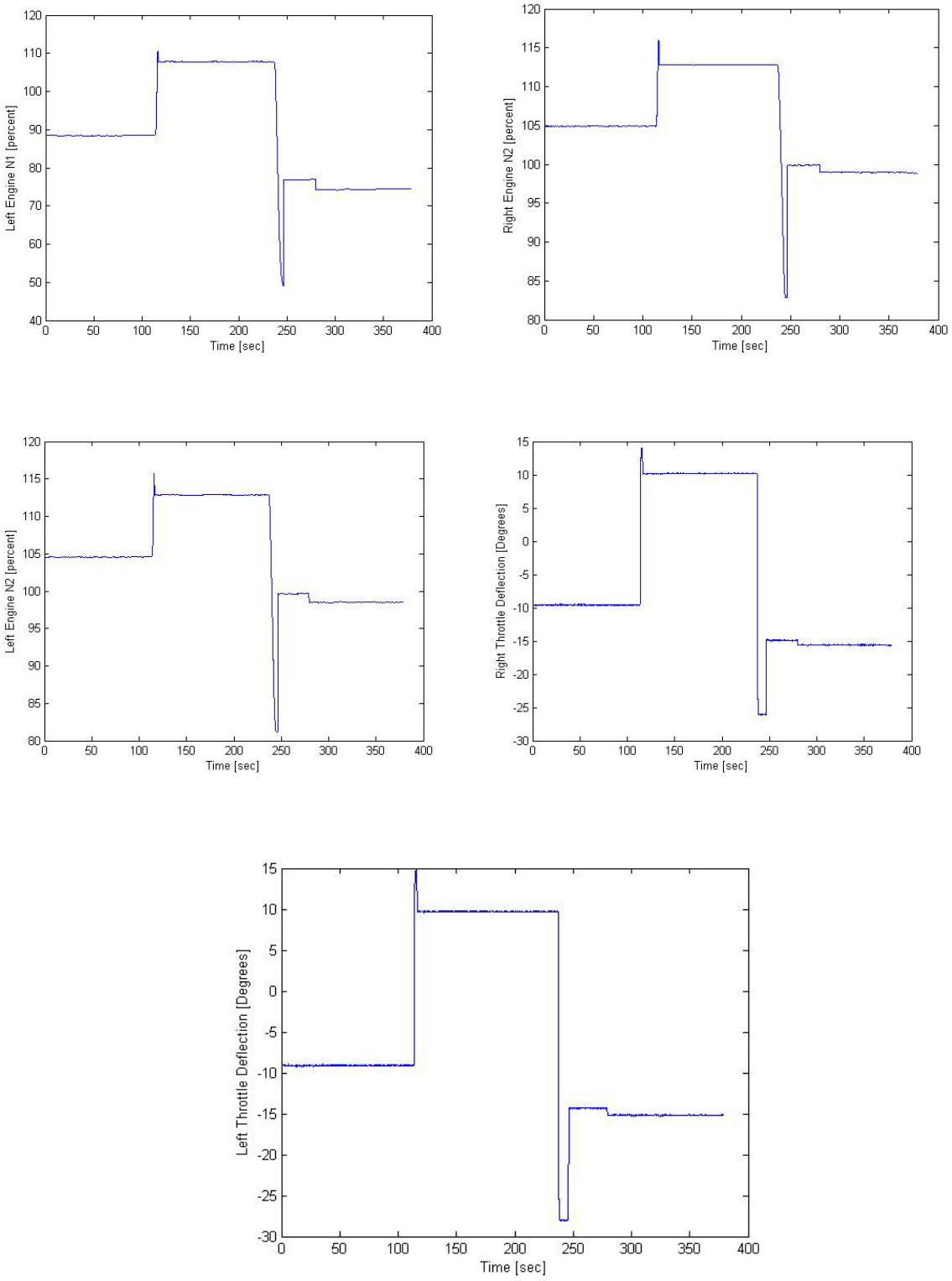


Figure A4. Control and Engine Variables of Altered Flight Data for a Short Period Test.

Appendix B – Altered Flight Test Data Segments Example

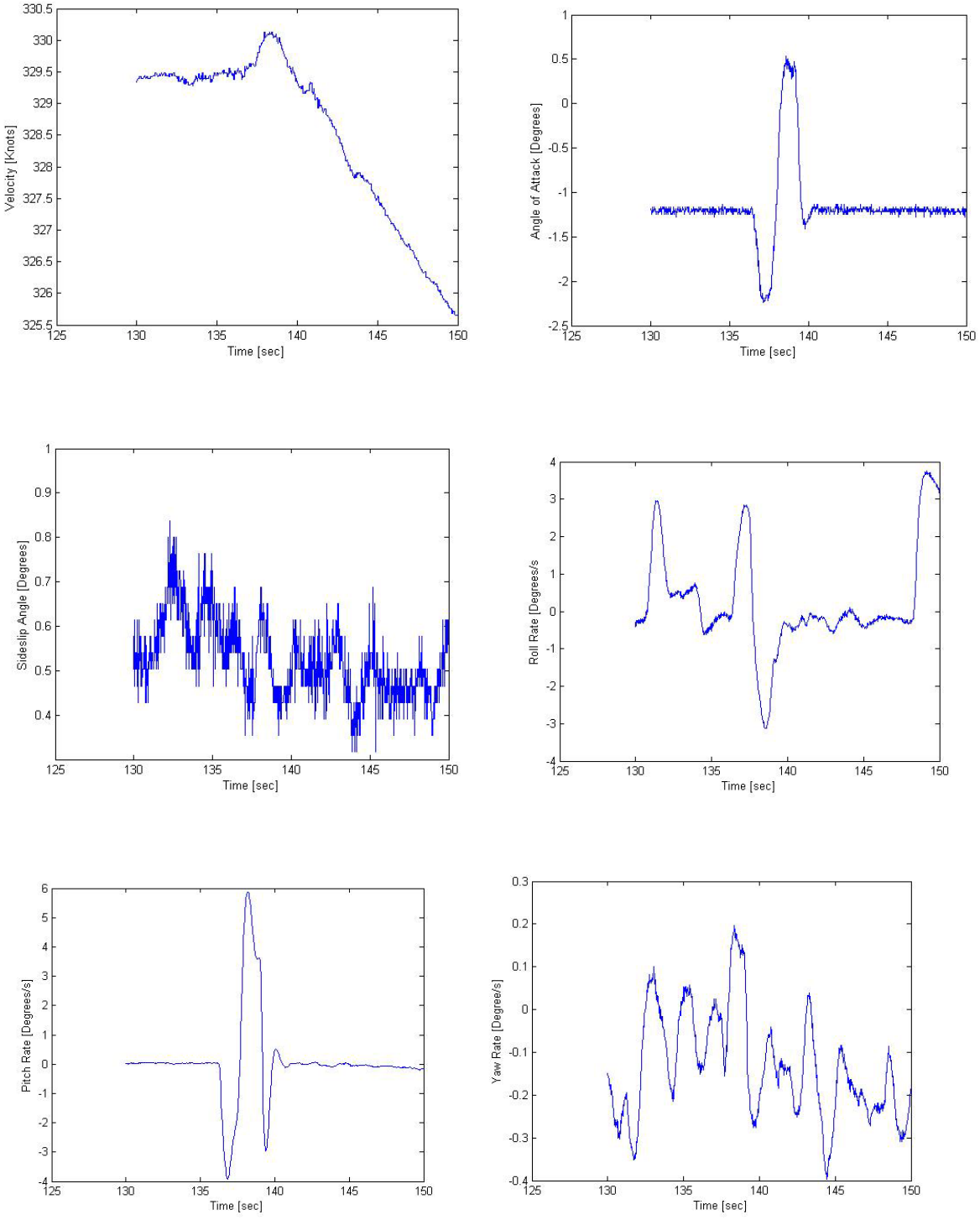


Figure B1. State Variables of Altered Flight Data Segment for a Short Period Test.

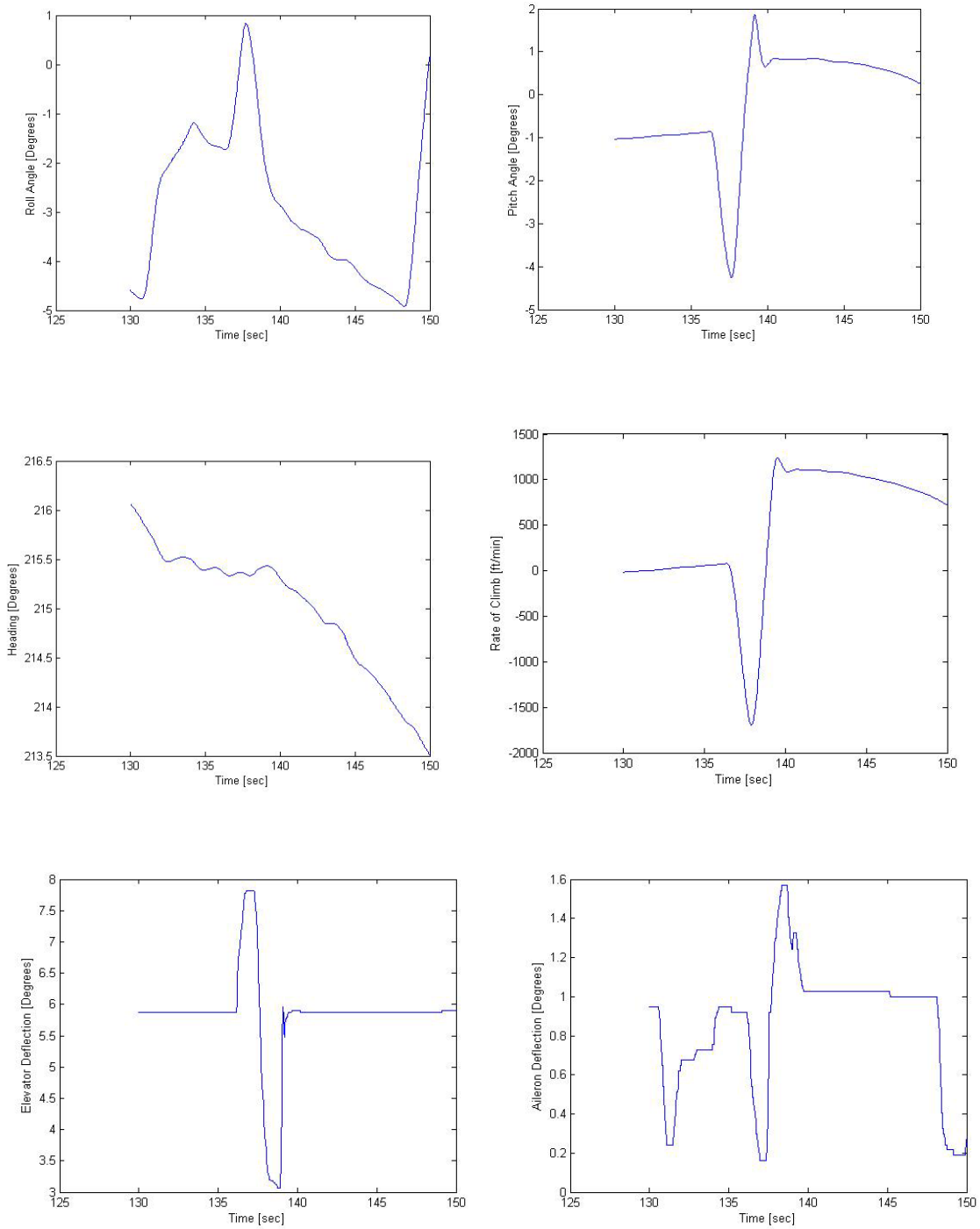


Figure B2. State and Control Variables of Altered Flight Data Segment for a Short Period Test.

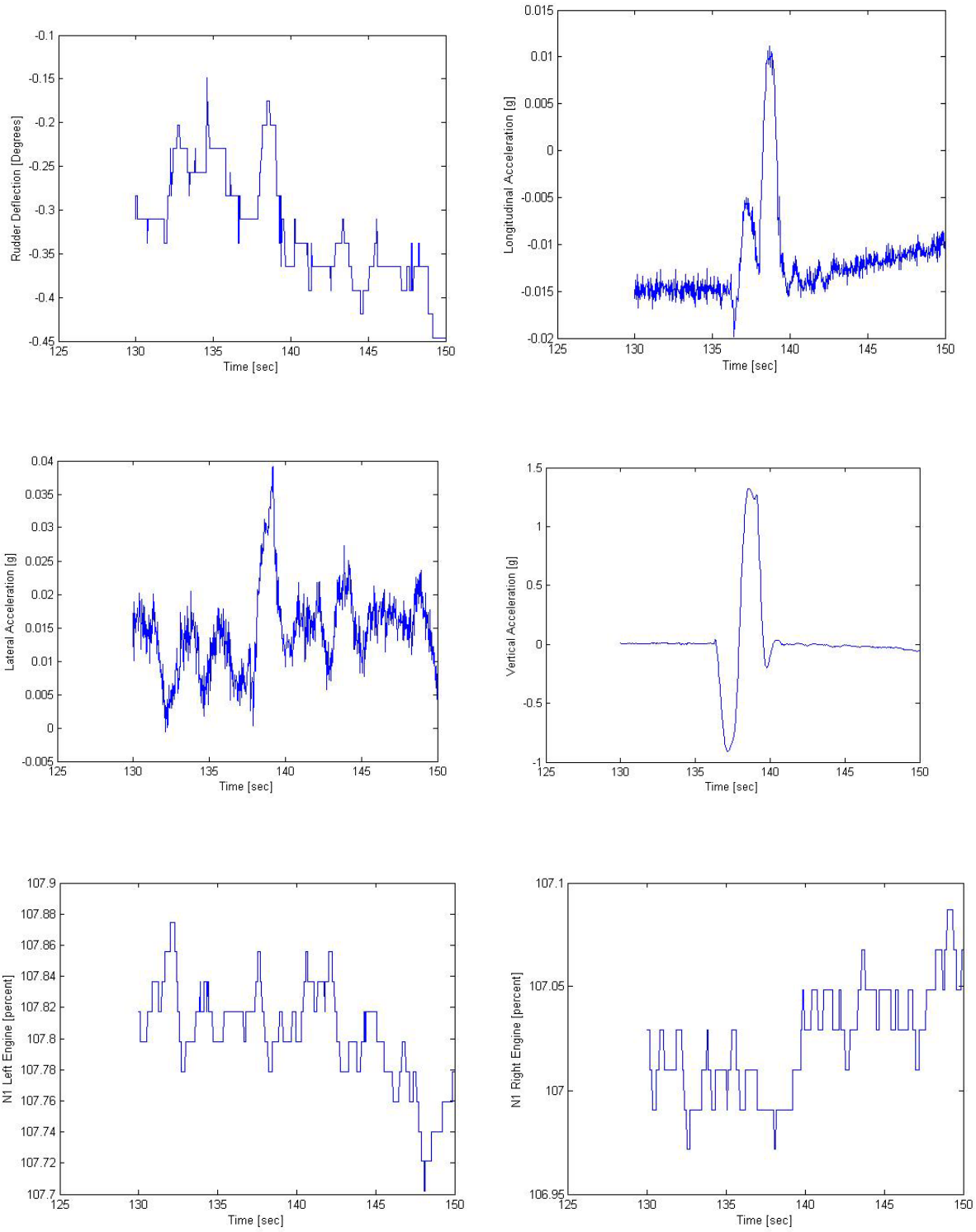


Figure B3. State, Control and Engine Variables of Altered Flight Data Segment for a Short Period Test.

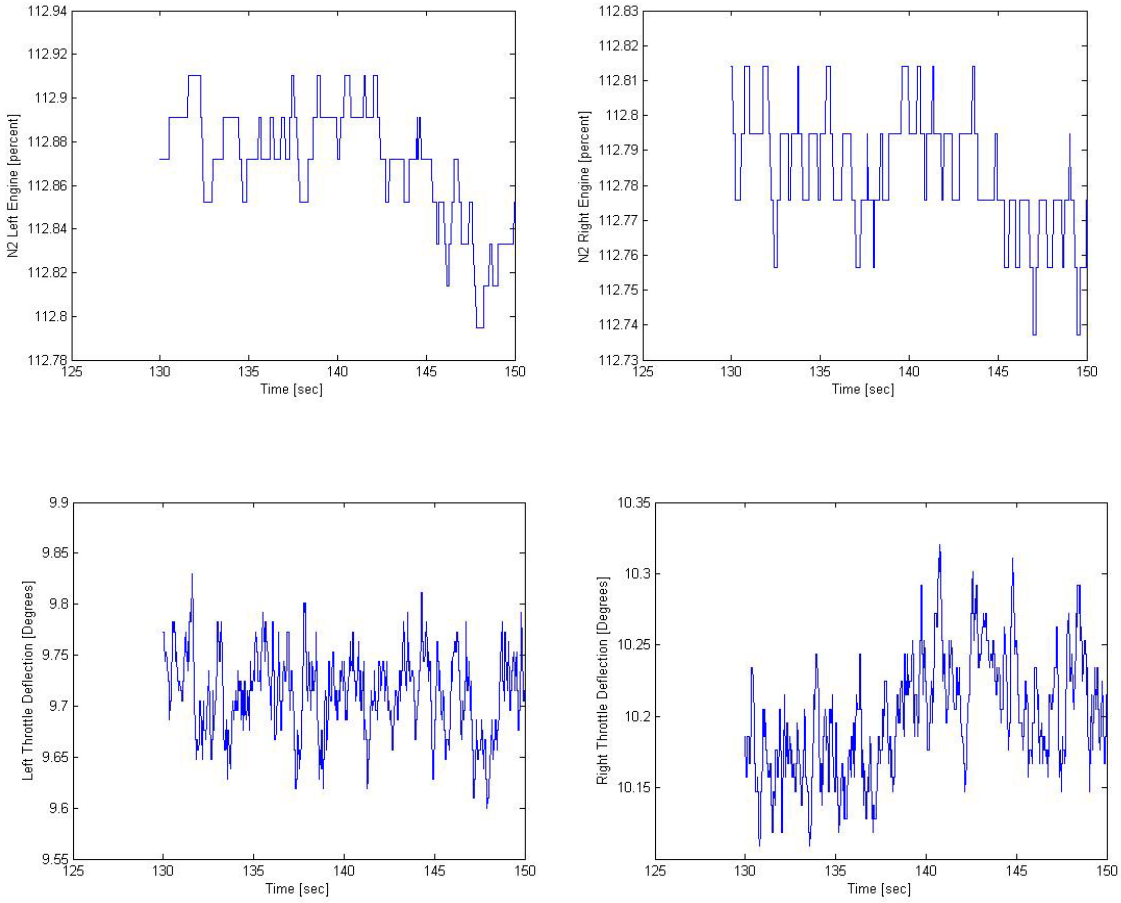


Figure B4. Engine Variables of Altered Flight Data Segment for a Short Period Test.

Appendix C – State Variable Model of Aircraft Dynamics

The longitudinal state and output equations are:

$$\begin{pmatrix} \dot{\alpha} \\ \dot{u} \\ \dot{q} \\ \dot{\theta} \end{pmatrix} = \begin{bmatrix} Z_\alpha{}' & Z_u{}' & Z_q{}' & Z_\theta{}' \\ X_\alpha{}' & X_u{}' & X_q{}' & X_\theta{}' \\ M_\alpha{}' & M_u{}' & M_q{}' & M_\theta{}' \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} \alpha \\ u \\ q \\ \theta \end{pmatrix} + \begin{bmatrix} Z_{\delta_E}{}' \\ X_{\delta_E}{}' \\ M_{\delta_E}{}' \\ 0 \end{bmatrix} (\delta_E) \tag{25}$$

$$\begin{pmatrix} a_z \\ \alpha \\ u \\ q \\ \theta \end{pmatrix} = \begin{bmatrix} Z_\alpha{}'' & Z_u{}'' & Z_q{}'' & Z_\theta{}'' \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \alpha \\ u \\ q \\ \theta \end{pmatrix} + \begin{bmatrix} Z_{\delta_E}{}'' \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} (\delta_E) \tag{26}$$

Where,

$$X'_\alpha = g\cos(\gamma_1) + X_\alpha \tag{27}$$

$$X'_u = X_u \tag{28}$$

$$X'_q = 0 \tag{29}$$

$$X'_\theta = -g\cos(\gamma_1) \tag{30}$$

$$X'_{\delta_E} = X_{\delta_E} \tag{31}$$

$$Z'_{\alpha} = \frac{g\cos(\gamma_1) + Z_{\alpha}}{(U_1 - Z_{\dot{\alpha}})} \tag{32}$$

$$Z'_{u} = \frac{Z_{u}}{(U_1 - Z_{\dot{\alpha}})} \tag{33}$$

$$Z'_{q} = \frac{(U_1 - Z_{q})}{(U_1 - Z_{\dot{\alpha}})} \tag{34}$$

$$Z'_{\theta} = \frac{g\sin(\gamma_1)}{(U_1 - Z_{\dot{\alpha}})} \tag{35}$$

$$Z'_{\delta_E} = \frac{Z_{\delta_e}}{(U_1 - Z_{\dot{\alpha}})} \tag{36}$$

$$M'_{\alpha} = M_{\dot{\alpha}}Z'_{\alpha} + M_{\alpha} \tag{37}$$

$$M'_{u} = M_{\dot{\alpha}}Z'_{u} + M_{u} \tag{38}$$

$$M'_{q} = M_{\dot{\alpha}}Z'_{q} + M_{q} \tag{39}$$

$$M'_{\theta} = M_{\dot{\alpha}}Z'_{\theta} \tag{40}$$

$$Z''_{\alpha} = U_1 Z'_{\alpha} - g\sin\gamma_1 \tag{41}$$

$$Z_u'' = U_1 Z_u' \tag{42}$$

$$Z_q'' = U_1 (Z_q' - 1) \tag{43}$$

$$Z_\theta'' = U_1 Z_\theta' + g sin\gamma_1 \tag{44}$$

$$Z_{\delta E}'' = U_1 Z_{\delta e}' \tag{45}$$

With the stability and control derivatives being,

$$X_u = \frac{-\bar{q}_1 S(C_{D_U} + 2C_{D_1})}{mU_1} \tag{46}$$

$$X_{T_u} = \frac{-\bar{q}_1 S(C_{T_{xu}} + 2C_{T_{x1}})}{mU_1} \tag{47}$$

$$X_\alpha = \frac{-\bar{q}_1 S(C_{D_\alpha} - C_{L_1})}{m} \tag{48}$$

$$X_{\delta_E} = \frac{-\bar{q}_1 S C_{D_{\delta E}}}{m} \tag{49}$$

$$Z_u = \frac{-\bar{q}_1 S(C_{L_U} + 2C_{L_1})}{mU_1} \tag{50}$$

$$Z_\alpha = \frac{-\bar{q}_1 S(C_{L_\alpha} + C_{D_1})}{m} \tag{51}$$

$$Z_{\dot{\alpha}} = \frac{-\bar{q}_1 S \bar{c} C_{L_{\dot{\alpha}}}}{2mU_1} \tag{52}$$

$$Z_q = \frac{-\bar{q}_1 S C_{L_q}}{2mU_1} \tag{53}$$

$$Z_{\delta_E} = \frac{-\bar{q}_1 S C_{L_{\delta E}}}{m} \tag{54}$$

$$M_u = \frac{\bar{q}_1 S \bar{c} (C_{m_U} + 2C_{m_1})}{U_1 I_{yy}} \tag{55}$$

$$M_{T_u} = \frac{\bar{q}_1 S \bar{c} (C_{m_{TU}} + 2C_{m_{T_1}})}{U_1 I_{yy}} \tag{56}$$

$$M_\alpha = \frac{\bar{q}_1 S \bar{c} C_{m_\alpha}}{I_{yy}} \tag{57}$$

$$M_{T_\alpha} = \frac{\bar{q}_1 S \bar{c} C_{m_{T\alpha}}}{I_{yy}} \tag{58}$$

$$M_{\dot{\alpha}} = \frac{\bar{q}_1 S \bar{c} C_{m_{\dot{\alpha}}}}{I_{yy}} \cdot \frac{\bar{c}}{2U_1} \tag{59}$$

$$M_q = \frac{\bar{q}_1 S \bar{c} C_{m_q}}{I_{yy}} \cdot \frac{\bar{c}}{2U_1} \tag{60}$$

$$M_{\delta_E} = \frac{\bar{q}_1 S \bar{c} C_{m_{\delta_E}}}{I_{yy}} \cdot \frac{\bar{c}}{2U_1} \tag{61}$$

The lateral-directional state and output equations are,

$$\begin{pmatrix} \dot{\beta} \\ \dot{p} \\ \dot{r} \\ \dot{\varphi} \end{pmatrix} = \begin{bmatrix} Y_\beta{}' & Y_p{}' & Y_r{}' & Y_\theta{}' \\ L_\beta{}' & L_p{}' & L_r{}' & 0 \\ N_\beta{}' & N_p{}' & N_r{}' & 0 \\ 0 & 1 & \tan(\theta_1) & 0 \end{bmatrix} \begin{pmatrix} \beta \\ p \\ r \\ \varphi \end{pmatrix} + \begin{bmatrix} Y_{\delta_A}{}' & Y_{\delta_R}{}' \\ L_{\delta_A}{}' & L_{\delta_R}{}' \\ N_{\delta_A}{}' & N_{\delta_R}{}' \\ 0 & 0 \end{bmatrix} \begin{pmatrix} \delta_A \\ \delta_R \end{pmatrix} \tag{62}$$

$$\begin{pmatrix} a_y \\ \beta \\ p \\ r \\ \varphi \end{pmatrix} = \begin{bmatrix} Y_\beta{}'' & Y_p{}'' & Y_r{}'' & Y_\varphi{}'' \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \beta \\ p \\ r \\ \varphi \end{pmatrix} + \begin{bmatrix} Y_{\delta_A}{}'' & Y_{\delta_R}{}'' \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} \delta_A \\ \delta_R \end{pmatrix} \tag{63}$$

Where,

$$Y'_\beta = \frac{Y_\beta}{U_1} \tag{64}$$

$$Y'_p = \frac{Y_p}{U_1} \tag{65}$$

$$Y'_r = \frac{Y_r}{U_1} - 1 \tag{66}$$

$$Y'_{\phi} = \frac{g\cos(\Theta_1)}{U_1} \tag{67}$$

$$Y'_{\delta A} = \frac{Y_{\delta a}}{U_1} \tag{68}$$

$$L'_{\beta} = \frac{(A_1 N_{\beta} + L_{\beta})}{(1 - A_1 B_1)} \tag{69}$$

$$L'_{p} = \frac{(A_1 N_{p} + L_{p})}{(1 - A_1 B_1)} \tag{70}$$

$$L'_{r} = \frac{(A_1 N_{r} + L_{r})}{(1 - A_1 B_1)} \tag{71}$$

$$L'_{\delta A} = \frac{(A_1 N_{\delta A} + L_{\delta A})}{(1 - A_1 B_1)} \tag{72}$$

$$L'_{\delta R} = \frac{(A_1 N_{\delta R} + L_{\delta R})}{(1 - A_1 B_1)} \tag{73}$$

$$N'_{\beta} = \frac{(B_1 L_{\beta} + N_{\beta})}{(1 - A_1 B_1)} \tag{74}$$

$$N'_p = \frac{(B_1 L_p + N_p)}{(1 - A_1 B_1)} \tag{75}$$

$$N'_r = \frac{(B_1 L_r + N_r)}{(1 - A_1 B_1)} \tag{76}$$

$$N'_{\delta A} = \frac{(B_1 L_{\delta A} + N_{\delta A})}{(1 - A_1 B_1)} \tag{77}$$

$$N'_{\delta R} = \frac{(B_1 L_{\delta R} + N_{\delta R})}{(1 - A_1 B_1)} \tag{78}$$

Where A_1 and B_1 and the lateral-directional stability and control derivatives are:

$$A_1 = \frac{I_{xz}}{I_{xx}} \tag{79}$$

$$B_1 = \frac{I_{xz}}{I_{xx}} \tag{80}$$

$$Y_\beta = \frac{\bar{q}_1 S C_{Y_\beta}}{m} \tag{81}$$

$$Y_p = \frac{\bar{q}_1 S C_{Y_p}}{m} \cdot \frac{b}{2U_1} \tag{82}$$

$$Y_r = \frac{\bar{q}_1 S C_{Y_r}}{m} \cdot \frac{b}{2U_1} \tag{83}$$

$$Y_{\delta_A} = \frac{\bar{q}_1 S C_{Y_{\delta_A}}}{m} \tag{84}$$

$$Y_{\delta_R} = \frac{\bar{q}_1 S C_{Y_{\delta_R}}}{m} \tag{85}$$

$$L_\beta = \frac{\bar{q}_1 S C_{l_\beta} b}{I_{xx}} \tag{86}$$

$$L_p = \frac{\bar{q}_1 S b C_{l_p}}{I_{xx}} \cdot \frac{b}{2U_1} \tag{87}$$

$$L_r = \frac{\bar{q}_1 S b C_{l_r}}{I_{xx}} \cdot \frac{b}{2U_1} \tag{88}$$

$$L_{\delta_A} = \frac{\bar{q}_1 S C_{l_{\delta_A}} b}{I_{xx}} \tag{89}$$

$$L_{\delta_R} = \frac{\bar{q}_1 S C_{l_{\delta_R}} b}{I_{xx}} \tag{90}$$

$$N_\beta = \frac{\bar{q}_1 S C_{n_\beta} b}{I_{zz}} \tag{91}$$

$$N_{T_\beta} = \frac{\bar{q}_1 S C_{n_{T_\beta}} b}{I_{zz}} \quad (92)$$

$$N_p = \frac{\bar{q}_1 S b C_{n_p}}{I_{zz}} \cdot \frac{b}{2U_1} \quad (93)$$

$$N_r = \frac{\bar{q}_1 S b C_{n_r}}{I_{zz}} \cdot \frac{b}{2U_1} \quad (94)$$

$$N_{\delta_A} = \frac{\bar{q}_1 S C_{n_{\delta_A}} b}{I_{zz}} \quad (95)$$

$$N_{\delta_R} = \frac{\bar{q}_1 S C_{n_{\delta_R}} b}{I_{zz}} \quad (96)$$

Appendix D – Flight Simulator Manual

To start the program type in JetSim in the Matlab command window. Once the program has started, the JetSim.fig will be the first figure to pop up.

There are two buttons that can be pressed. The FAA Compliance button is for the FAA tests that are required to certify the simulator. The Full Envelope button is the button for the model with all of the components, i.e. landing gear, flaps, trim, that combined for the final simulation model.

For FAA Compliance

Press the FAA Compliance button to continue.

The FAA Compliance button will then run the sel_modelfile.m file.

The sel_modelfile.m file will then open the FlightCondition.fig figure. On this menu, it will ask which condition out of the six. Each condition in the drop down menu is used as a flag for the programming code. The flag name is sel_FC and they run from 1 to 9 which are:

sel_FC	Flight Condition
1	Cruise
2	Approach
3	Normal Climb, Clean

4	Steady State Sideslip
5	Acceleration on the Ground
6	Acceleration on Approach
7	Longitudinal Trim
8	Flaps Operating Time
9	Landing Gear Operating Time

Once the Continue button on the FlightCondition.fig menu has been pressed, the program then runs the sel_inp.m file. This file in turn then opens the PilotInput.fig figure. On this figure the types of inputs are displayed on the left side of the figure. When all stick inputs is selected and the Continue button is pressed the program loads the allstk.m file. For Pre-recorded inputs the file allrec.m is loaded and the channels that are recorded and the names of the files are required on this figure. The mixed input loads the mixed.m file and may use the pre-recorded channels part of the menu if applicable. Finally if the all from flight file is selected the allflight.m file is loaded. Regardless of which input is used, the file Ok_Pilotinput.m is loaded once the continue button is pressed.

The Ok_Pilotinput.m file then loads the corresponding figure for sel_FC flag. Where:

sel_FC=1 -- Reg_Cruise.fig

sel_FC=2 -- Reg_Approach.fig

sel_FC=3 -- Reg_Climb.fig

sel_FC=4 -- Reg_Sideslip.fig

sel_FC=5 -- Reg_Ground.fig

sel_FC=6 -- Reg_Acc.fig

sel_FC=7 -- Reg_LgTrim.fig

sel_FC=8 -- Reg_Flaps.fig

sel_FC=9 -- Reg_LdGear.fig

Each figure is set up similarly in that the drop menu is used to select the wanted requirements. This drop down menu is also used to get flags that are named sel_test. Once the requirement is selected the program then runs the InitializeFE.m file.

The InitializeFE.m file is responsible for loading all of the aerodynamic look up tables, thrust, and constant parameters. Also, from the sel_FC and sel_test values, the corresponding flight data is loaded and the initial conditions of the flight are loaded. For the flight data it is important to make sure that the file path is correct so that it will be loaded. The flight data loads the following data and what the variable name is in the program.

Name	Variable in Program
Angle of Attack	alpha_saved

Velocity	kias_saved
Sideslip Angle	beta_saved
Roll Angle	phi_saved
Pitch Angle	theta_saved
Yaw Angle	psi_saved
Roll Rate	p_saved
Pitch Rate	q_saved
Yaw Rate	r_saved
Longitudinal Acceleration	accx_saved
Lateral Acceleration	accy_saved
Vertical Acceleration	accz_saved
Start Time	start_save
End Time	end_save
Aileron Deflection	da_saved
Elevator Deflection	de_saved
Rudder Deflection	dr_saved

Left Throttle Deflection	thrL_saved
N1 of Left Engine	N1_L_saved
N2 of Left Engine	N2_L_saved
Right Throttle Deflection	thrR_saved
N1 of Right Engine	N1_R_saved
N2 of Right Engine	N2_R_saved
Rate of Climb	ROC_saved

For each flight condition and test, an initial conditions file is loaded that has the initial conditions of the test, which include the mass moments of inertia. These files' names are in the format of IC_##_$$. Where the ## inidicates the number of the flight the data was taken from and $$ indicates the numbering of the file. Finally, the time step used for the simulation and the end time of the simulation are created in this file. The time step is the variable known as T and the end time of the simulation is SimTime.

Once the data is loaded then the sel_FC flags are used to open two more files. These are:

sel_FC	File 1	File 2
1	OpenScopes2.m	Jet_FE.mdl
2	OpenScopes2.m	Jet_FE.mdl

3	OpenScopes3.m	Jet_FE.mdl
4	OpenScopes2.m	Jet_FE.mdl
5	OpenScopes4.m	Jet_FE.mdl
6	OpenScopes4.m	Jet_FE.mdl
7	OpenScopes4.m	Jet_FE.mdl

For sel_FC=8 and sel_test=1, the FlapsOT_R.m the switch in the Jet_FE.mdl for flaps must be switched to FAA compliance to read the inputs.

For sel_FC=8 and sel_test=2, the FlapsOT_E.m the switch in the Jet_FE.mdl for flaps must be switched to FAA compliance to read the inputs.

For sel_FC=9 and sel_test=1, the LdGearOT_R the switch in the Jet_FE.mdl for landing gear must be switched to FAA compliance to read the inputs.

For sel_FC=9 and sel_test=2, the LdGearOT_E the switch in the Jet_FE.mdl for landing gear must be switched to FAA compliance to read the inputs.

Once the OpenScopes files are loaded, the 2, 3, or 4 following the OpenScopes in the file name is used to open up the corresponding Scopes.fig.

In these new scopes figure, the scopes that are required can be selected and viewed as the program is running. Once the OK button is pressed the corresponding Ok_Scopes.m file is loaded which is what is used to open the scopes.

In the Jet_FE.mdl files, in the Data Manager file, the block named FlightData15 takes all the variables saved from the flight data above and then sends it to the scopes to be compared with the simulator outputs.

The Jet_FE.mdl files are then used by selecting the amount of time to run the simulation or uses the SimTime variable for the test. In this block are a Scopes1, Scopes2, and To Workspace block. The Scopes blocks have the scopes that are opened up by the OpenScopes.m files. The To Workspace block contains blocks that save the simulation data, the limits of the tests, and the flight data to matrices in the workspace. These matrices have the data saved such that the first column is the simulation data, the second column is the flight data, the third column is the upper limit and the fourth is the lower limit for the FAA Compliance Tests.

Once the simulation is run, double-clicking the Plots for FAA Compliance block will run the Subplot_data.m file. In this file, for the corresponding sel_FC and sel_test, the To Workspace variables that are needed for the requirements of the test are then used to plot and find other parameters, i.e. time constants, damping, frequency, and to display the initial conditions of the flight.

Full Envelope

Once the full envelope button is pushed, sel_modelfile.m is run and the PilotInput.fig is opened. From this menu only the all stick input selection can be selected which uses the allstk.m file. Once that is selected, and the OK button is pressed, the OK_PilotInput.m file is run and will then open the Initial Conditions of the simulation. There are currently three initial conditions. The first initial condition is the airplane is in the air with a velocity of 125 m/s and 4450 m ASL with the sel_test value being 1. The second condition is starting on the ground with zero velocity and

at 4450 m ASL and the sel_test value being 2. The third condition has a starting velocity of 65 m/s and an altitude of 530 m. Once the OK button is selected on this the InitializeFE.m file is run. In this file, all of the aerodynamic look up tables, the thrust look up tables, constants, sensor bias, and mass moment of inertia are loaded. Then the Jet_FE.mdl file is opened. For this model, there is no subplot data block.

In the Jet_FE simulink file, there is a block with that is titled Pilot1, this block contains all of the inputs. The joystick block is the actual physical input device and its outputs. The axes outputs then go to the longitudinal, lateral, or directional channels, or the button outputs are the button outputs that go to the brake command, aileron trim, elevator trim, or rudder trim. These buttons commands are then separated using a selector block. For the default the selector for the brake command uses button 1 which when you double click selector will show that value in an element vector. For the aileron trim, the default buttons are 3 and 4 in that order, the position where the 3 is in makes the aileron trim left while the position of 4 trims the aileron right. For the elevator trim, 7 and 8 are the default buttons and the position of 7 trims the elevators for a nose down position while the 8 position trims the elevators nose up. The rudder trim selectors default values are 5 and 6 where the position of 5 trims the rudder for the nose left and the 6 trims the nose right. These values can be changed for a more convenient button selection. To calibrate the joystick axes the constants kstkdr (for the directional channel), kstklg (for the longitudinal channel), kstklt (for the lateral channel) are used for the calibration, and kthrhist (for the thrust channel). Increasing these values will increase the effectiveness of control input to the control surface deflections. These values will depend on the kind of joystick or input device you are using so it must be set up for individual devices.

Lastly for the full envelope another figure is opened up that has four different buttons. The first button is the pause button and this button will pause the simulation until the pause button is hit again. The position freeze button is a toggle button that will freeze the aircraft at a certain but will be able to be rotated about all three axis while in the frozen position. The Altitude freeze button is a toggle that will freeze the aircraft at its current altitude while the heading and rotation about the 3 axis can be changed. Reposition button will bring up another menu to change the heading, latitude, longitude, and altitude of the aircraft and is confirmed by hitting the OK button at the bottom of the figure.

Table of Variables

InitializeFE.m

InitializeFE.m	Descriptor	Type
alphaLT	Angle of attack	conversion factor
t	Altitude for thrust look up table	look up Table
b	Wing span	variable
brake_cmd	Braking Input	input
c	Chord	variable
Cx0	Initial X-axis force coefficient	variable
CxV	X-axis force variation with respect to velocity	variable
Cxa	X-axis force variation with respect to alpha	variable
Cxq	X-axis force variation with respect to pitch rate	variable
Cxadot	X-axis force variation with respect to rate of change of alpha	variable
Cxde	X-axis force variation with respect to elevator deflection	switch
CxLG	X-axis force variation with respect to landing gear	variable
CxFL	X-axis force variation with respect to flaps	variable
Cz0	Initial Z-axis force coefficient	variable
CzV	Z-axis force coefficient variation with respect to velocity	switch
Cza	Z-axis force coefficient variation with respect to alpha	variable
Czq	Z-axis force coefficient variation with respect to pitch rate	variable
Czadot	Z-axis force coefficient variation with respect to change in alpha	variable
Czde	Z-axis force coefficient variation with respect to deflection of elevator	variable
CzFL	Z-axis force coefficient variation with respect to flap deflection	variable
Cm0	Initial pitch coefficient	variable
CmV	Pitch coefficient variation with respect to velocity	look up table

Cma	Pitch coefficient variation with respect to alpha	look up table
Cmq	Pitch coefficient variation with respect to rate of pitch	look up table
Cmadot	Pitch coefficient variation with respect to rate of change in alpha	look up table
Cmde	Pitch coefficient variation with respect to elevator deflection	look up table
CmFL	Pitch coefficient variation with respect to flaps	variable
Cy0	Initial side force coefficient	variable
Cyb	Side force coefficient variation with respect to sideslip angle	variable
Cyp	Side force coefficient variation with respect to roll rate	variable
Cyr	Side force coefficient variation with respect to yaw rate	variable
Cyda	Side force coefficient variation with respect to aileron deflection	variable
Cydr	Side force coefficient variation with respect to rudder deflection	variable
Cl0	Initial lift coefficient	variable
Clb	Lift coefficient variation with respect to sideslip angle	variable
Clp	Lift coefficient variation with respect to rate of roll	variable
Clr	Lift coefficient variation with respect to rate of yaw	variable
Clda	Lift coefficient variation with respect to aileron deflection	variable
Cldr	Lift coefficient variation with respect to rudder deflection	variable
Cn0	Initial yaw coefficient	variable
Cnb	Yaw coefficient variation with respect to sideslip angle	variable
Cnp	Yaw coefficient variation with respect to roll rate	variable
Cnr	Yaw coefficient variation with respect to yaw rate	variable
Cnda	Yaw coefficient variation with respect to aileron deflection	variable
Cndr	Yaw coefficient variation with respect to rudder deflection	variable
cg_at	Center of gravity	variable
c1N	Damping constant nose gear	variable

c1M	Damping constant main gear	variable
d2r	Degrees to radians	variable
dirHist_s.time	Recorded pilot input dir input time	input
dirHist_s.signals.values	Recorded pilot input dir input	input
dirHist_s.signals.dimensions	Recorded pilot input dir input dimensions	input
dynPresLT	dynamic pressure	variable
engang1	Engine angles	vector
fric_coef	Friction coefficient	variable
Fric_ratio_N	Friction ratio for nose gear	variable
Fric_ratio_M	Friction ratio for main gear	variable
g	Gravity	variable
GM1	Mass moment of inertia vector	variable
GM2	Inertia coefficients vector	variable
Igrnd	Velocity threshold for alpha and beta	variable
Ixx	Mass moment of inertia	variable
Iyy	Mass moment of inertia	variable
Izz	Mass moment of inertia	variable
Ixy	Mass moment of inertia	variable
Ixz	Mass moment of inertia	variable
Iyz	Mass moment of inertia	variable
I	Inertia coefficients	variable
I1	Inertia coefficients	variable
I2	Inertia coefficients	variable
I3	Inertia coefficients	variable
I4	Inertia coefficients	variable
I5	Inertia coefficients	variable
I6	Inertia coefficients	variable
kLG	Landing gear switch	variable
kFL	Flaps switch	switch
k1N	Spring constant 1 nose gear	variable
K2N	Spring constant 2 nose gear	variable
k1M	Spring constant 1 main gear	variable
k2M	Spring constant 2 main gear	variable
kstklg	Longitudinal stick constant	variable
kstklt	Lateral Stick constant	variable
kstkdr	Directional stick constant	variable
klghist	Longitudinal time history constant	vector

klthist	Lateral time history constant	vector
kdirhist	Directional time history constant	variable
kthrhist	Throttle time history constant	variable
Lgdeploy	Velocity threshold for alpha and beta	variable
Lecg	Left engine center of gravity	variable
lgHist_s.time	Recorded pilot input longitudinal input time	input
lgHist_s.signals.values	Recorded pilot input longitudinal input	input
lgHist_s.signals.dimensions	Recorded pilot input longitudinal input dimensions	input
ltHist_s.time	Recorded pilot input lateral input time	input
ltHist_s.signals.values	Recorded pilot input lateral input	input
ltHist_s.signals.dimensions	Recorded pilot input lateral input dimensions	input
MachLT	Mach for thrust look up table	variable
mwgN	Mass of nose wheel	variable
mwgM	Mass of main wheels	variable
max_brake	Maximum braking rorce	variable
N1w1	Engine model	variable
N1zeta1	Engine model	variable
N1w2	Engine model	variable
N1zeta2	Engine model	variable
N1delay	Engine model	variable
N1k	Engine model	variable
N2w1	Engine model	variable
N2zeta1	Engine model	variable
N2delay	Engine model	variable
N2k	Engine model	variable
posN	Nose gear position in body axes	variable
posML	Left landing gear position in body axes	vector
posMR	Right landing gear position in body axes	variable
PureDelayN1	Engine model	variable
PureDelayN2	Engine model	variable
P1	Inertia coefficients	variable
Pm	Inertia coefficients	variable
Pn	Inertia coefficients	variable

Ppp	Inertia coefficients	variable
Ppq	Inertia coefficients	variable
Ppr	Inertia coefficients	variable
Pqq	Inertia coefficients	variable
Pqr	Inertia coefficients	variable
Prr	Inertia coefficients	variable
Q1	Inertia coefficients	variable
Qm	Inertia coefficients	variable
Qn	Inertia coefficients	variable
Qpp	Inertia coefficients	variable
Qpq	Inertia coefficients	variable
Qpr	Inertia coefficients	variable
Qqq	Inertia coefficients	variable
Qqr	Inertia coefficients	variable
Qrr	Inertia coefficients	variable
Recg	Right engine center of gravity	variable
RtireN	Nose tire radius	variable
RtireM	Main tire radius	variable
R1	Inertia coefficients	variable
Rm	Inertia coefficients	variable
Rn	Inertia coefficients	variable
Rpp	Inertia coefficients	variable
Rpq	Inertia coefficients	variable
Rpr	Inertia coefficients	variable
Rqq	Inertia coefficients	variable
Rqr	Inertia coefficients	variable
Rrr	Inertia coefficients	variable
S	Wing surface area	variable
SimTime	Time increment	vector
steer_ang	Steering angle	variable
SPEV_Force.time	Force special event time	special event
SPEV_Force.signals.values	Force special event values	special event
SPEV_Force.signals.dimensions	Force special event Dimensions	special event
SPEV_Moment.time	Moment special event time	special event
SPEV_Moment.signals.values	Moment special event values	special event
SPEV_Moment.signals.dimensions	Moment special event dimenstions	special event
T	Time increment	variable
Thrust	Thrust	variable
Vhelp	Avoid division by zero	variable
Vthres	Avoid division by zero	variable
VrefND	Reference velocity	variable

xawN	Length of nose landing gear	variable
xawM	Length of main landing gear	variable
xainN	Nose gear height above ground	variable
xainM	Main gear height above ground	variable

IC_##_$$

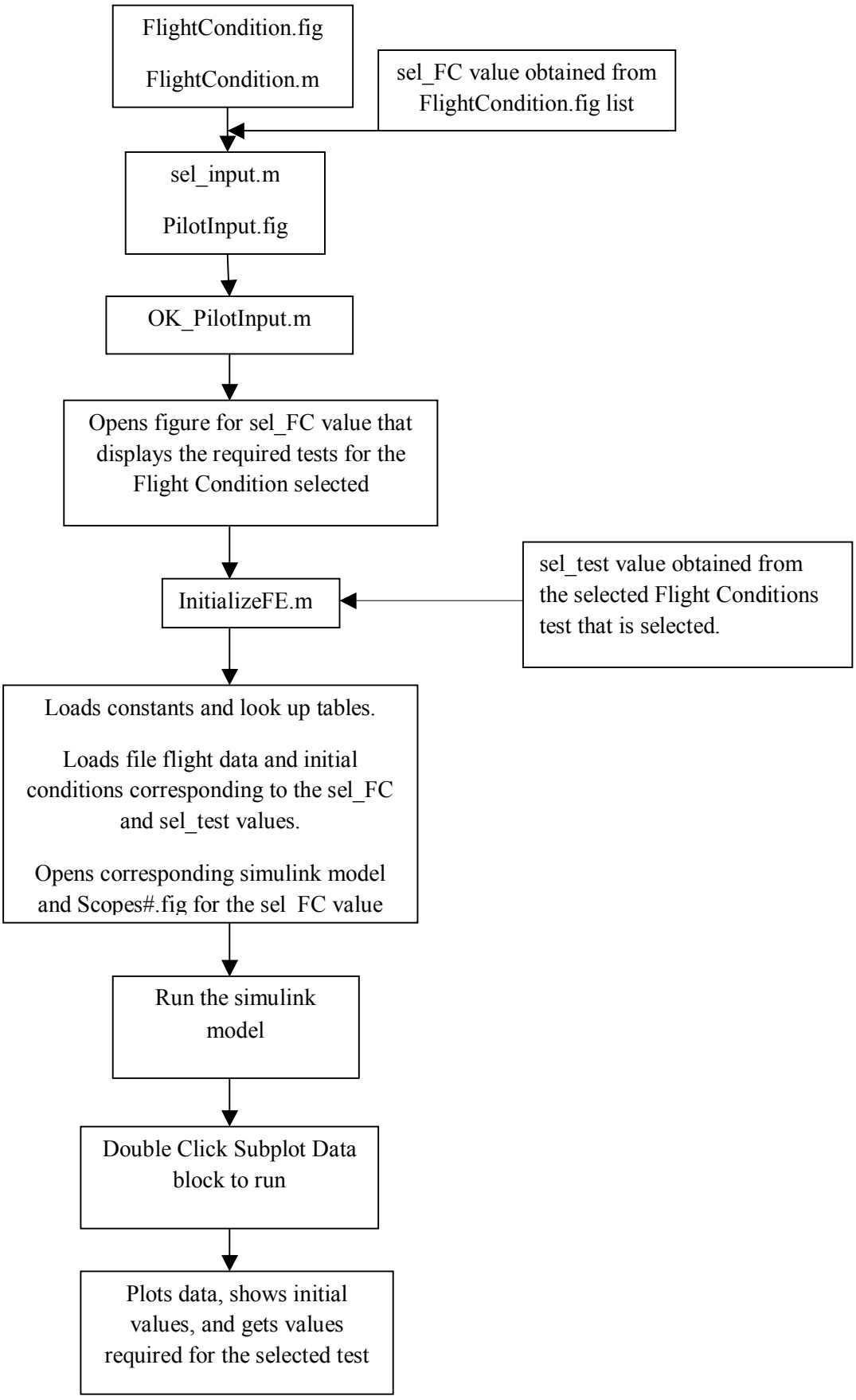
IC_##_$$.m	Descriptor	Unit	Type
alpha0	Initial angle of attack	radians	variable
alt	Altitude	m	variable
alt_ft	Altitude	feet	variable
alt_ft_aport	Altitude of airport in feet	feet	variable
alt_aport	Altitude of airport in meters	meters	variable
beta0	Initial sideslip angle	radians	variable
Init_Long	Initial longitude position	N/a	variable
Init_lat	Initial lateral position	N/A	variable
in_flaps	Initial flaps position	Degrees	variable
N1_init	Initial N1	%	variable
N2_init	Initial N2	%	variable
p0	Initial roll rate	rad/s	variable
psi0	Initial yaw angle	radians	variable
phi0	Initial roll angle	radians	variable
q0	Initial pitch rate	rad/s	variable
ro	density	kg/m^3	variable
r0	Initial yaw rate	rad/s	variable
uaero0	Initial surface deflections	radians	variable
uprop0	Initial throttle	N/A	variable
u0inco	Initial velocity	m/s	variable
V0	Initial velocity	m/s	variable
v0inco	Initial velocity	m/s	variable
w0inco	Initial velocity	m/s	variable
xinco	Initial conditions	N/A	vector
X0	Initial position in X	m	variable
y0	Initial controls vector	N/A	vector
Y0	Initial position in Y	m	variable
Z0	Initial position in Z(altitude)	m	variable

File Functions

File_name	Function
allflight.m	Function if all flight input selected
allrec.m	Function if all recorded input selected
allstk.m	Function if all stick input selected
AltF31.m	Altitude table(needed for ALTflight.mat file)
ALTflight.mat	Table of altitudes
AltitudeFreeze.m	Function freezes altitude
Ans_Block.mat	Gives the initial conditions and damping and frequency answers in cells
Jet_FE.mdl	Actual model
JetSim.fig	Figure that starts the whole simulation
JetSim.m	Model
CR.mat	Climb rate flight data files
crate.m	Creates the CR.mat climb rate data files
dirhist.mat	Directional history file (converts flight data into a table matlab can use)
dirhist_crnt.mat	Directional history file (converts flight data into a table matlab can use)
dirhist_zero.mat	Directional history file (converts flight data into a table matlab can use)
flaps_OT_E.mat	Faps extension data
flaps_OT_R.mat	Flaps retraction data
FlightCondition.fig	Flight condition figure
FlightConditon.m	
IC_$$_##.m	Initial condition files used to load initial conditions for the correct test in the InitializeFE.m file
IntializeFE.m	Initializes all flight model data, including landing gear and engine modeling
ldgear_OT_E.mat	Landing gear extenstion data
ldgear_OT_R.mat	Landing gear retraction data
lghist.mat	Longitudinal history file (converts flight data into a table matlab can use)
lghist_crnt.mat	Longitudinal history file (converts flight data into a table matlab can use)
lghist_save.mat	Longitudinal history file (converts flight data into a table matlab can use)
lghist_zero.mat	Longitudinal history file
lthist.mat	Lateral history file (converts flight data into a table matlab can use)
lthist_crnt.mat	Lateral history file (converts flight data into a table matlab can use)
lthist_save.mat	Lateral history file (converts flight data into a table matlab can use)
lthist_zero.mat	Lateral history file (converts flight data into a table matlab can use)
mixed.m	Function to load mixed signals i.e. recorded and pilot input
Modes.m	General form of an S-function
OK_PilotInput.m	OK button function if pilot input selected

OK_Repos.m	OK button function for the repositon button
OK_Scopes.m	OK button function for scopes
OK_Scopes2.m	OK button function for scopes
OK_Scopes3.m	OK button function for scopes
OK_Scopes4.m	OK button function for scopes
OK_ScopesFE.m	OK button function for scopes
PauseFig.fig	Figure that pauses the simulation
PauseFig.m	
PauseFlags.m	Pause flags
PauseReturn.m	
PilotInput.fig	Pilot input figure (initial conditions for pilot input)
PilotInput.m	
PostionFreeze.m	Position freeze function for the position freeze button
rec_input.m	Function to load recorded inputs
Reg_Acc.fig	Acceleration flight condition FAA compliance menu
Reg_Approach.fig	Approach flight condition FAA compliance menu
Reg_Climb.fig	Climb flight condition FAA compliance menu
Reg_Cruise.fig	Cruise flight condition FAA compliance menu
Reg_FE.fig	Piloted flight, flight simulation menu
Reg_Flaps.fig	Flaps flight condition FAA compliance menu
Reg_Ground.fig	Ground flight condition FAA compliance menu
Reg_LdGear.fig	Landging gear flight conditon FAA compliance menu
Reg_LgTrim.fig	Longitudinal trim flight conditon FAA compliance menu
Reg_Sideslip.fig	Sideslip flight condition FAA compliance menu
Reposition.m	Repostion function
RespostionFig.fig	Figure to select new position after the reposition button is selected
Scopes#.fig	Scopes figures
thrhist.mat	Thrust history file (converts flight data into a table matlab can use)
thr_crnt.mat	Thrust history file (converts flight data into a table matlab can use)
thrhist_save.mat	Thrust history file (converts flight data into a table matlab can use)
thrhist_zero.mat	Thrust history file (converts flight data into a table matlab can use)

FAA Compliance Diagram



Full Envelope Diagram