Investigation of Lateral-Directional Coupling in Longitudinal Responses of a Transfer Function Simulation Model

John R. Leonard

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute & State University
In partial fulfillment of the requirements for the degree of

Master of Science
In
Aerospace Engineering

Dr. Wayne Durham – Chair
Dr. Frederick Lutze
Dr. Craig Woolsey
Dr. Naira Hovakimyan

December 05, 2003
Blacksburg, Virginia

Keywords: Flight Simulation, Transfer Functions, Mode Coupling, Longitudinal Responses

Investigation of Lateral-Directional Coupling in Longitudinal Responses of a Transfer Function Simulation Model

John R. Leonard

(ABSTRACT)

The linear variable stability Transfer Function Simulation Model (TFSM), inspired by the United States Air Force's NF-16D Variable stability In-flight Simulator Test Aircraft (VISTA) and created by Henrik Pettersson, can simulate any desired aircraft. The TFSM represents a non-linear aircraft model with its stability parameters – a collection of gain constants, time constants, damping ratios, and natural frequencies. Stability parameters for aircraft generally fall into two uncoupled modes: longitudinal and lateral-directional. Unfortunately, flight tests using the TFSM exhibited undesired lateral-directional coupling in the longitudinal responses.

An *S*-turn maneuver, formation flight test, and an uncontrolled simulation with an initial bank angle of 60 degrees were the foundation for the investigation to pinpoint the TFSM's errors. The flight tests and subsequent analysis showed that although this model is highly versatile, it has three fundamental problems. First, the original creation of the TFSM incorrectly assumed that the time rate of change for the pitch angle (in the local-horizontal reference frame) is equal to the body-axis pitch-rate for all flight conditions. Second, the TFSM's dynamics do not contain gravity terms. Third, the TFSM cannot generate the angular rates needed in a turn.

Integrating the aircraft's pitch, roll, and yaw angles with the equations of motion for aircraft fixed the first problem. Unfortunately, resolving this issue only intensified the second two problems. The results from this thesis show that the last two problems are part of the TFSM and cannot be fixed explicitly.

Acknowledgments

I would like to thank my family for their steadfast love and support during my entire life. To my parents, thank you for making me believe in myself. Observing you throughout my life, I have grown to appreciate and adopt your drive to always work to the best of your abilities and perfect what you do. Granted this kept me working for long hours during my career here at Virginia Tech, but I was always happy with my final products. To Kevin, you always gave me an older brother that I was able to look up to and depend on. You were always a role model and paved many roads before me, making my life that much smoother.

I would also like to thank my advisor, Dr. Wayne Durham, for his continuous support and great store of knowledge. I would never have gotten to where I am today without his help. To the rest of my advisory committee, Dr. Craig Woolsey, Dr. Naira Hovakimyan, and the late Dr. Frederick Lutze, thank you for your time and invaluable insight. To Bill Oetjens, thank you for always being around to lend a helping hand and for keeping the Flight Simulation Laboratory running so smoothly. To my friends, thank you for being my friends. You have made my time here at Virginia Tech unforgettable. Last but not least, to Theresa Crooks, you were invaluable to me for keeping me sane during my long weeks at work and for editing this thesis.

Thank you all.

Table of Contents

List of Figures

List of Tables

Nomenclature

Acronyms:

CASTLE	Control Analysis and Simulation Test Loop Environment
RK4	Runge-Kutta fourth-order Integration Method
TFSM	Transfer Function Simulation Model
VISTA	Variable stability In-flight Simulator Test Aircraft

Aircraft Variables:

V_T	Total velocity
α	Angle of attack
β	Sideslip angle
u	Body x-axis velocity
v	Body y-axis velocity
w	Body z-axis velocity
φ	Euler roll angle
θ	Euler pitch angle
ψ	Euler yaw angle
P	Body-axis roll rate
Q	Body-axis pitch rate
R	Body-axis yaw rate
U	Body-axis x-axis velocity
V	Body-axis y-axis velocity

W	Body-axis z-axis velocity
p_N	North position of the aircraft
p_E	East position of the aircraft
h	Aircraft's altitude
g	Acceleration due to gravity
X	Body-axis x-axis force
Y	Body-axis y-axis force
Z	Body-axis z-axis force
L	Body-axis rolling moment
M	Body-axis pitching moment
N	Body-axis yawing moment
m	Aircraft mass
I_{xx}	x-axis moment of inertia
I_{yy}	y-axis moment of inertia
I_{zz}	z-axis moment of inertia
I_{xz}	x-z product of inertia
δ_{th}	Throttle setting
δ_e	Elevator deflection
δ_a	Aileron deflection
δ_r	Rudder deflection

Transfer Function Variables:

$K_{\alpha\delta e}$	Gain for δe to $\delta\alpha$ transfer function
$K_{\alpha\delta th}$	Gain for δth to $\delta\alpha$ transfer function
$K_{\beta\delta a}$	Gain for δa to $\delta\beta$ transfer function
$K_{\beta\delta r}$	Gain for δr to $\delta\beta$ transfer function
$K_{P\delta a}$	Gain for δa to δR transfer function
$K_{P\delta r}$	Gain for δr to δR transfer function
$K_{Q\delta e}$	Gain for δe to δQ transfer function
$K_{Q\delta th}$	Gain for δth to δQ transfer function
$K_{R\delta a}$	Gain for δa to δP transfer function

$K_{R\delta r}$	Gain for δr to δP transfer function
$K_{U\delta e}$	Gain for δe to δU transfer function
$K_{U\delta th}$	Gain for δth to δU transfer function
ω_{dr}	Dutch roll natural frequency
ω_{sp}	Short period natural frequency
ω_{p}	Phugoid natural frequency
$\omega_{U\delta T}$	Natural frequency for δth to δu transfer function
$\omega_{U\delta e}$	Natural frequency for δe to δu transfer function
$\omega_{\alpha\delta e}$	Natural frequency for δe to $\delta \alpha$ transfer function
$\omega_{\beta\delta a}$	Natural frequency for δa to $\delta \beta$ transfer function
$\omega_{P\delta a}$	Natural frequency for δa to δP transfer function
$\omega_{R\delta a}$	Natural frequency for δa to δR transfer function
$\omega_{R\delta r}$	Natural frequency for δr to δR transfer function
ζ_{dr}	Dutch roll damping ratio
ζ_{p}	Phugoid damping ratio
ζ_{sp}	Short Period damping ratio
$\zeta_{U\delta T}$	Damping ratio for δth to δu transfer function
$\zeta_{U\delta e}$	Damping ratio for δe to δu transfer function
$\zeta_{\alpha\delta e}$	Damping ratio for δe to $\delta \alpha$ transfer function
$\zeta_{\beta\delta a}$	Damping ratio for δa to $\delta \beta$ transfer function
$\zeta_{P\delta a}$	Damping ratio for δa to δP transfer function
$\zeta_{R\delta a}$	Damping ratio for δa to δR transfer function
$\zeta_{R\delta r}$	Damping ratio for δr to δR transfer function
T_{r}	Roll mode time constant
T_{s}	Spiral mode time constant
$T_{U\delta T}$	Time Constant for δth to δu transfer function
$T_{U\delta e}$	Time Constant for δe to δu transfer function
$T_{\alpha\delta T1}$	1st Time Constant for δth to $\delta \alpha$ transfer function
$T_{\alpha\delta T2}$	2nd Time Constant for δth to $\delta \alpha$ transfer function

$T_{\alpha\delta T3}$	3rd Time Constant for δth to $\delta\alpha$ transfer function
$T_{\alpha\delta e}$	Time Constant for δe to $\delta\alpha$ transfer function
$T_{Q\delta T1}$	1st Time Constant for δth to δQ transfer function
$T_{Q\delta T2}$	2nd Time Constant for δth to δQ transfer function
$T_{Q\delta e1}$	1st Time Constant for δe to δQ transfer function
$T_{Q\delta e2}$	2nd Time Constant for δe to δQ transfer function
$T_{Q\delta e3}$	3rd Time Constant for δe to δQ transfer function
$T_{\beta\delta a}$	Time Constant for δa to $\delta\beta$ transfer function
$T_{\beta\delta r1}$	1st Time Constant for δr to $\delta\beta$ transfer function
$T_{\beta\delta r2}$	2nd Time Constant for δr to $\delta\beta$ transfer function
$T_{\beta\delta r3}$	3rd Time Constant for δr to $\delta\beta$ transfer function
$T_{P\delta a}$	Time Constant for δa to δP transfer function
$T_{P\delta r1}$	1st Time Constant for δr to δP transfer function
$T_{\phi\delta r2}$	2nd Time Constant for δr to δP transfer function
$T_{P\delta r3}$	3rd Time Constant for δr to δP transfer function
$T_{R\delta a}$	Time Constant for δa to δR transfer function
$T_{R\delta r}$	Time Constant for δr to δR transfer function

Chapter 1

Introduction

1.1 Background

Henrik Pettersson [1] designed the linear variable stability Transfer Function Simulation Model (TFSM) to have one model that could easily simulate any desired aircraft. The Air Force's NF-16D Variable stability In-flight Simulator Test Aircraft (VISTA) (Figure 1.1) was the inspiration behind the original development of the TFSM. The United States Air Force Research Laboratory (AFRL) in conjunction with Veridian Engineering Flight Research Group and Lockheed Martin developed the VISTA to be a high performance in-flight simulator and a unique test bed for flight research. An F-16D airframe was highly modified to create the five degree-of-freedom (pitch, roll, yaw, direct lift, and thrust modulation) in-flight simulator that had the basic flight characteristics of modern fighter aircraft. Tables 1.1 and 1.2 give the VISTA's flight performance and simulation characteristics limitations with the Variable Stability System (VSS). [2] These limits exist because the VISTA is dependent on the actual F-16D airframe's capabilities. Unlike the VISTA, the TFSM, as a ground-based simulation, does not have any restrictions on its flight envelope because it is not dependent on the performance of an actual airframe. The VISTA does have limitations, but as shown in Table 1.2, it can emulate any number of aircraft by changing its aircraft dynamics (like its Short Period and Dutch Roll

stability parameters). Similarly, the TFSM calculates responses to control inputs using the original aircraft's stability parameters (time constants (T), damping ratios (ζ), and natural frequencies (ω)) rather than large amounts of tabulated data like most ground-based simulations.

Stability parameters form the basis for many of the flying qualities requirements for the United States military and for commercial airlines. They present a simple way to describe numerically the individual aspects of an aircraft's performance. For example, in the military specification MIL-F-8785C, [3] the Short Period's flying qualities are determined largely from the aircraft's Short Period natural frequency, damping ratio, and the ratio between load factor and angle of attack. In a non-linear model that is dependent upon the aircraft's geometry, it is very difficult to change an individual stability parameter while holding the rest constant because many parameters depend on the same physical properties of the aircraft (for example, the chord length). Within the TFSM, varying a single stability parameter does not affect any of the other parameters. The TFSM is therefore an ideal platform to use for flying qualities research since it is modeled using the very same stability parameters that describe an aircraft's flying qualities. However, using the TFSM does not remove the need for the full non-linear models normally used. Determining the new stability parameters requires the linearization of the non-linear model and the factorization of the resulting transfer functions. With the help of engineering tools like MATLAB, generating and updating the stability parameters in the TFSM is much quicker than replacing the data lookup tables and their interfaces in an existing simulation.

The United States Air Force (USAF) contracted Virginia Polytechnic Institute and State University (Virginia Tech) to do testing on Pilot-Induced Oscillations (PIOs) in early 2002. The USAF used the VISTA as the test aircraft by programming the Short Period and Phugoid modes to make the aircraft as stable or unstable as they wanted for a given test. [4] Therefore, the TFSM was the obvious choice for Virginia Tech's test model. Unfortunately, errors within the TFSM became apparent when the pilot experienced unnatural responses to commanded inputs during the flight tests. The most pronounced error was the inability to increase the aircraft's turn-rate during a banked turn. A less drastic, but just as disorienting, problem was the lateral-directional coupling experienced in the longitudinal response. Fixing these errors was essential in order to use the TFSM as an actual test bed for the PIO research.

Figure 1.1 – NF-16D Variable stability In-flight Simulator Test Aircraft (VISTA) [2]

Table 1.1 – Flight Envelope for the NF-16D VISTA [2]

Maximum Airspeed:	440 KIAS / 0.9 Mach
Angle-of-Attack Limits:	-10° to +16°
Normal Load Factor Limits:	-2.4 to +6.8 gs
Maximum Sideslip:	±10°

Table 1.2 – Examples of the VISTA's Variable Stability System Aircraft Dynamics Modeling Capabilities [2]

Parameter	**Minimum Value**	**Maximum Value**
Short Period Frequency, ω_{sp} (rad/sec)	0.5	14.7
Short Period Damping Ratio, ζ_{sp}	-0.05	3.2
Nz/a (g/rad)	-3.2	85
Dutch Roll Frequency, ω_{dr} (rad/sec)	1.2	7.9
Dutch Roll Damping Ratio, ζ_{dr}	-0.2	2.1
Dutch Roll Phi/Beta Ratio, $\lvert\phi/\beta\rvert$	0.9	14

The validation tools for the TFSM were the original non-linear models used to create the TFSM. The linear TFSM would be a correct representation of the original model when it matched its parent non-linear model within a desired amount of error. This thesis describes the approach used to identify and correct the existing TFSM and falls into six chapters as follows:

Chapter 2 presents the mathematics and process for linearizing a non-linear model and forming the state-space representations of the linearized transfer functions that form the basis for the linear variable stability TFSM.

Chapter 3 briefly discusses each version of the TFSM used during this research along with any proof models used for comparison.

Chapter 4 presents the non-linear F-16 model and the TFSM's results from the flight tests.

Chapter 5 contains the analysis of the results from Chapter 4.

Chapter 6 summarizes the work and contains the conclusions drawn from the research.

1.2 Research Objectives

The objective of this thesis is to investigate the lateral-directional coupling in longitudinal responses found in the TFSM. The TFSM will be compared to the non-linear basis model to give insight into where the TFSM simulation failed. Real-time piloted flight tests using the non-linear model and the TFSM will provide the data for the analysis.

Chapter 2

Linearization and the Development of the Transfer Function Simulation Model

2.1 Introduction

This chapter goes through the process used to linearize an airframe model and generate the state-space realizations for the transfer functions needed for a Transfer Function Simulation Model (TFSM).

2.2 Linearization and the State-Space Representation

Linearizing the original six degree-of-freedom non-linear aircraft model is the key to creating the linear variable stability TFSM. The motion of a rigid aircraft in a stationary atmosphere over a flat earth is described by 12 non-linear, coupled, first order, ordinary differential equations describe. [5] Grouping these into four sets of three equations each leads to:

Force equations:

$$\dot{U} = RV - QW - g\sin\theta + \frac{X}{m} \tag{2.1}$$

$$\dot{V} = -RU + PW + g\sin\phi\cos\theta + \frac{Y}{m} \tag{2.2}$$

$$\dot{W} = QU - PV + g\cos\phi\cos\theta + \frac{Z}{m} \tag{2.3}$$

Kinematic equations:

$$\dot{\phi} = P + \tan\theta\left(Q\sin\phi + R\cos\phi\right) \tag{2.4}$$

$$\dot{\theta} = Q\cos\phi - R\sin\phi \tag{2.5}$$

$$\dot{\psi} = \frac{Q\sin\phi + R\cos\phi}{\cos\theta} \tag{2.6}$$

Moment Equations:

$$\dot{P} = \frac{I_{zz}}{I_{xx}I_{zz} - I_{xz}^2}\left[L + I_{xz}PQ - \left(I_{zz} - I_{yy}\right)QR\right] + \frac{I_{xz}}{I_{xx}I_{zz} - I_{xz}^2}\left[N - I_{xz}QR - \left(I_{yy} - I_{xx}\right)PQ\right] \tag{2.7}$$

$$\dot{Q} = \frac{1}{I_{yy}}\left[M - \left(I_{xx} - I_{zz}\right)PR - I_{xz}\left(P^2 - R^2\right)\right] \tag{2.8}$$

$$\dot{R} = \frac{I_{xz}}{I_{xx}I_{zz} - I_{xz}^2}\left[L + I_{xz}PQ - \left(I_{zz} - I_{yy}\right)QR\right] + \frac{I_{xx}}{I_{xx}I_{zz} - I_{xz}^2}\left[N - I_{xz}QR - \left(I_{yy} - I_{xx}\right)PQ\right] \tag{2.9}$$

Navigation Equations:

$$\dot{p}_N = U\cos\theta\cos\psi + V\left(-\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi\right)$$

$$+ W\left(\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi\right) \tag{2.10}$$

$$\dot{p}_E = U\cos\theta\sin\psi + V\left(\cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi\right)$$

$$+ W\left(-\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi\right) \tag{2.11}$$

$$\dot{h} = U\sin\theta - V\sin\phi\cos\theta - W\cos\phi\cos\theta \tag{2.12}$$

Numerical integration of these equations is the basis for real-time flight simulation now that computers are fast enough to integrate the above differential equations in real-time. To do so, the integration algorithm must propagate a state x forward in discrete time steps. One of the simplest forms of numerical integration is the first-order Euler method, which approximates the differential $\frac{dx}{dt}$ with $\frac{\Delta x}{\Delta t}$. Therefore, the time derivative of the state, x, is the discrete change in x, Δx, calculated over the time step Δt. For sufficiently small time steps, this very simple integration scheme is very accurate. A more complex approach is the Runge-Kutta fourth-order integration algorithm, which requires four function evaluations per time step. This, in turn, allows the use of larger time steps to retain a given accuracy with respect to Euler's method. [6]

A Taylor Series expansion through the first-order terms will linearize the above non-linear ordinary differential equations. This expansion assumes small perturbations about a reference flight condition, so it is only accurate for small disturbances in the states and/or controls. Equation 2.13 is a sample first order Taylor Series expansion for n independent variables about a given reference condition (indicated by the "Ref" subscript). [5]

$$f\left(x_1, x_2, ..., x_n\right) \approx f\left(x_1, x_2, ..., x_n\right)_{\text{Ref}} + \left.\frac{\partial f}{\partial x_1}\right|_{\text{Ref}} \Delta x_1 + \left.\frac{\partial f}{\partial x_2}\right|_{\text{Ref}} \Delta x_2 + ... + \left.\frac{\partial f}{\partial x_n}\right|_{\text{Ref}} \Delta x_n + \underbrace{H.O.T.}_{0} \tag{2.13}$$

In equation 2.13, the partial derivatives with respect to the independent variables are assumed to be small. Therefore, the higher order terms (denoted by *H.O.T.*), which are combinations of the partial derivatives, are negligible and equated to zero. All the terms in the above equation are collected to one side to generalize the above equation so that after the linearization, the function evaluated at the reference condition will be zero. The next step is to expand this representation

to encompass a system of equations: the 12 equations of motion. In equation 2.14, let x be a vector that represents n states, let $\dot{x}$ be the time derivative of the state vector x, and let u be the m-dimensional control input vector. A system of n linear ordnary differential equations is created from a system of n states and m controls by assuming that the reference conditions are zero and by linearizing the original system about those reference conditions.

$$\mathrm{f}(\dot{x},x,u)=\left.\frac{\partial \mathrm{f}}{\partial \dot{x}}\right|_{\mathrm{Ref}}\Delta\dot{x}+\left.\frac{\partial \mathrm{f}}{\partial x}\right|_{\mathrm{Ref}}\Delta x+\ldots+\left.\frac{\partial \mathrm{f}}{\partial u}\right|_{\mathrm{Ref}}\Delta u=0 \tag{2.14}$$

In this equation, $\mathrm{f}(\dot{x},x,u)$ is an n-dimensional vector-valued function. The above partial derivatives evaluated at the reference condition are the Jacobian matrices that follow the form in equation 2.15, where v is a p-dimensional vector.

$$\frac{\partial \mathrm{f}}{\partial v}=\begin{bmatrix}\partial \mathrm{f}_1/\partial v_1 & \partial \mathrm{f}_1/\partial v_2 & \cdots & \partial \mathrm{f}_1/\partial v_p\\ \partial \mathrm{f}_2/\partial v_1 & \partial \mathrm{f}_2/\partial v_2 & \cdots & \partial \mathrm{f}_2/\partial v_p\\ \vdots & \vdots & \vdots & \vdots\\ \partial \mathrm{f}_n/\partial v_1 & \partial \mathrm{f}_n/\partial v_2 & \cdots & \partial \mathrm{f}_n/\partial v_p\end{bmatrix} \tag{2.15}$$

With respect to the linearized equations of motion, $\partial \mathrm{f}/\partial \dot{\mathrm{x}}$ and $\partial \mathrm{f}/\partial \mathrm{x}$ are square $n\times n$ matrices and $\partial \mathrm{f}/\partial u$ is a $n\times m$ matrix. Solving equation 2.14 for $\Delta\dot{x}$ yields:

$$\Delta\dot{x}=-\left[\frac{\partial \mathrm{f}}{\partial \dot{\mathrm{x}}}\right]_{\mathrm{Ref}}^{-1}\left\{\left[\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right]_{\mathrm{Ref}}\Delta x+\left[\frac{\partial \mathrm{f}}{\partial \mathrm{u}}\right]_{\mathrm{Ref}}\Delta u\right\} \tag{2.16}$$

This is normally written as:

$$\Delta\dot{x}=A\Delta x+B\Delta u \tag{2.17}$$

Where

$$A=-\left[\frac{\partial \mathrm{f}}{\partial \dot{\mathrm{x}}}\right]_{\mathrm{Ref}}^{-1}\left[\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right]_{\mathrm{Ref}}\ \text{and}\ B=-\left[\frac{\partial \mathrm{f}}{\partial \dot{\mathrm{x}}}\right]_{\mathrm{Ref}}^{-1}\left[\frac{\partial \mathrm{f}}{\partial \mathrm{u}}\right]_{\mathrm{Ref}} \tag{2.18}$$

The actual expansion of the 12 equations of motion leads to 12 new, linear, coupled, time-invariant, ordinary differential equations. The Jacobian matrices A and B are constant when linearized about a constant reference condition that makes equation 2.17 time-invariant.

The ideal pilot tasks for this model only contain small perturbations from the reference condition because of the Taylor Series expansion style linearization used.

State-space equations relate a system's states, inputs, and outputs together. When considering a linear system with the above definitions for the state (x) and control (u) vectors, and defining the p-dimensional output vector y, this leads to the following state-space equations:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.19}$$

$$y(t) = Cx(t) + Du(t) \tag{2.20}$$

The constant Jacobian matrices A, B, C, and D are the state, input, output, and the direct transmission matrices respectively. [7] The next section describes the process used to create transfer functions from these state-space equations that the TFMS requires.

2.3 Transfer Functions

Solving the Laplace transforms for the linearized, time-invariant, differential equations generate the aforementioned transfer functions. The benefit behind working in the Laplace domain over the time domain is that the derivative of a time function is a multiplication of the transform by a complex variable s. This leads to algebraic equations in the Laplace domain instead of differential equations in the time domain. [7] Taking the Laplace transformations of the state-space equations 2.19 and 2.20 leads to:

$$sX(s) - x(0) = AX(s) + BU(s) \tag{2.21}$$

$$Y(s) = CX(s) + DU(s) \tag{2.22}$$

Solving for $X(s)$ in the state equation (2.21) and inserting it into the output equation (2.22) results in the following two equations:

$$X(s) = (sI - A)^{-1}[x(0) + BU(s)] \tag{2.23}$$

$$Y(s) = C(sI - A)^{-1}[x(0) + BU(s)] + DU(s) \tag{2.24}$$

Let the transfer function matrix, *G(s)*, relate the system outputs, *Y(s)*, to the system inputs, *U(s)*. Each location, $g_{ij}(s)$, in the transfer function matrix contains a single-input, single-output (SISO) transfer function relating the j^{th} input to the i^{th} output. Therefore, equation 2.25 represents the SISO transfer functions in each position in the matrix after assuming that all the reference conditions, *x(0)*, are zero.

$$g_{ij}(s) = \frac{Y_i(s)}{U_j(s)} = c_i(sI - A)^{-1}b_j + d_{ij} \tag{2.25}$$

The transfer function matrix can be rewritten using index notation as:

$$G(s) = \left[g_{ij}(s)\right] \text{ for } i \in \{1,2,...p\} \text{ and } j \in \{1,2,...m\} \tag{2.26}$$

$$G(s) = \left[\frac{c_i adj(sI - A)b_j + d_{ij}|sI - A|}{|sI - A|}\right] \tag{2.27}$$

when the matrix inverse for $(sI - A)$ is replaced by its adjoint matrix divided by its determinant, as seen in equation 2.28.

$$H^{-1} = \frac{adj(H)}{|H|} \text{ for all square matrices } H \text{ where } |H| \neq 0 \tag{2.28}$$

Stevens and Lewis [6] define a transfer function's zeros to be the values for *s* that make the numerator vanish (the magnitude is zero). The transfer function's poles occur where the magnitude goes to infinity, or where the denominator becomes zero. As seen in equations 2.26 and 2.28, all the transfer functions that share a given state matrix will have the same poles. In some cases, poles and zeros may cancel each other out if a given mode does not exist in the response. As stated in Ogata, [7] the transfer function is a property of the system and completely independent of the input. Therefore, it is a valid way to approximate an airframe model around the reference conditions regardless of the control inputs used during a simulation. Transfer functions give insight into the system's nature by studying the system's response to various inputs.

2.4 TFSM Construction

Non-linear numerical aircraft models are the basis for the linear TFSMs. The first step in converting the non-linear model into a usable form is to linearize it. Subroutine *JACOB* and its supporting subroutines in the Stevens and Lewis F-16 simulation code found in Appendix C perform the required numerical linearization. Table 2.1 contains the reference wings level flight conditions (β and ϕ identically zero) used for all the models in this thesis. Within *LINEARIZE*, the longitudinal states (total velocity, angle of attack, pitch angle, and pitch-rate) and controls (throttle and elevator) were manually decoupled from the lateral-directional states (sideslip angle, roll angle, roll-rate, and yaw-rate) and controls (aileron and rudder) by linearizing each mode separately. Each control affects a certain set of states because the F-16 model's control surfaces effectively act in either the longitudinal or the lateral-directional mode, but not in both. As described above, each pair of controls and their corresponding states make up the longitudinal and lateral-directional modes for the F-16. The longitudinal states are present in the x-z body-axis plane whereas the lateral-directional generally states appear in the x-y body-axis plane. The lateral-directional states also include the aircraft's roll angle and roll-rate. The right-handed body-axis coordinate system in this thesis has its origin at the aircraft's cg; its x-axis points out the aircraft's nose, its y-axis points out the right wing, and its z-axis points down through the aircraft's plane of symmetry. *LINEARIZE* creates two unique uncoupled state-space systems when called for both modes and stores the each system's A, B, C, and D Jacobian matrices in a mode-specific output file. MATLAB internally recreates the state-space equations from the F-16's Jacobian matrices and then creates the transfer functions for the outputs with respect to the inputs as shown in Appendix A.

Table 2.1 – Reference Flight Conditions for the Stevens and Lewis F-16 Linearization

Total Velocity:	500 ft/sec
Altitude:	10000 ft
XCG Location (% Chord):	0.25
Angle of Attack:	3.788625°
Pitch Angle:	3.788625°
Heading Angle:	0.0°
Throttle (0-1):	0.1962366
Elevator Deflection:	-3.851317°
Aileron Deflection:	0.0°
Rudder Deflection:	0.0°

Factoring the transfer functions' numerators and denominators puts them in a form where the stability parameters are readily obvious. The stability parameters are the gain constants (K), time constants (T), and damping ratios (ζ) with their corresponding natural frequencies (ω_n). For aircraft dynamics, the zeros/poles of the factorization will appear in one of three forms: s, $(s+1/T)$, or $(s^2+2\zeta\omega_n s+\omega_n^2)$. Each gain, time constant, damping ratio, and natural frequency is associated with a mode described by the relationship between the input and output variables. For example, the time constant found in the transfer function numerator between the state alpha and the elevator input is the time constant for angle of attack due to elevator ($T_{\alpha\delta th}$). The 12 transfer functions relating the six states (total velocity, angle of attack, pitch-rate, sideslip angle, roll-rate, and yaw-rate) to the four controls (throttle, elevator, aileron, and rudder) are presented in equations 2.29 – 2.42. Chapter 3 addresses the state reduction from the original 12 discussed above to the six now used. The complete set of results for the transfer functions and their corresponding factorizations are found in Appendix A. In the following transfer functions for total velocity, angle of attack, pitch-rate, sideslip angle, roll-rate, and yaw-rate there is no restriction placed on the sign of any stability parameter.

$$DENOM_{long} = \left(s^2 + 2\zeta_{SP}\omega_{SP}s + \omega_{SP}^2\right)\left(s^2 + 2\zeta_{P}\omega_{P}s + \omega_{P}^2\right) \tag{2.29}$$

$$\frac{V_T(s)}{\delta_T(s)} = \frac{K_{U\delta_T}\left(s+\frac{1}{T_{U\delta_T}}\right)\left(s^2+2\zeta_{U\delta_T}\omega_{U\delta_T}s+\omega_{U\delta_T}^2\right)}{DENOM_{long}} \tag{2.30}$$

$$\frac{V_T(s)}{\delta_e(s)} = \frac{K_{U\delta_e}\left(s+\frac{1}{T_{U\delta_e}}\right)\left(s^2+2\zeta_{U\delta_e}\omega_{U\delta_e}s+\omega_{U\delta_e}^2\right)}{DENOM_{long}} \tag{2.31}$$

$$\frac{\alpha(s)}{\delta_T(s)} = \frac{K_{\alpha\delta_T}s\left(s+\frac{1}{T_{\alpha\delta_{T1}}}\right)\left(s+\frac{1}{T_{\alpha\delta_{T2}}}\right)}{DENOM_{long}} \tag{2.32}$$

$$\frac{\alpha(s)}{\delta_e(s)} = \frac{K_{\alpha\delta_e}\left(s+\frac{1}{T_{\alpha\delta_e}}\right)\left(s^2+2\zeta_{U\delta_e}\omega_{U\delta_e}s+\omega_{U\delta_e}^2\right)}{DENOM_{long}} \tag{2.33}$$

$$\frac{Q(s)}{\delta_T(s)} = \frac{K_{Q\delta_T}s\left(s+\frac{1}{T_{Q\delta_T}}\right)}{DENOM_{long}} \tag{2.34}$$

$$\frac{Q(s)}{\delta_e(s)} = \frac{K_{Q\delta_e}s\left(s+\frac{1}{T_{Q\delta_{e1}}}\right)\left(s+\frac{1}{T_{Q\delta_{e2}}}\right)}{DENOM_{long}} \tag{2.35}$$

The lateral-directional transfer functions are as follows:

$$DENOM_{lat-dir} = \left(s^2+2\zeta_{DR}\omega_{DR}s+\omega_{DR}^2\right)\left(s+\frac{1}{T_R}\right)\left(s+\frac{1}{T_S}\right) \tag{2.36}$$

$$\frac{\beta(s)}{\delta_a(s)} = \frac{K_{\beta\delta_a}\left(s+\frac{1}{T_{\beta\delta_a}}\right)\left(s^2+2\zeta_{\beta\delta_a}\omega_{\beta\delta_a}s+\omega_{\beta\delta_a}^2\right)}{DENOM_{lat-dir}} \tag{2.37}$$

$$\frac{\beta(s)}{\delta_r(s)} = \frac{K_{\beta\delta_r}\left(s+\frac{1}{T_{\beta\delta_{r1}}}\right)\left(s+\frac{1}{T_{\beta\delta_{r2}}}\right)\left(s+\frac{1}{T_{\beta\delta_{r3}}}\right)}{DENOM_{lat-dir}} \tag{2.38}$$

$$\frac{P(s)}{\delta_a(s)} = \frac{K_{P\delta_a}\left(s+\frac{1}{T_{P\delta_a}}\right)\left(s^2+2\zeta_{P\delta_a}\omega_{P\delta_a}s+\omega_{P\delta_a}^2\right)}{DENOM_{lat-dir}} \tag{2.39}$$

$$\frac{P(s)}{\delta_r(s)} = \frac{K_{P\delta_r}\left(s+\frac{1}{T_{P\delta_{r1}}}\right)\left(s+\frac{1}{T_{P\delta_{r2}}}\right)\left(s+\frac{1}{T_{P\delta_{r3}}}\right)}{DENOM_{lat-dir}} \tag{2.40}$$

$$\frac{R(s)}{\delta_a(s)} = \frac{K_{R\delta_a}\left(s+\frac{1}{T_{R\delta_a}}\right)\left(s^2+2\zeta_{R\delta_a}\omega_{R\delta_a}s+\omega_{R\delta_a}^2\right)}{DENOM_{lat-dir}} \tag{2.41}$$

$$\frac{R(s)}{\delta_r(s)} = \frac{K_{R\delta_r}\left(s+\frac{1}{T_{R\delta_r}}\right)\left(s^2+2\zeta_{R\delta_r}\omega_{R\delta_r}s+\omega_{R\delta_r}^2\right)}{DENOM_{lat-dir}} \tag{2.42}$$

A new set of state-space equations relating the new state variables s^3, s^2, s^1, and s^0 (termed x_1, x_2, x_3, and x_4) to the original control inputs and the aircraft states (outputs) is created from the above transfer functions. Each control input has its own set of four first-order ordinary differential equations that describes its state variables: x_1, x_2, x_3, and x_4. Therefore, with the new state-space representations, the TFSM integrates the set of four first-order ordinary differential equations for each of the four controls instead of the 12 equations of motion. MATLAB generated the new set of systems and placed them in a canonical form such that the negative of the transfer function denominator became the differential equation for the internal state x_1 with the addition of the control input and the other differential equations followed the form:

$$\dot{x}_i = x_{i-1} \qquad \text{where } i \in \{2,3,4\} \tag{2.43}$$

The Jacobian matrices, A and B, are common to all the longitudinal states because the longitudinal denominator contains the coefficients for the $\dot{x}_1$ equation and the other states follow

equation 2.43. An identical argument exists for the lateral-directional states. Equations 2.44 – 2.46 contain the state-space representation for the total velocity output state with respect to the throttle control and the new internal states x_i.

$$\{\dot{x}_{\delta T}(i)\} = \begin{bmatrix} C_{3long} & C_{2long} & C_{1long} & C_{0long} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \{x_{\delta T}(i)\} + \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \delta_T \qquad \text{for } i \in \{1,2,3,4\} \tag{2.44}$$

$$\text{where} \quad \begin{aligned} C_{3long} &= -2\left(\zeta_{sp}\omega_{sp} + \zeta_p\omega_p\right) \\ C_{2long} &= -\left(\omega_p^2 + 4\zeta_{sp}\omega_{sp}\zeta_p\omega_{sp} + \omega_{sp}^2\right) \\ C_{1long} &= -2\left(\zeta_{sp}\omega_{sp}\omega_p^2 + \zeta_p\omega_p\omega_{sp}^2\right) \\ C_{0long} &= -\omega_{sp}^2\omega_p^2 \end{aligned} \tag{2.45}$$

$$V_T(\bar{x}) = K_{U\delta_T}\left[x(1) + \left(2\zeta_{U\delta_T}\omega_{U\delta_T} + \frac{1}{T_{U\delta_T}}\right)x(2) + \left(\frac{2\zeta_{U\delta_T}\omega_{U\delta_T}}{T_{U\delta_T}} + \omega_{U\delta_T}^2\right)x(3) + \frac{\omega_{U\delta_T}^2}{T_{U\delta_T}}x(4)\right] \tag{2.46}$$

Equation 2.46 is only one of the components that generate the total velocity state. The other portion is calculated with respect to the elevator and has a similar composition. However, the coefficients and zeros in the velocity to elevator transfer function are different from those in equation 2.46 because the total velocity's responses to each control input are different. Similarly, each control input has a corresponding part in all the other six aircraft states. Obviously, if a control does not influence a state then its contribution to that state is zero. Appendix B contains the entire formation of the transfer functions and their state-space realizations, including numeric values.

The TFSM has sixteen linear ordinary differential equations to integrate rather than the 12 equations of motion. This increases the amount of integration, but it is essential to the TFSM's versatility. The four state-space systems allow the aircraft to be modeled by stability parameters alone, whereas the equations of motion require information about the aircraft's geometry (like wingspan and chord length). If the wingspan is changed in a regular simulation to affect the short period then all the other modes will be affected by the change as well. For example, in the linearization process described above, many of the values inside the A matrix will change with the wingspan and will therefore change most of the other stability parameters. On the other hand, the TFSM isolates each mode individually by representing an aircraft with its

stability parameters. The TFSM's short period mode can be manipulated explicitly by changing the values for its natural frequency and damping ratio without affecting the other modes. This is the TFSM's major benefit and reason behind its creation.

State-space systems corresponding to transfer functions only calculate six of the 12 aircraft states. The aircraft's other six states (its roll, pitch, and yaw angles, its north and east positions, and its altitude) are being integrated from the kinematic and navigation equations (equations 2.4 – 2.6 and 2.10 – 2.12). Depending on the TFSM version, a first-order Eulerian or fourth-order Runge-Kutta integration scheme calculates the remaining six states. Both methods perform almost identically with small enough simulation time steps. A simple dynamic inversion of the force and moment equations (equations 2.1 – 2.3 and 2.7 – 2.9) calculates the TFSM's body-axis forces and moments using the current aircraft states. *CASTLE* requires the six total body-axis forces and moments as inputs. It integrates them to get all of the aircraft states for the real-time simulator used during this investigation. The model is ready for testing now that it calculates the states, forces and moments at each time step.

Chapter 3

Version History for the Transfer Function Simulation Model

3.1 Introduction

This chapter details the various models used to isolate the problems within the Transfer Function Simulation Model (TFSM).

3.2 Simulation Environments

Two different development stations were used to implement the TFSM and the proof model desktop simulations. The *Phantom* simulation and its resulting linear variable stability TFSM FORTRAN simulations were constructed on a Silicon Graphics Origin 2000^{TM} Deskside System with dual 180 MHz CPUs with 256 MB of RAM and a 4 GB disk. [1] The F-16 models, both the Stevens and Lewis non-linear and the TFSM, were implemented on an Alienware Area-51M^{TM} with a 2.4 GHz CPU with 512 MB of RAM and a 40 GB disk. All desktop simulations except the non-linear F-4 model from the original TFSM were made compatible with *CASTLE* (Control Analysis and Simulation Test Loop Environment) for real-time simulation. Naval Air Warfare

Center's Manned Flight Simulator branch (MFS) developed *CASTLE* – a six degree of freedom non-linear aircraft simulation package. [8] The Virginia Tech Flight Simulation Laboratory implemented the *CASTLE* architecture on a modified 2F122A cantilevered, motion-based, three degree of freedom flight simulator. [9]

All simulations, real-time piloted or canned inputs, ran at 100 Hz collecting data at each time step. When running canned inputs, *CASTLE* used a set of recorded control deflections similar to the files used as inputs to the desktop simulations rather than stick commands from the cockpit. Piloted flights exhibiting the TFSM's problems were the basis for the time histories of control deflections. Otherwise, the control deflections followed simple mathematical functions like steps, ramps, sine and cosine functions. The cockpit models the United States Navy's A-6E *Intruder* aircraft flight controls, instruments, and flight systems. An electronic control loader produces the stick feel and can simulate any aircraft's stick response. For this thesis, it models the F/A-18 E/F's characteristics. Pilot headsets simulate the aircraft noise experienced during flight. The visual system, consisting of three downward projecting monitors, display the surrounding terrain during take-off, landing and normal flight maneuvers. [10] Terrain conditions include environments like Naval air stations, the CVN-68 *Nimitz* aircraft carrier, and the KA-6 Tanker aircraft. However, the calligraphic visuals only simulate dusk or nighttime conditions.

3.3 The Original (v3.0)

Henrik Pettersson designed and created the first version of the TFSM by linearizing a six-degree of freedom F-4 *Phantom* FORTRAN model and generating the required transfer functions as described in Chapter 2. The Naval Air Test Center (NATC) at Patuxent River, Maryland developed the non-linear *Phantom* model. The TFSM uses transfer function state-space representations to calculate the nine aircraft states that fully describe its orientation and flight path: total velocity, angle of attack, sideslip angle, roll angle, pitch angle, heading angle, roll-rate, pitch-rate, and yaw-rate. During each time step, the *RK4* subroutine (Appendix C.1.2) integrates the internal state-space representations' states. *RK4* also integrates the navigation equations (equations 2.10-12) to determine the aircraft's position.

This version's implementation in *CASTLE* was very hard for a pilot to control due to the model's extreme sensitivity to stick commands and poor lateral-directional characteristics present in the linearized F-4 model. The pilot's inability to control the model showed that the TFSM required improvements to permit a proper analysis of its validity. During the piloted tests, *CASTLE* stored all the aircraft states calculated by the transfer functions in a specified output file. However, the *CASTLE*'s output from the array, YOUT, had its array indices shifted – a given state's values were actually the values for the previous state in the array. For example, the output for the aircraft's angle of attack, YOUT(2), was actually the total velocity, YOUT(1). The aircraft state array was stored as a local variable in the transferfunction.f module used by *CASTLE* for the TFSM. If *CASTLE* internally shifted the index for local arrays, then fixing this problem would be a possible solution for the uncontrollable and possibly incorrect model.

3.4 Improved Lateral-Directional *Phantom* (v3.1)

This version is an improvement on the original simulation model. The local aircraft state array, YOUT, was made into a global, aircraft-specific variable to fix the index-shifting present in the outputted data. The new model's flight test output (using the same control inputs) showed that it performed identically to the previous version. Although the aircraft flew the same, changing the arrays fixed the index-shifting problem in *CASTLE*'s output. Therefore, it showed that *CASTLE* mishandled local variables only during its data output algorithms and not during the simulation itself. With this understanding, improving the model's flying qualities was the next step.

The approach used to increase the model's stability and performance was to change most of its lateral-directional natural frequencies, damping ratios, and time constants. For example, increasing the spiral mode time constant by a factor of twenty-eight made its response time more like the Phugoid mode rather than the Short Period mode. These modifications helped improve the model's controllability, enabling piloted flight that was worth analyzing. While testing the improved *Phantom* model, the pilot quickly noticed a major problem: during conventional (banked) turns, an elevator command only increased the pitch angle and not the turn rate. The assumption that the pitch-rate (Q) is equal to the pitch angle's rate of change with respect to time ($\dot{\theta}$) caused the problem in the elevator command's response. Obviously, once the aircraft is

away from straight and level flight, $\dot{\theta}$ no longer equals the pitch-rate. Equation 2.5 explicitly shows this flight characteristic for banked flight. Fixing the other less fundamental problems became the priority so that the rest of the model would behave properly when addressing the major problems.

During the initial flights, the model's pitch and heading angle drifted away from nominal conditions after small commanded inputs. Therefore, model following was implemented in the body-axis force and moment calculations to reduce the error in the transfer function linearization. The model following implementation compares the nine aircrafts states from the transfer functions to the same states that *CASTLE* generates by integrating the force, kinematic, and moment equations (equations 2.1-9) with the body-axis forces and moments it receives as inputs. When solved for the body-axis forces and moments, the additions to the TFSM's force and moment equations are as follows:

$$X = ... + m\lambda\left(V_{Ttf}\cos\alpha_{tf}\cos\beta_{tf} - V_{Tc}\cos\alpha_{c}\cos\beta_{c}\right) \tag{3.1}$$

$$Y = ... + m\lambda\left(V_{Ttf}\sin\beta_{tf} - V_{Tc}\sin\beta_{c}\right) \tag{3.2}$$

$$Z = ... + m\lambda\left(V_{Ttf}\sin\alpha_{tf}\cos\beta_{tf} - V_{Tc}\sin\alpha_{c}\cos\beta_{c}\right) \tag{3.3}$$

$$L = ... + \lambda\left(I_{xx}\left(P_{tf} - P_{c}\right) - I_{xz}\left(R_{tf} - R_{c}\right)\right) \tag{3.4}$$

$$M = ... + \lambda I_{yy}\left(Q_{tf} - Q_{c}\right) \tag{3.5}$$

$$N = ... + \lambda\left(-I_{xz}\left(P_{tf} - P_{c}\right) + I_{zz}\left(R_{tf} - R_{c}\right)\right) \tag{3.6}$$

The subscript 'tf' indicates the internal transfer function state and 'c' indicates a state from *CASTLE*. This model following method fixed the majority of the drift in pitch angle and sideslip angle, but the problem with the aircraft angles during a banked turn persisted. The model following implementation brought about the lateral-directional coupling in longitudinal responses. The pilot originally noticed this phenomenon when there was no response to throttle inputs after lateral-directional maneuvers. The model's flight test data showed that its velocity and pitch angle dropped unnaturally during the maneuver, which caused the body-axis x-directional force to grow more negative as the simulation continued.

3.5 Banked Turn *Phantom* (v4.0)

The realization that during most flight conditions the body-axis pitch-rate did not equal the pitch angle's time derivative showed that the transfer functions' state-space representation for the three aircraft angles (roll, pitch, and yaw) had to be removed. Therefore, code was added to calculate the missing angles by integrating the kinematic equations (equations 2.4 – 2.6). To save computation time during simulation runs, the TFSM used the aircraft angles calculated by *CASTLE* during its integration of the body-axis forces and moments. The TFSM also stored *CASTLE*'s values for the aircraft's center of gravity position (north, east, and altitude) in the simulation world as its own instead of integrating the navigation equations. Using *CASTLE*'s values removed redundancy from the simulation since both the model and *CASTLE* performed similar operations to calculate the aircraft angles and the aircraft's center of gravity position. Integrating the kinematic equations corrected the aircraft's basic behavior during conventional turns; it removed the large changes in pitch angle instead of turn rate during banked turns. However, pointing the aircraft during the turns became difficult due to the limited response to longitudinal stick inputs. Large longitudinal stick deflections barely increased the aircraft's turn rate, and the lateral stick control was still very sensitive during all flight conditions.

The biggest problem during testing arose in the simulation's visual representation in the cockpit. The object of the flight test was to stay in formation flight with another aircraft slightly ahead of the piloted aircraft (Figure 3.1 (a)). During the flights, the visuals indicated that the pilot was located under the aircraft's center of gravity rather than level with and ahead of it. The correct response is for a pure rotation of the visuals about the x-axis as shown in Figure 3.1 (b). However, when initiating a bank angle, the visuals showed that the aircraft was swinging out to the side and banking in reference to the formation aircraft directly ahead of it. Figure 3.1 (c) shows the lateral shift in the visuals with rotation about the x-axis. The problem seemed to be in the pilot's location relative to the aircraft's center of gravity. However, the pilot and center of gravity location variables in *CASTLE* were correct. Cursory looks through the code did not yield any indication as to what was causing the unnatural flight behavior.

Similar to the previous *Phantom* model, the x-directional body-axis force grew negatively after lateral-directional maneuvers. This observation indicates that the model following implementation itself was not the reason for the lateral-directional coupling with the longitudinal

response. The coupling's cause was the integration of the kinematic equations to obtain the aircraft angles. As seen in Equation 2.5, the pitch angle's time derivative is an explicit function of pitch-rate, yaw-rate and bank angle that brings the lateral-directional states into the longitudinal states. The original TFSM did not have this problem because its longitudinal and lateral-directional states were completely uncoupled. Although the coupling was absent, the model was incorrect with the assumption that the aircraft's pitch-rate was the pitch angle's time derivative.

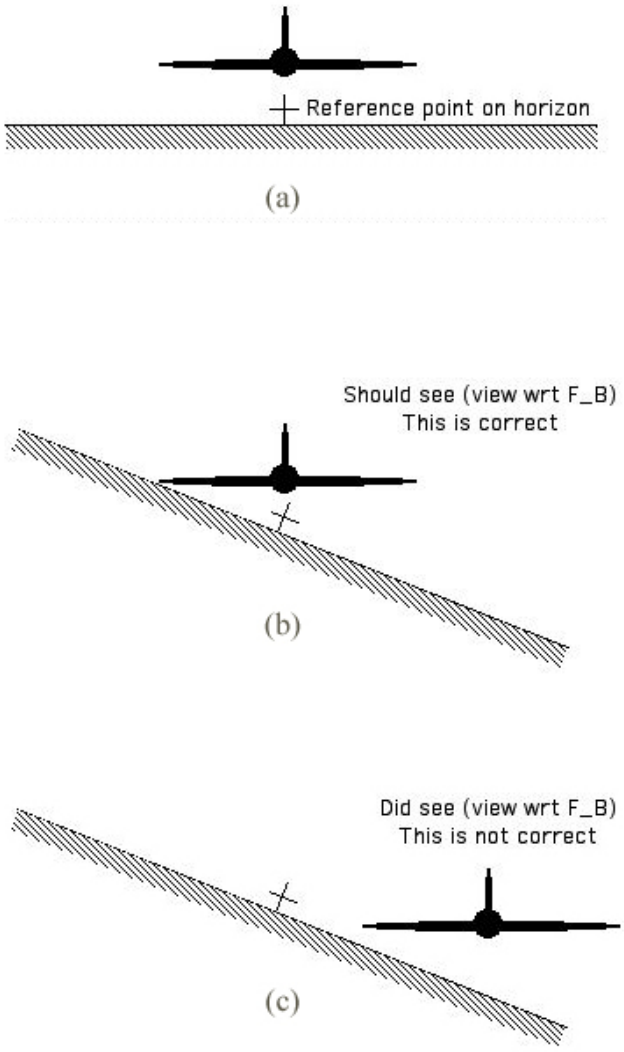

Figure 3.1 – Visual representation for a commanded bank angle in the Banked Turn *Phantom*

3.6 Stevens and Lewis F-16 Model

In order to insure that the newest model implementation actually performed unnaturally, the TFSM's performance was compared to a proven benchmark model. The chosen proof model was the non-linear six degree of freedom F-16 simulation in Stevens and Lewis [6]. The simulation program on the desktop and *CASTLE*'s real-time flight control algorithm called the single FORTRAN file containing the F-16 code presented in this thesis (Appendix C). The desktop version's simulation program lets the user access the subroutines that will run a simulation, trim the aircraft at a given reference flight condition (*TRIMMER*), and to linearize the non-linear model (*LINEARIZE*). Subroutine *F* does the majority of the work for these functions, which calculates all the body-axis forces, moments, and the time derivatives for each of the 12 aircraft states at each time step. Then the chosen integration method integrates the state derivatives. This simulation uses a Runge-Kutta fourth-order method. The control subroutine, Control.f, which was created for *CASTLE*, also calls subroutine *F* to calculate the body-axis forces and moments that *CASTLE* requires. Both simulation versions performed identically for a given set of control inputs. The time histories and pilot comments for the piloted flight and the time histories of the subsequent desktop simulation for the same control inputs showed that the non-linear F-16 simulation behaved like an actual aircraft. The simulation obeyed the 12 equations of motion given in Section 2.2 for the given control inputs and was used as the standard for the TFSM.

3.7 Original F-16 TFSM (v4.2)

This version of the TFSM uses the same methodology as the Banked Turn *Phantom* version. Even though the approach was the same, this model was built from the ground up because of the problem with the visuals in the previous version. The verified F-16 model already in *CASTLE* provided the basis for the TFSM *CASTLE* modules. The TFSM's data and algorithms replaced the aircraft specific data within the F-16's modules. The three aircraft angles were calculated by

CASTLE when it integrates the force, moment, and kinematic equations. Likewise, the TFSM took the aircraft's position from *CASTLE*'s integration of the navigation equations. State-space representations for total velocity, angle of attack, sideslip angle, roll-rate, pitch-rate, and yaw-rate, obtained from the Stevens and Lewis Desktop F-16 were still used to calculate the six aircraft states. Although the majority of the transfer functions had the same form (same number and type of zeros) as their counterparts in the *Phantom* version, there were significant differences between the state-space representations for the F-16 and the F-4 models. Obviously, the changes in the transfer functions caused the values for the gains, time constants, damping ratios, and natural frequencies to change.

Flying this model in real-time was very successful. The model was very responsive to pilot inputs but not sensitive enough to make it hard to control. Recoding the model from the working F-16 modules with the new linearization data fixed the problem with the visual misrepresentation. Although this version was much easier to fly in real-time, the longitudinal response problem persisted. Unfortunately, the TFSM did not match the data generated by the Stevens and Lewis F-16 Model for the same control input.

3.8 No *CASTLE* F-16 TFSM (v5.0 Old)

CASTLE's inability to output data correctly from arrays raised doubts about its ability to process any variable arrays correctly and its accuracy in performing the required integration to determine the aircraft angles and position. Deleting *CASTLE* from the simulation process removed the possibility of it causing the problems with the TFSM's velocity and pitch-angle. Individual variables replaced the global arrays previously used. *CASTLE* still controlled these variables but it was the first step needed to isolate the simulation from *CASTLE*. This was also a way to determine if *CASTLE* mishandled arrays during the simulation. Since this was the only change made to the previous version, comparing its flight test results to those from the previous version helped determine *CASTLE*'s ability to use arrays. As stated during the description in Section 3.2, *CASTLE* correctly uses arrays internally.

The next step was to let the TFSM integrate the kinematic and navigation equations itself, without obtaining them from *CASTLE*. The Runge-Kutta fourth order integration module used

previously uses arrays as its input/output variables; therefore, it would not be an efficient way to integrate individual variables. A simple first-order Euler integration scheme now integrated all of the internal state-space states as well as the kinematic and navigation equations. Simplifying from a fourth-order method to a first-order method may have caused a loss in integration accuracy. *CASTLE* ran two models, identical except for their integration type, with the same input time histories and time steps to determine the accuracy of the first-order method with respect to RK4. The results from these tests showed that the first-order method matched the RK4 method almost identically during real-time flight with a time step of 0.01 seconds. Another set of tests was run with *CASTLE* comparing the integration of the kinematic and navigation equations using the Eulerian integration model and *CASTLE*'s own first-order Adams-Bashforth method. Once again, there was no loss in the accuracy between the integration schemes. Even with these changes, the longitudinal response problem still existed.

3.9 Final F-16 TFSM (v5.0)

This version is almost identical to the one presented in Section 3.8 above. The original conceptual design of the TFSM assumed that pitch-rate was equal to the pitch angle's time derivative during all flight conditions. That is not a valid assumption away from wings-level flight. However, the transfer functions for the angular rates from each control should be the derivatives of the transfer functions from aircraft angles to each of the controls respectively. To double check that this is indeed true, the Stevens and Lewis F-16 Desktop Model was re-linearized with respect the angular rates instead of the aircraft angles, which are no longer being calculated with state-space representations. The variable names in the model changed to reflect the new linearization scheme. There were no changes in the actual transfer functions when linearizing with respect to the angles or to the rates; because this TFSM version calculated the rates, it used variables that corresponded to the rates rather than the angles for the state-space representations (for instance, time constant for aileron to the roll terms was $T_{P\delta a}$ instead of $T_{\phi\delta a}$). As expected, with no change in the transfer functions themselves, the simulation remained the same – still flawed.

Chapter 4

Simulation Results

4.1 Introduction

Before any flight tests, the F-16 Transfer Function Simulation Model (TFSM) underwent control impulse testing similar to the initial verification testing done by Pettersson on the original TFSM [1]. The control impulses consisted of deflecting the control surfaces and the throttle setting to -10 degrees and 0.7 respectively, for the simulation's first 0.01 seconds before returning them to their trim settings. A 60-second simulation run tested the impulse responses individually. The TFSM's responses matched the Stevens and Lewis' F-16 model almost identically. This verification is not included in this thesis because it is not relevant to this investigation beyond making sure that the TFSM is modeling the F-16.

This chapter presents the results from two flight tests in which the pilot experienced the lateral-directional coupling in the longitudinal responses. The following two sections contain the data collected during the two flight tests. Each flight test section also contains the results from the test when run with either longitudinal or lateral-direction controls to help isolate the problem. In this chapter's plots, the solid line represents the TFSM and the dotted line represents the non-linear F-16 simulation. The first test, called an *S*-turn, is similar to the flight that initially illustrated the problem in the velocity response to lateral-directional states. In this maneuver, the pilot turns the aircraft to the right and then back to the left before straightening out the flight

path. The test pilot flew the Final F-16 TFSM's *CASTLE* implementation to generate the required control inputs for the *S*-turn flight maneuver. Formation flight was the second flight test. The piloted aircraft started 500 ft behind and 100 ft below a KA-6 Tanker aircraft and the pilot tried to close the distance between the two aircraft and then maintain position with the Tanker. The formation flight test used the Stevens and Lewis F-16 *CASTLE* implementation to generate the required time histories for the control inputs. *CASTLE* read in the time histories as canned inputs and sent the corresponding controls to the Final F-16 TFSM (Section 3.9) for its formation flight simulation run. All of the flight tests' reference conditions were the same as the reference conditions used to linearize the TFSM (Appendix B.2). During the flight tests, the sideslip angle's magnitude was always less than one degree and in most cases, it was less than 0.1 degrees. In addition, the general shapes of the plots were the same for the two models during any given simulation. Therefore, the analysis ignores the effects from the sideslip angle; the only time history for the sideslip angle presented in this chapter is in Figure 4.1. This time history illustrates the two points made above: the TFSM's response had the same shape as the F-16's response, and the sideslip angle's magnitude was very small relative to the other angles.

The last two sections present one major problem in the TFSM: the absence of gravity. Section 4.4 contains the results from a simulation run with a 60-degree bank angle added to the initial conditions used by the two previous flight tests, and no control inputs. Section 4.5 presents the simulation results for the same flight conditions as Section 4.4, but this model calculated its angular rates with gravity terms included.

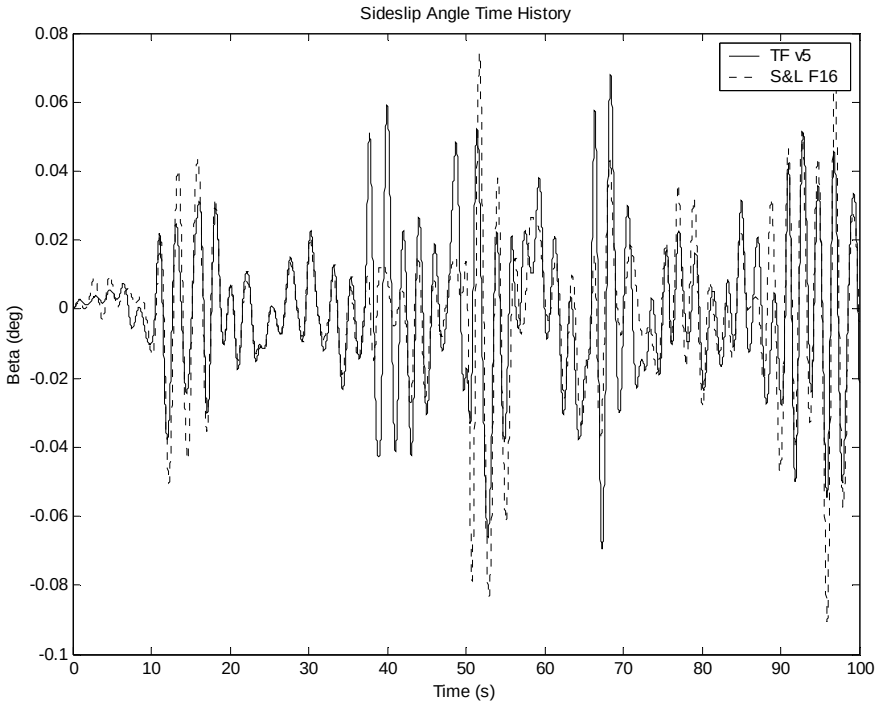

Figure 4.1 – Time History of Sideslip Angle for the Formation Flight Test ($\delta r = 0.0°$)

4.2 *S*-turn Maneuver

4.2.1 Control Inputs

Figures 4.2 and 4.3 are the control deflection time histories used in the *S*-turn flight test. The aileron, elevator, and rudder deflections are in degrees and the throttle deflection is a percentage between zero and one. Note that the rudder deflection sat at -1 degrees, but the test pilot did not utilize the rudder pedals during the simulation. The small deflection and noise existed because the flight simulator's rudder pedals had a small, non-zero nominal voltage that the real-time control algorithm converted into a minus one degree rudder deflection.

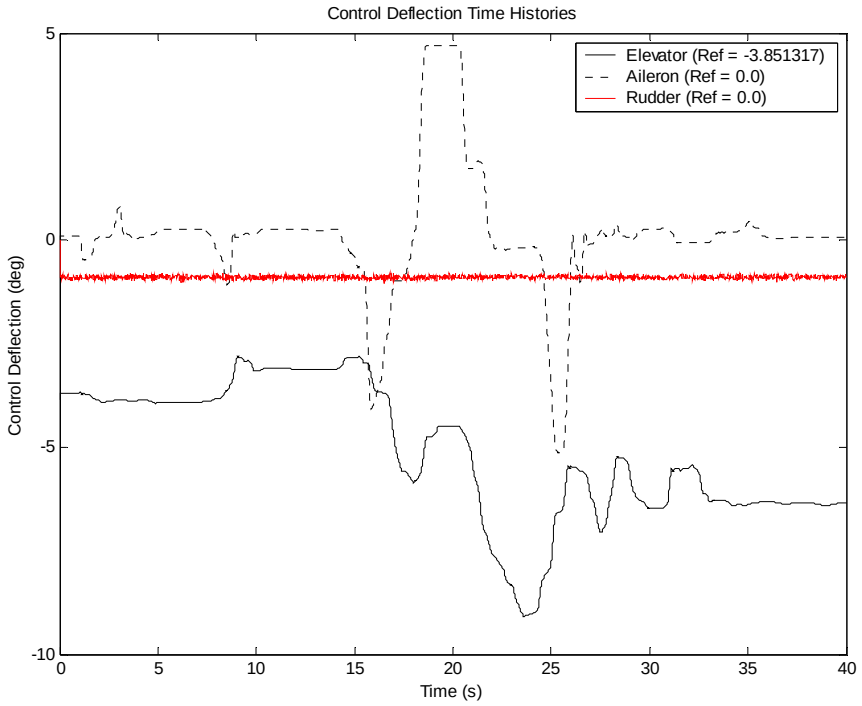

Figure 4.2 – Time History of Control Surface Deflections for the *S*-turn Maneuver

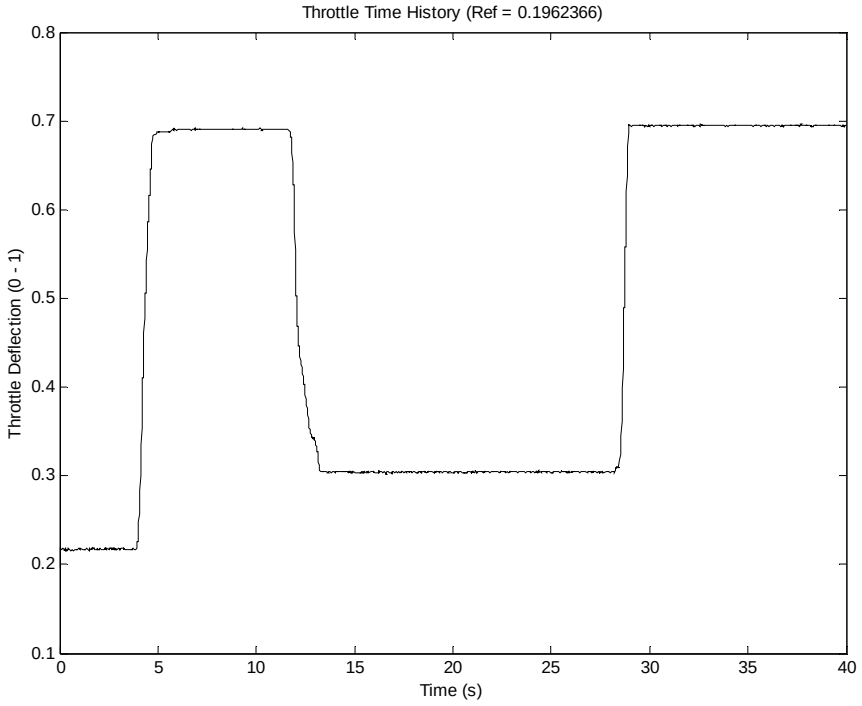

Figure 4.3 – Time History of the Throttle Deflection for the *S*-turn Maneuver

4.2.2 Full Controls

The following 10 figures present the time histories for the aircraft states when the canned inputs contained all four controls. As seen in Figure 4.4, the TFSM's velocity did not behave as desired and fell off rapidly after the initiation of the lateral-directional maneuver. Similarly, the TFSM's angle of attack (Figure 4.5) dropped between 17 and 27 seconds leaving the angle of attack with a negative one degree offset after the roll maneuver. Time histories for the three body-axis angular rates are in Figures 4.6 – 4.8. The roll-rate (Figure 4.6) and the pitch-rate (Figure 4.7) for the TFSM behaved very well with respect to the F-16 model, except when there were large changes in the control deflections from the reference settings. At these points, the TFSM's rates could not match the F-16's rates at their peaks. There was a 10 degree/sec difference in the roll-rate at 17 seconds and 20 seconds. Similarly, the TFSM's pitch-rate was two degrees/sec less at 18 seconds; then, when the elevators returned closer to their trim setting, the pitch-rate behaved more desirably and followed the F-16, generating a permanent offset between the two. Between 22 and 25 seconds, the process repeated itself and increased the offset to three degrees/sec by the end of the simulation. Unlike the other two angular rates, the yaw-rate only behaved correctly for the first 20 seconds of the simulation. After the initiation of the large aileron deflections, the linear approximation for R broke down and it started to lag behind the F-16 with a much smaller magnitude as well.

Figures 4.9 through 4.11 contain the aircraft roll, pitch, and yaw angles respectively. The TFSM integrated these angles using the kinematic equations instead of having state-space systems directly calculate the new states. Therefore, any error in the states calculated by state-space systems was present in these variables as well. The TFSM's roll angle was consistent with the F-16 simulation until the initiation of the large aileron deflections at 15 seconds, and then the linear model could not quite follow the F-16 simulation to its peak values. There was approximately a 10-degree difference at the roll peaks during the S-turn; after completing the maneuver, the TFSM's error had built up causing the slight discrepancy between the models after 27 seconds. As stated above, the TFSM velocity's behavior was not the same as the F-16's; likewise, the pitch angle response was different. During the large roll angles, the pitch angle dropped off and became negative before it pushed slowly positive for the rest of the simulation. Similar to the roll angle, the linear model's yaw angle response (Figure 4.11) could not handle the large inputs and missed the yaw angle's peak value at 25 seconds by 10 degrees.

TFSM also integrated the aircraft's position. As for the angles mentioned above, the position also inherited the errors in the velocity, angle of attack, sideslip angle, and angular rates during their integration. The navigation equations show that altitude (Figure 4.12) depended most on the pitch angle and velocity components. With this in mind, it is easy to see why the TFSM loses altitude rather than climbing with the F-16 by looking at the navigation equations. Similarly, the TFSM did not travel as far east as the F-16 (Figure 4.13) because the TFSM's yaw angle never reached the higher values.

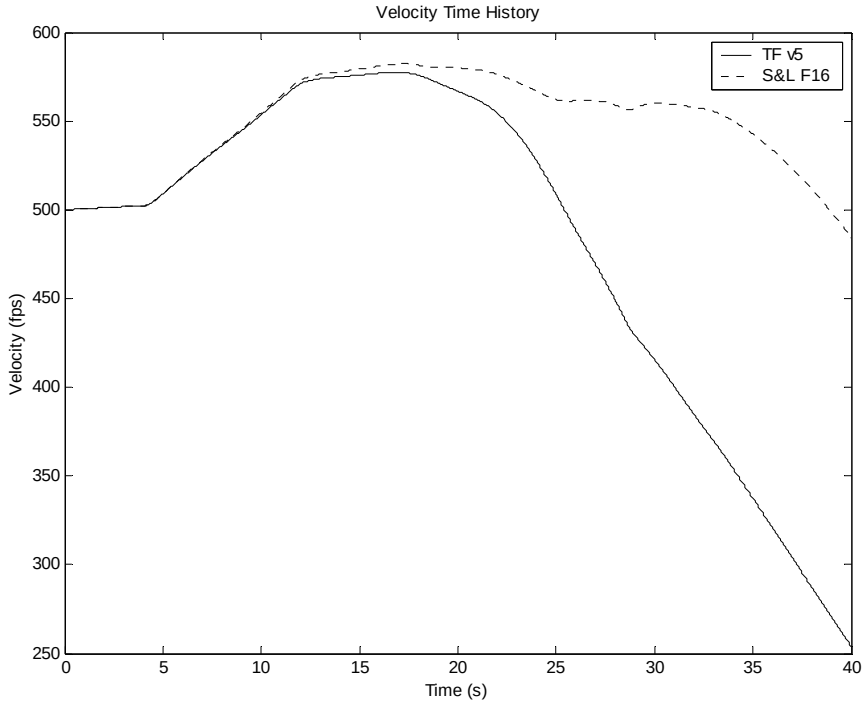

Figure 4.4 – Time History of Total Velocity for the *S*-turn Maneuver

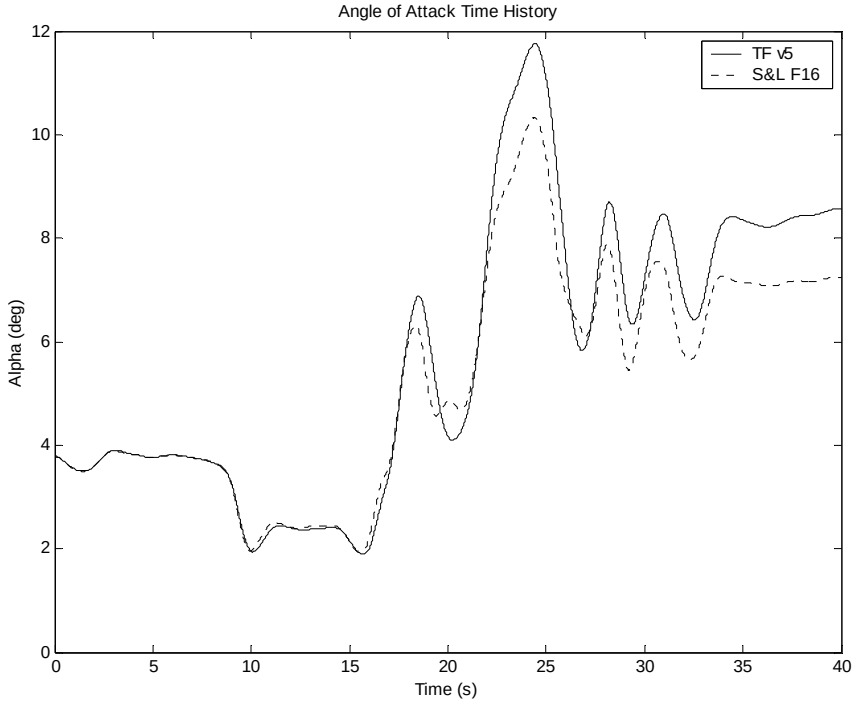

Figure 4.5 – Time History of Angle of Attack for the *S*-turn Maneuver

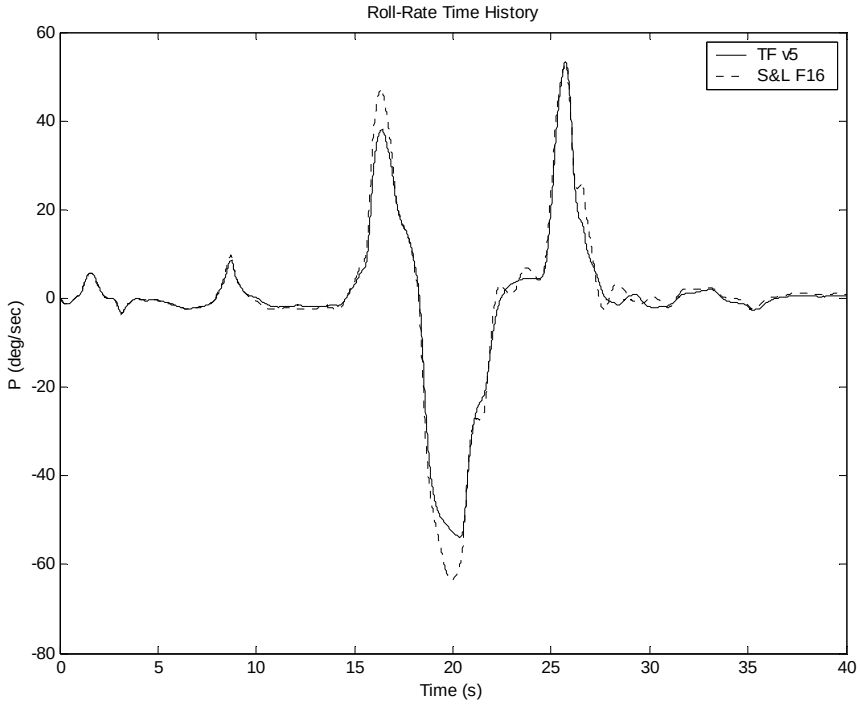

Figure 4.6 – Time History of Roll-Rate for the *S*-turn Maneuver

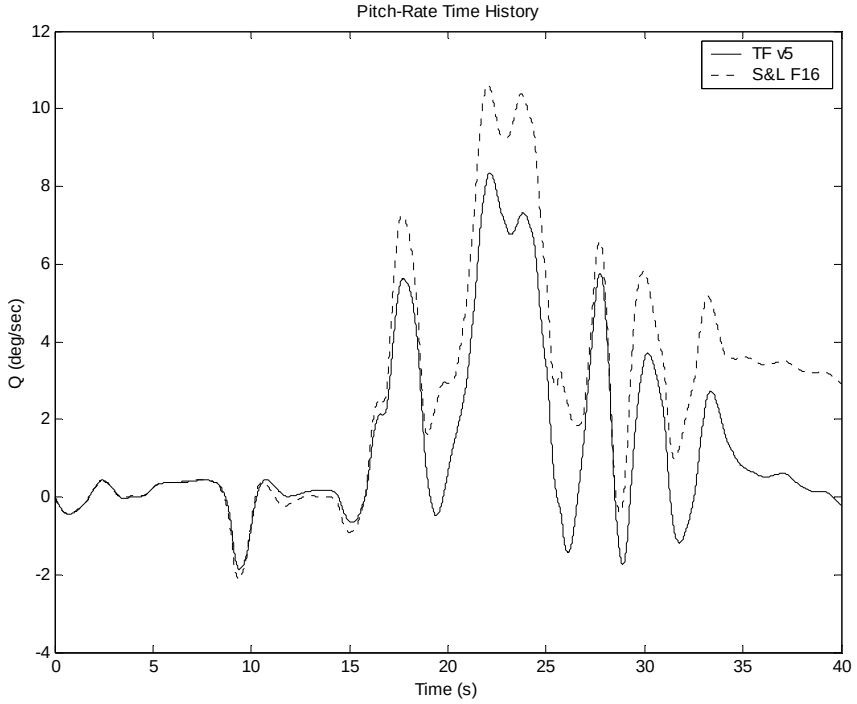

Figure 4.7 – Time History of Pitch-Rate for the *S*-turn Maneuver

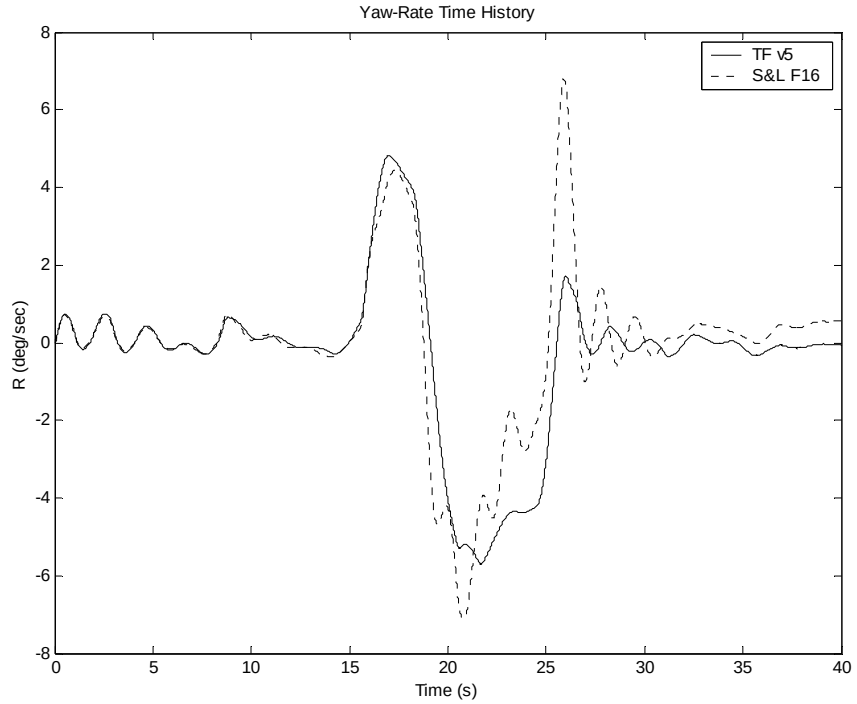

Figure 4.8 – Time History of Yaw-Rate for the *S*-turn Maneuver

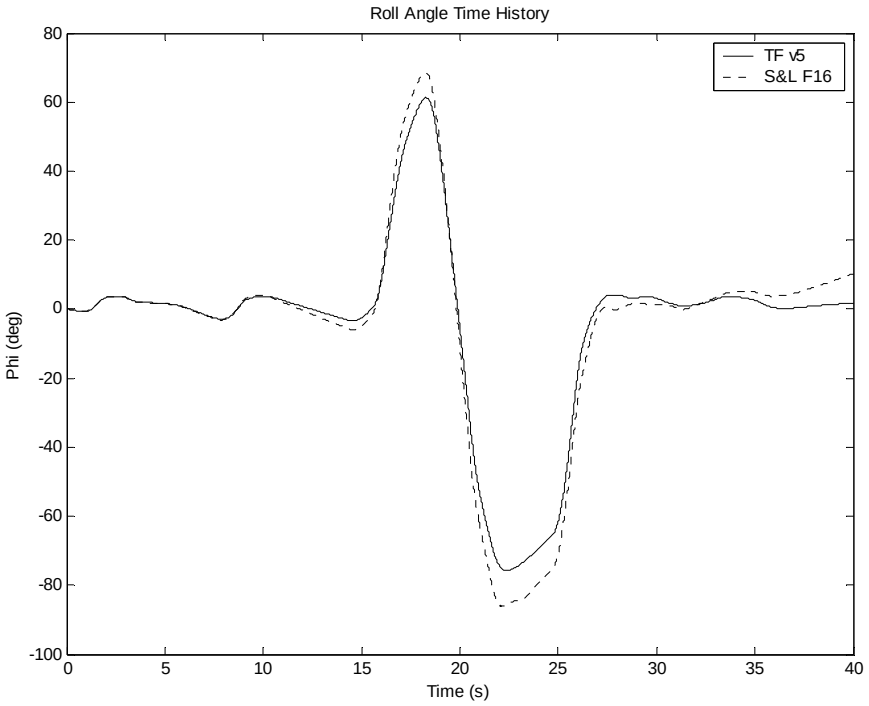

Figure 4.9 – Time History of Roll Angle for the *S*-turn Maneuver

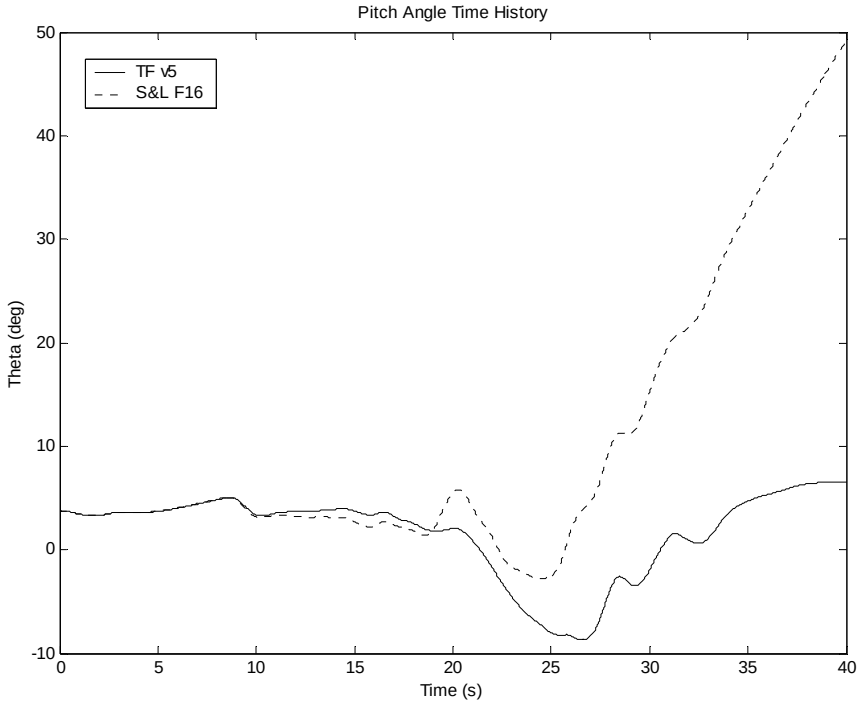

Figure 4.10 – Time History of Pitch Angle for the *S*-turn Maneuver

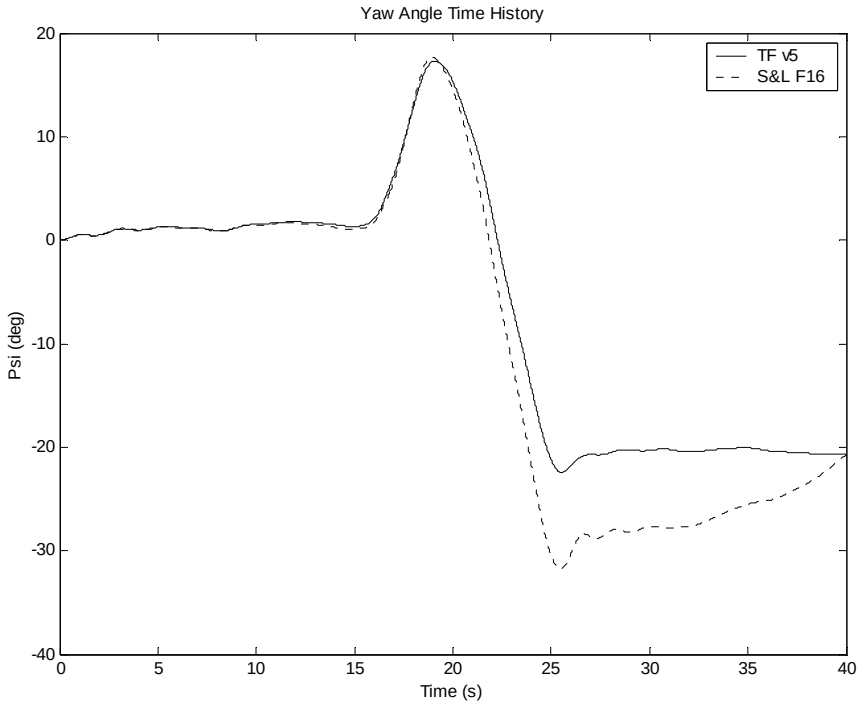

Figure 4.11 – Time History of Yaw Angle for the *S*-turn Maneuver

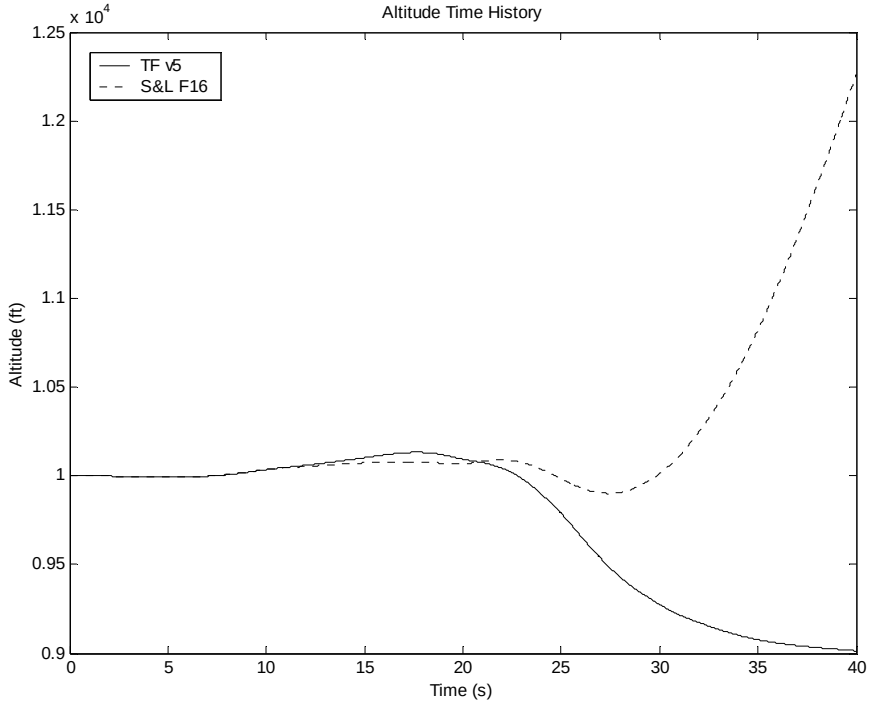

Figure 4.12 – Time History of Altitude for the *S*-turn Maneuver

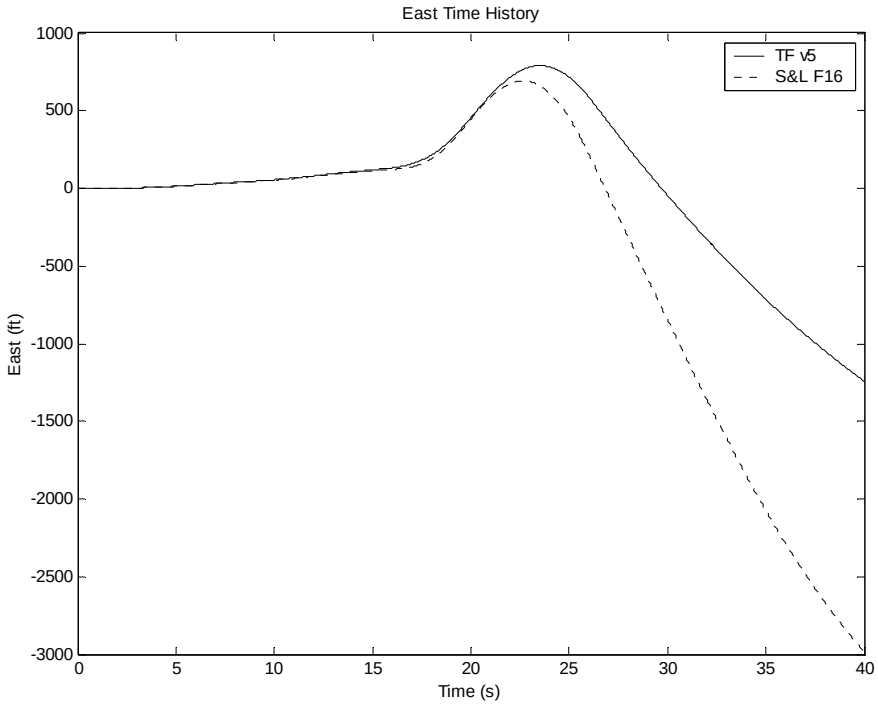

Figure 4.13 – Time History of East Position for the *S*-turn Maneuver

4.2.3 Longitudinal Controls Only

Setting the lateral-directional controls to their trim values of zero isolated the longitudinal responses to longitudinal controls. The lateral-directional responses in the F-16 simulation to the longitudinal controls were negligible because they were at least two orders of magnitude less than their longitudinal counterparts, and approximately zero. When the TFSM was constructed, the longitudinal and lateral-directional states were uncoupled, so all of its lateral-directional responses to longitudinal controls were zero. Therefore, there are no time histories for lateral-directional states in this section.

Unlike the complete *S*-turn simulation, the TFSM's simulation with only longitudinal controls behaved very well even for the larger control inputs. The largest error in the velocity time history (Figure 4.14) between the TFSM and the F-16 corresponded to the greatest throttle deflection. However, the error was within acceptable limits for the linearized model. The angle of attack and pitch-rate responses, Figures 4.15 and 4.16 respectively, matched very well with small differences generated at 10 seconds, when the elevator deflections started changing more dramatically. Even with the large control deflections the TFSM's angular responses stayed close

to the F-16's throughout the simulation. Following equation 2.5, the time derivative of the pitch angle is the body-axis pitch-rate when the roll angle is zero. Comparing the errors in the pitch-rate response and the pitch angle response (Figure 4.17) shows that the error in pitch-rate transfers directly to the pitch angle. The errors in the other states combined to make the TFSM's altitude trajectory lag slightly behind the F-16 (Figure 4.18).

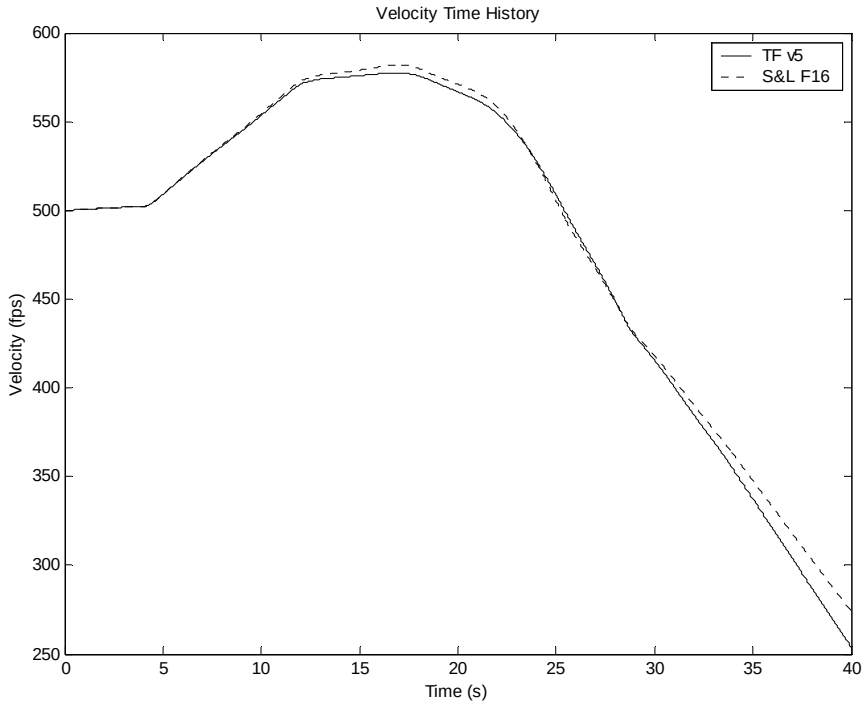

Figure 4.14 – Time History of Total Velocity for the *S*-turn's Longitudinal Controls

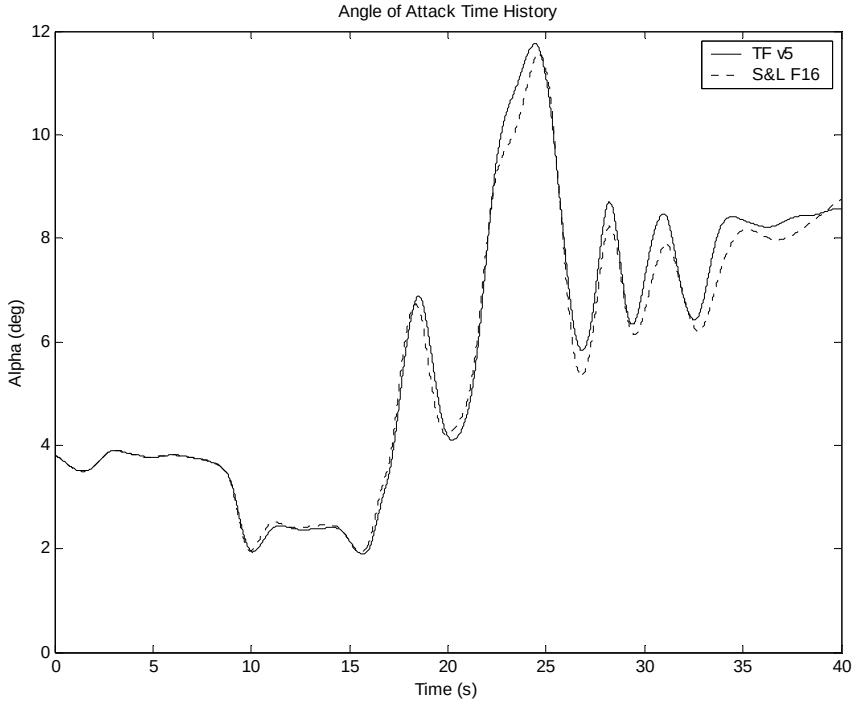

Figure 4.15 – Time History of Angle of Attack for the *S*-turn's Longitudinal Controls

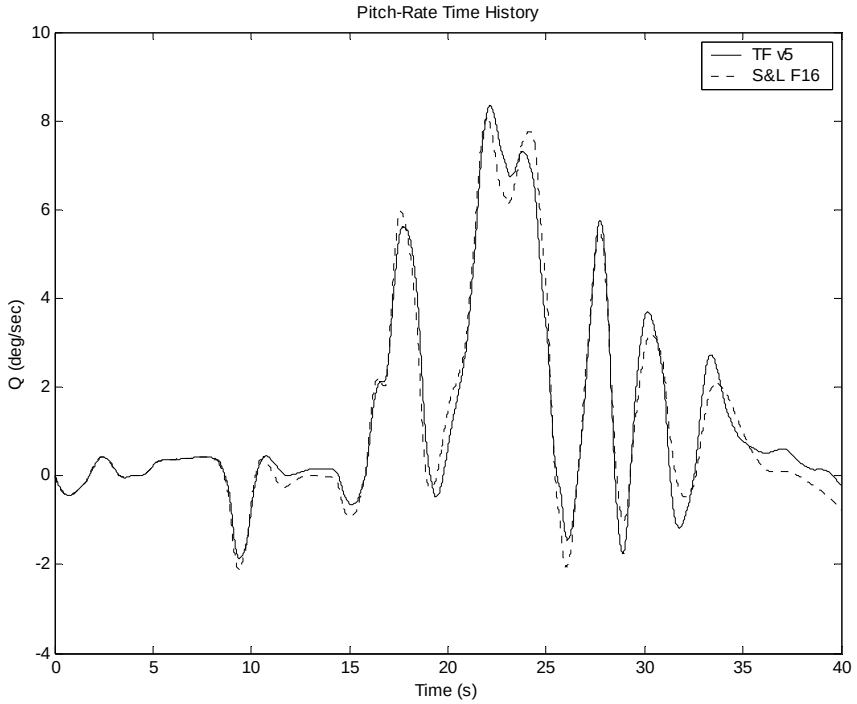

Figure 4.16 – Time History of Pitch-Rate for the *S*-turn's Longitudinal Controls

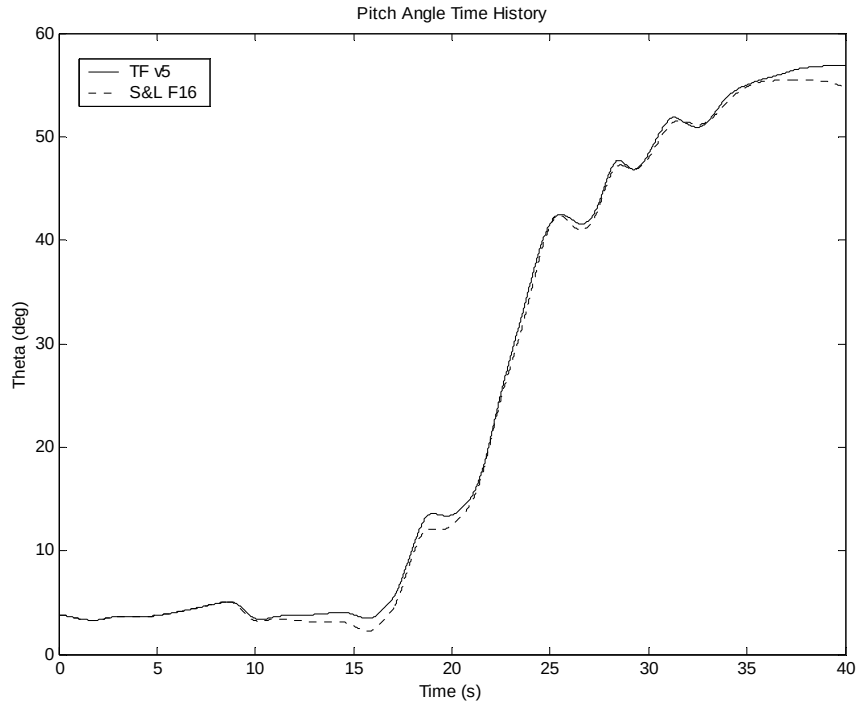

Figure 4.17 – Time History of Pitch Angle for the *S*-turn's Longitudinal Controls

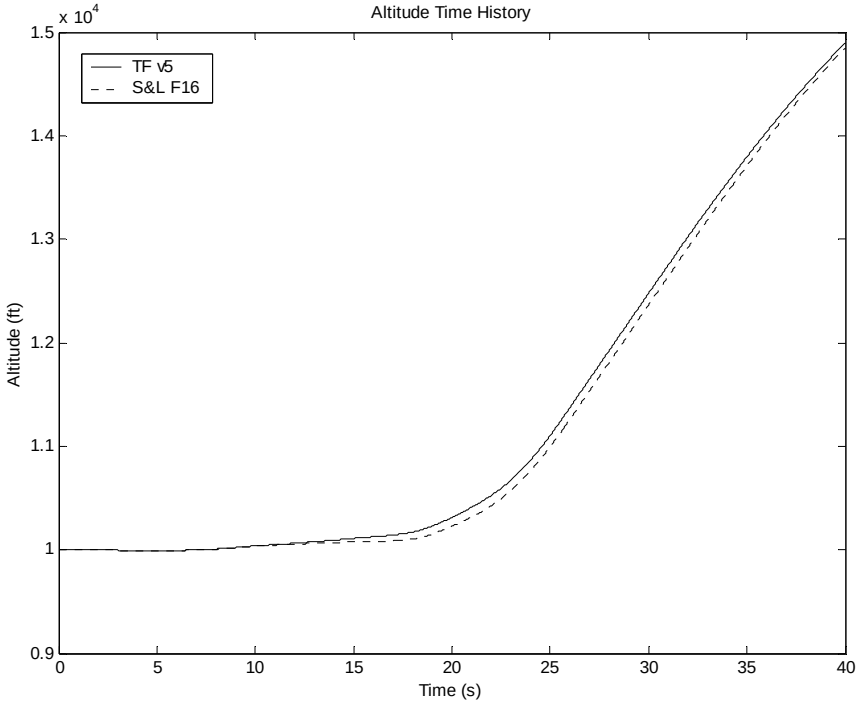

Figure 4.18 – Time History of Altitude for the *S*-turn's Longitudinal Controls

4.2.4 Lateral-Directional Controls Only

Similar to the longitudinal control isolation, this section contains the results from the aileron and rudder control inputs. It was necessary to include the longitudinal states in this section because the two models had very different results. Throughout the entire simulation, the elevator and throttle controls equaled their trim settings from Appendix B. This way, with no changes in the lateral-directional controls, the aircraft would remain trimmed. The largest error in this simulation was in the longitudinal states: velocity, angle of attack, pitch-rate, pitch angle, and altitude (Figures 4.19 – 4.23). Because the TFSM's longitudinal and lateral-directional states and controls were decoupled, it could not approximate the change in velocity due to the flight dynamics when there were no longitudinal controls. Both models should simulate the nose dropping during the roll doublet (seen in Figures 4.24 and 4.25), which would cause the aircraft to lose altitude and pick up speed, triggering a Phugoid mode type of response. However, the TFSM's velocity, angle of attack, and pitch-rate remained zero because of their state-space representations. The pitch angle and altitude changed because their time derivatives depended on the lateral-directional angles and rates, as well as the longitudinal ones. The pitch angle's and altitude's behaviors were undesirable because the other longitudinal variables did not change in a similar fashion. Hence, the TFSM continued to dive instead of pulling out of the dive by converting its potential energy into kinetic energy.

Without the longitudinal controls, the models' roll responses were almost identical. Figures 4.24 and 4.25 present the time histories for the roll-rate and angle. In both figures, the TFSM matched the F-16 perfectly until very small errors immerged after 20 seconds with the large aileron deflections. Once again, the linearized yaw responses failed to match the non-linear model for the large control deflections (see Section 4.2.3 and Figures 4.26 and 4.27). The variation in the yaw responses generated the differences in the east trajectories (Figure 4.28).

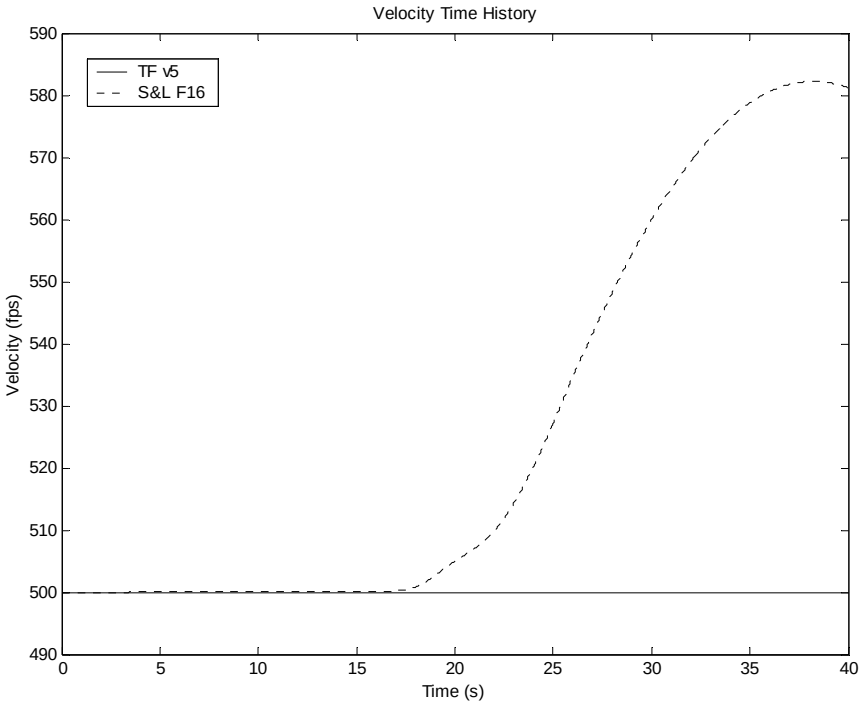

Figure 4.19 – Time History of Total Velocity for the *S*-turn's Lateral-Directional Controls

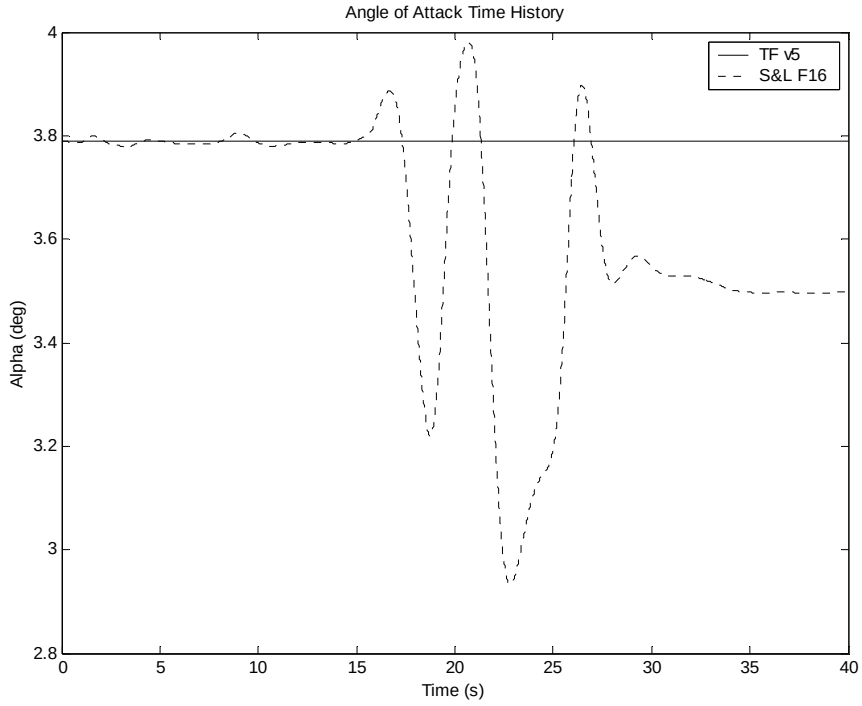

Figure 4.20 – Time History of Angle of Attack for the *S*-turn's Lateral-Directional Controls

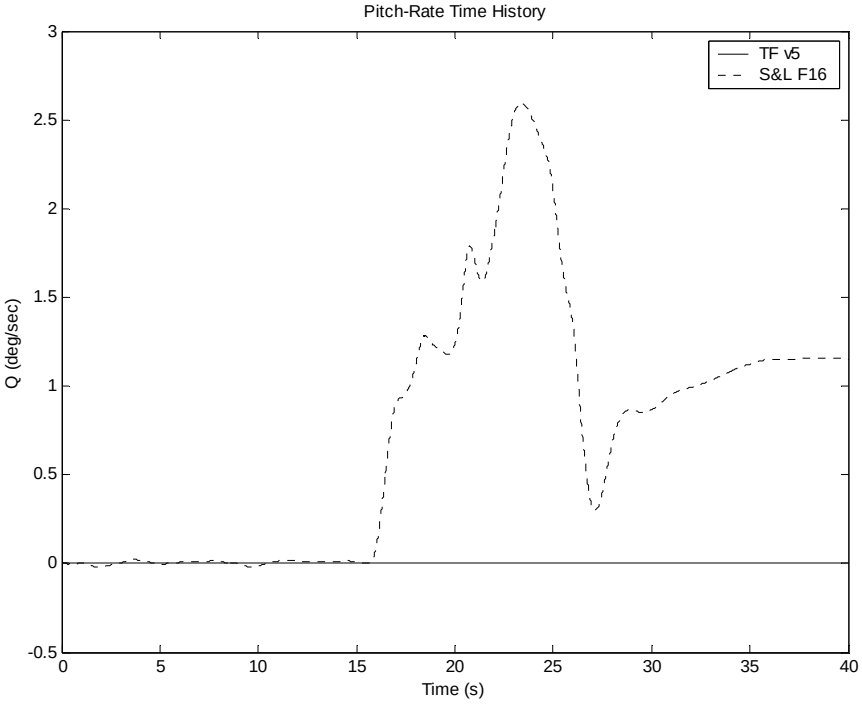

Figure 4.21 – Time History of Pitch-Rate for the *S*-turn's Lateral-Directional Controls

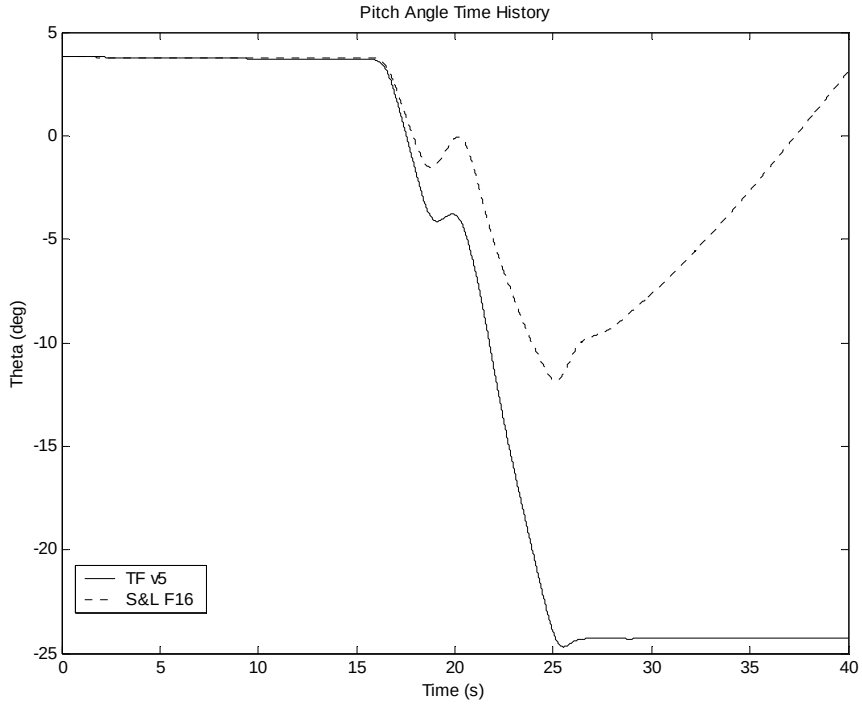

Figure 4.22 – Time History of Pitch Angle for the *S*-turn's Lateral-Directional Controls

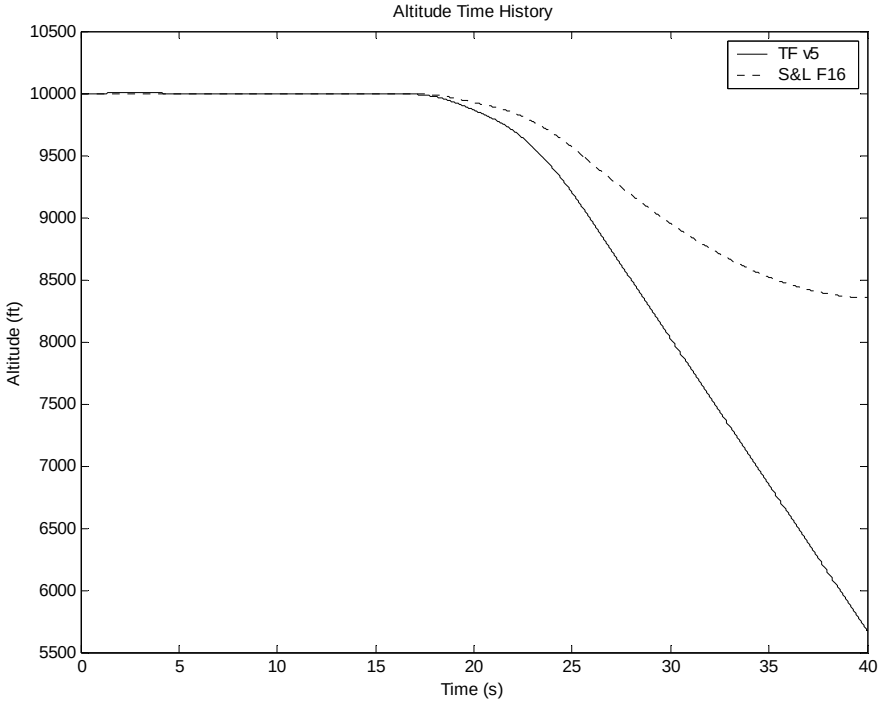

Figure 4.23 – Time History of Altitude for the *S*-turn's Lateral-Directional Controls

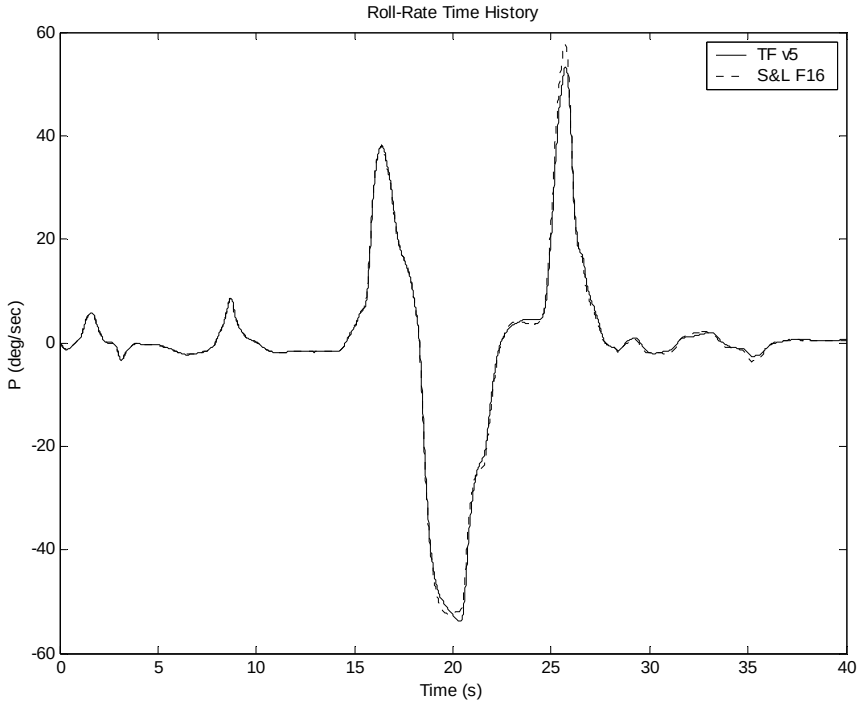

Figure 4.24 – Time History of Roll-Rate for the *S*-turn's Lateral-Directional Controls

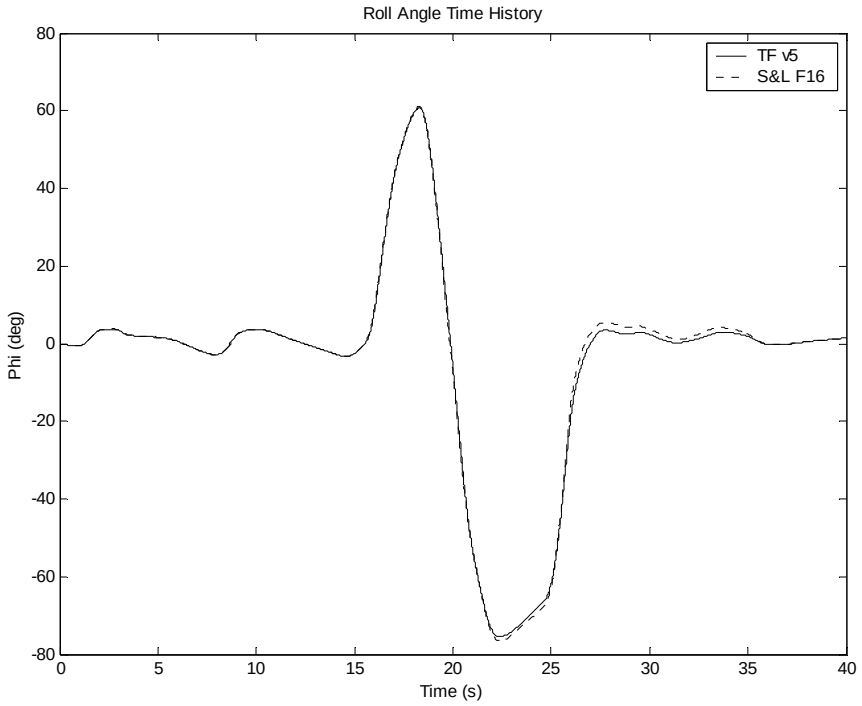

Figure 4.25 – Time History of Roll Angle for the *S*-turn's Lateral-Directional Controls

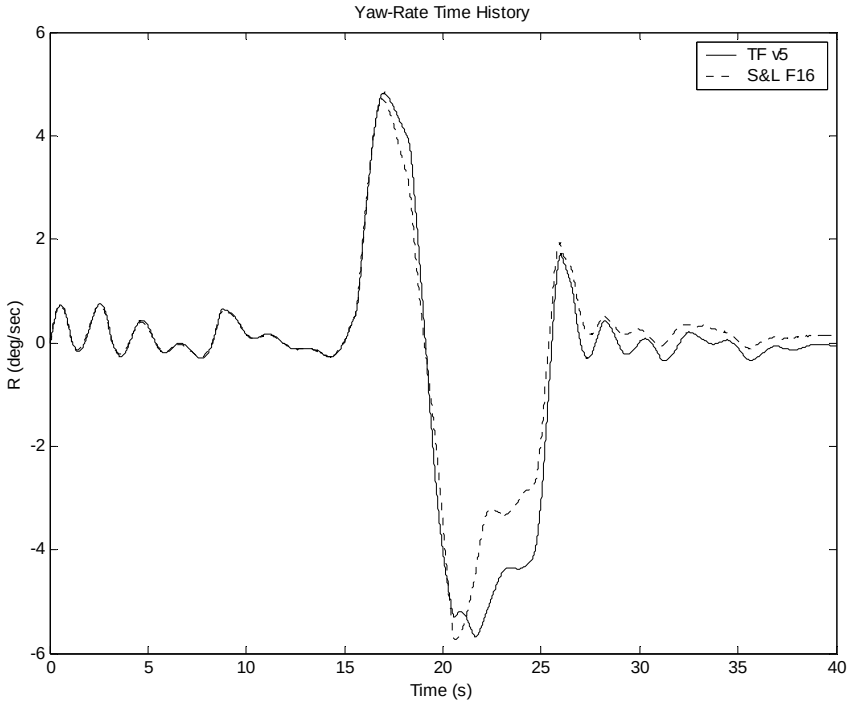

Figure 4.26 – Time History of Yaw-Rate for the *S*-turn's Lateral-Directional Controls

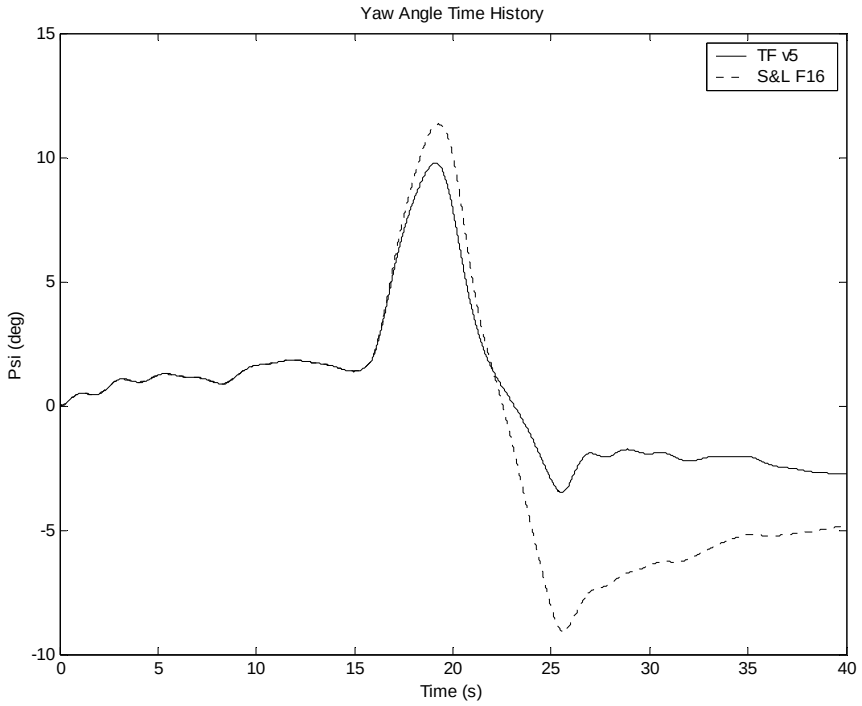

Figure 4.27 – Time History of Yaw Angle for the *S*-turn's Lateral-Directional Controls

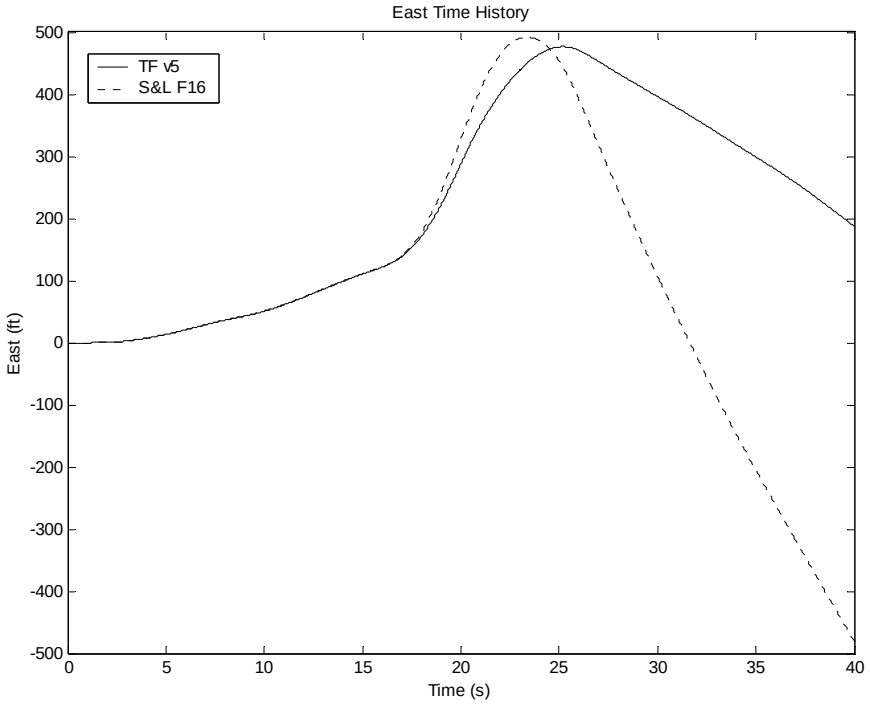

Figure 4.28 – Time History of East Position for the *S*-turn's Lateral-Directional Controls

4.3 Formation Flight

4.3.1 Control Inputs

The TFSM is a linear simulation that is valid for very small perturbations. Yet, the previous flight test exhibited large changes in the state variables that possibly violated the Taylor Series expansion used during the linearization. Therefore, running the simulations with a more conservative control input history might enable the simulation to behave more like the Stevens and Lewis F-16 model. A formation-flying task was the flight test of choice. Not only would the control inputs be much smaller than the *S*-turn's, but this was also the type of flight used to do the PIO research. Figures 4.30 and 4.31 contain the formation flight's control deflections; the control's units are the same as those presented in Section 4.2.1. Note that in this flight test, the rudder deflection is zero degrees.

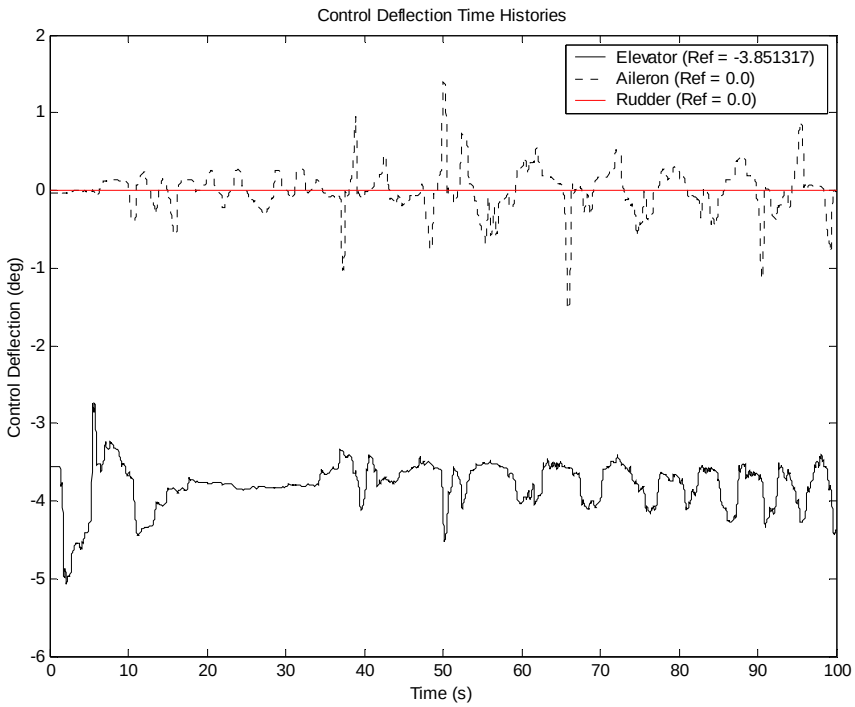

Figure 4.29 – Time History of Control Surface Deflections for the Formation Flight Test

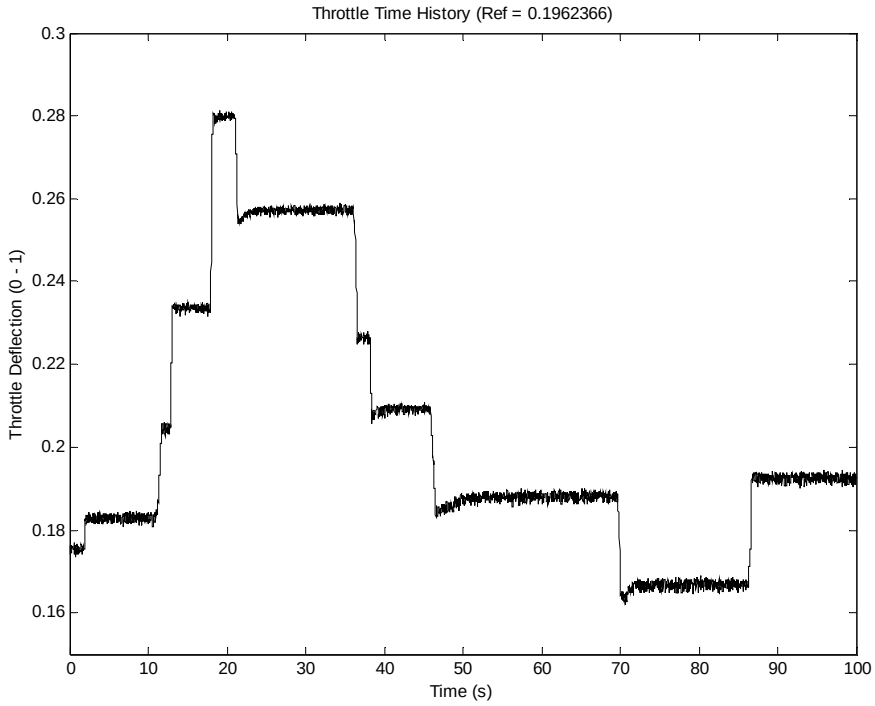

Figure 4.30 – Time History of the Throttle Deflection for the Formation Flight Test

4.3.2 Full Controls

The formation flight's control time histories generated this section's aircraft state plots for both simulations. The velocity response in Figure 4.31 shows that the TFSM behaved more like the F-16 for the conservative control inputs than it did for the *S*-turn test. There was an error of approximately four ft/sec between 37 and 80 seconds, but the two responses began to converge at the end of the simulation. Figures 4.32 through 4.35 contain the other four aircraft state variables calculated by the TFSM's state-space representations: angle of attack, roll-, pitch-, and yaw-rate. The responses for the states had almost no error between the two simulations except at the peak values during the entire flight test. The yaw-rate had the most pronounced difference, but even that was only a five percent error at the peak values. Figures 4.36 – 4.38 plot the aircraft angles and show that the well-behaved rates lead to better angular responses. Because the roll angle response closely followed the roll-rate, the TFMS's roll angle response almost matched the F-16's response. However, the longitudinal response problem, noticed during the S-turn flight test, persisted into this test. The TFSM's pitch angle response began to fall away from the F-16's after 40 seconds of simulation when the roll angles started to get larger. Since the

pitch angle was drifting, the TFSM's yaw angle response also drifted to obey the kinematic equations. Altitude and east positions followed the pitch and yaw angles closely, so it came as no surprise that the TFSM's altitude fell off as it drifted to the east (Figures 4.39 and 4.40)

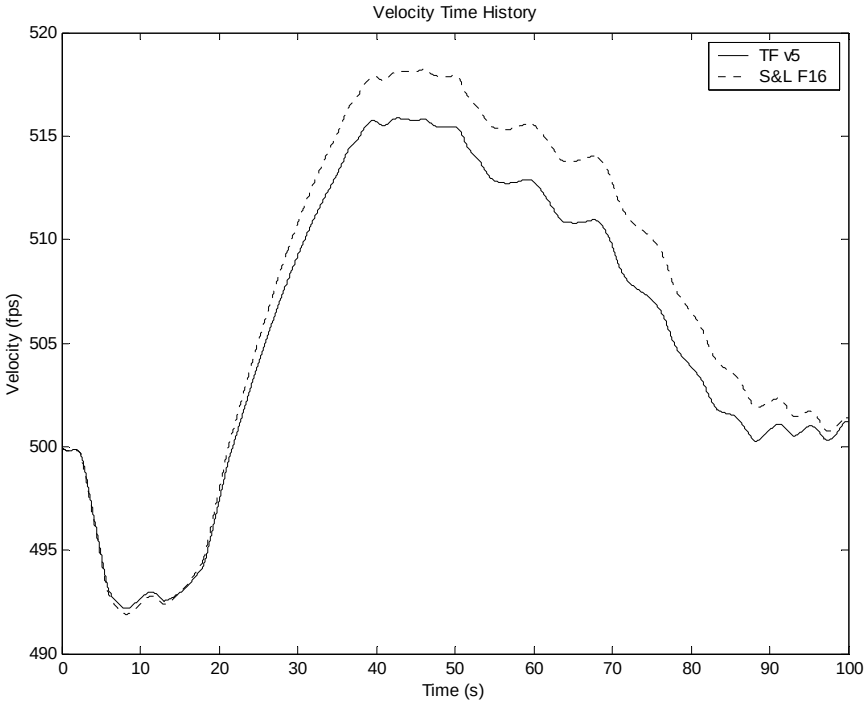

Figure 4.31 – Time History of Total Velocity for the Formation Flight Test

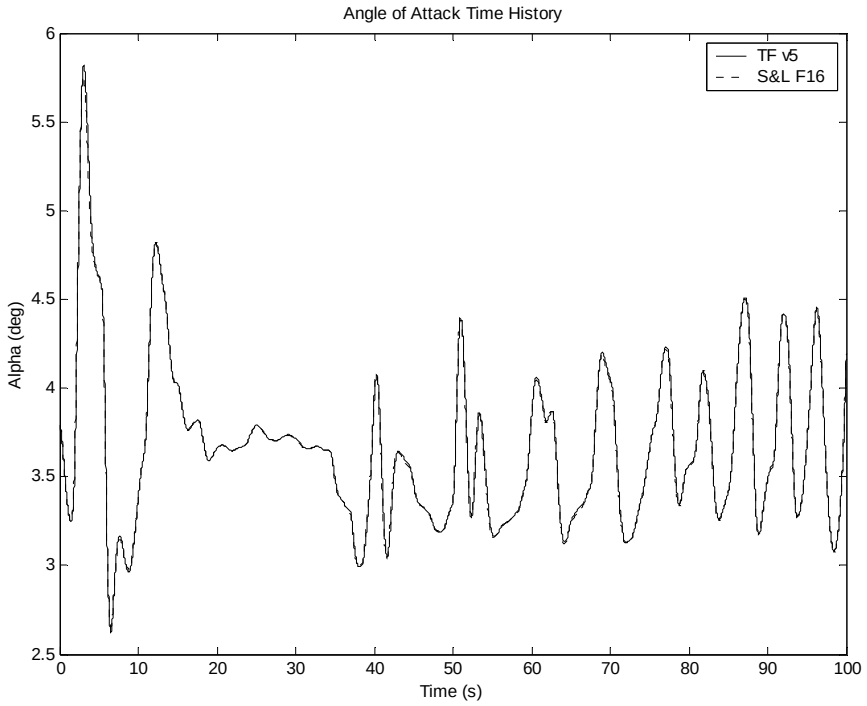

Figure 4.32 – Time History of Angle of Attack for the Formation Flight Test

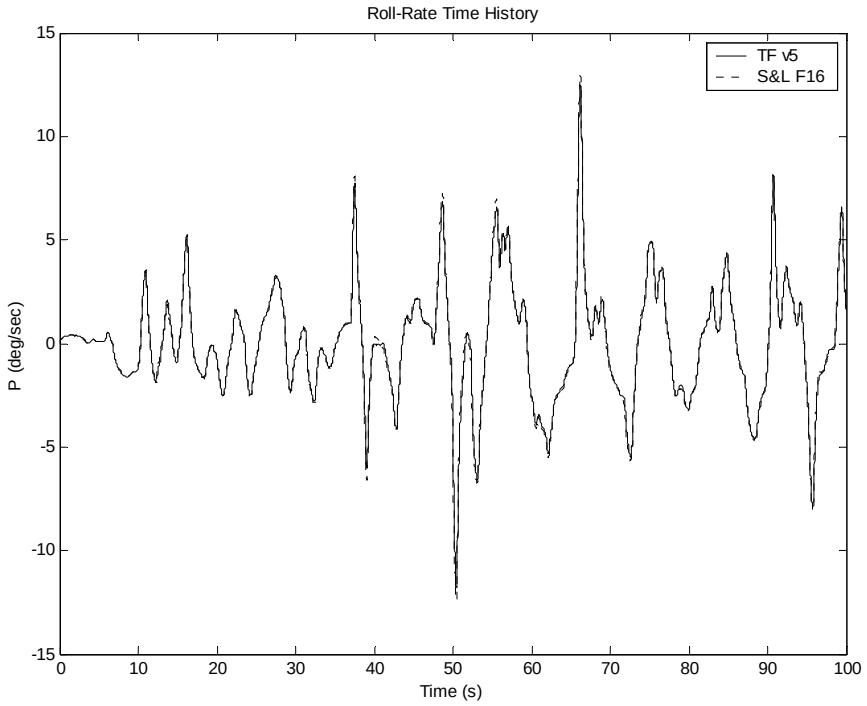

Figure 4.33 – Time History of Roll-Rate for the Formation Flight Test

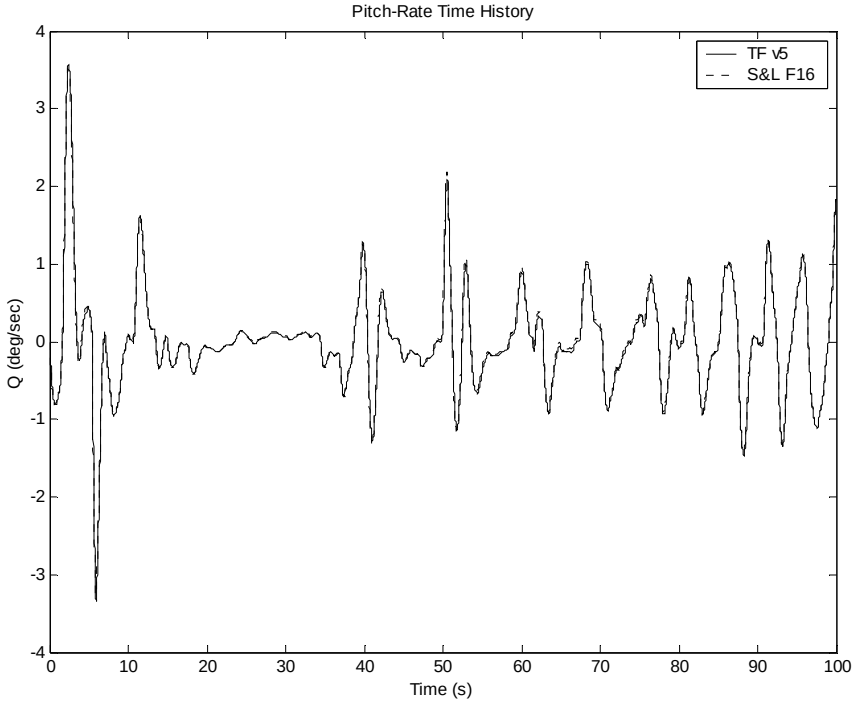

Figure 4.34 – Time History of Pitch-Rate for the Formation Flight Test

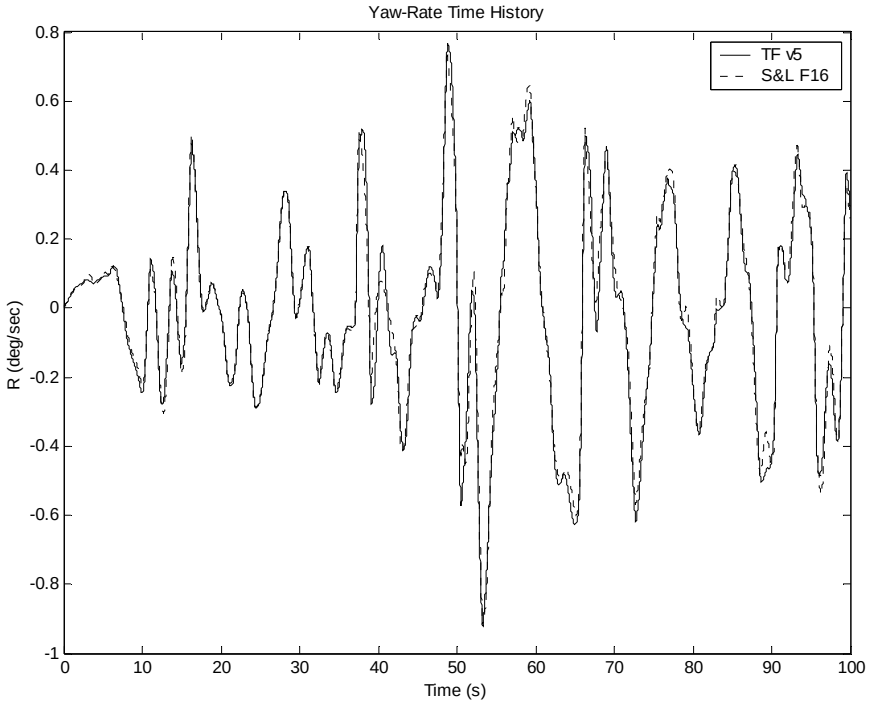

Figure 4.35 – Time History of Yaw-Rate for the Formation Flight Test

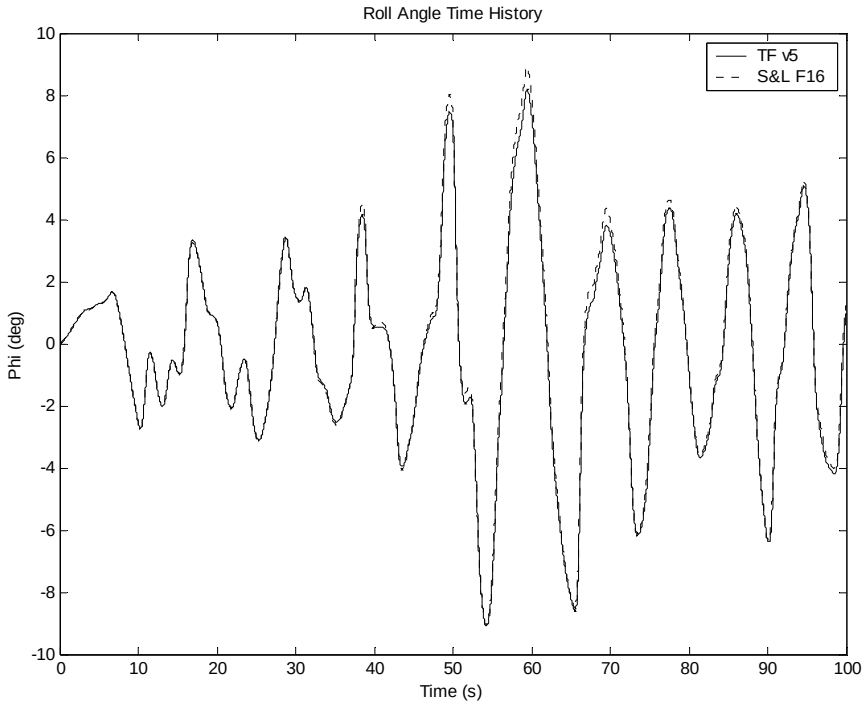

Figure 4.36 – Time History of Roll Angle for the Formation Flight Test

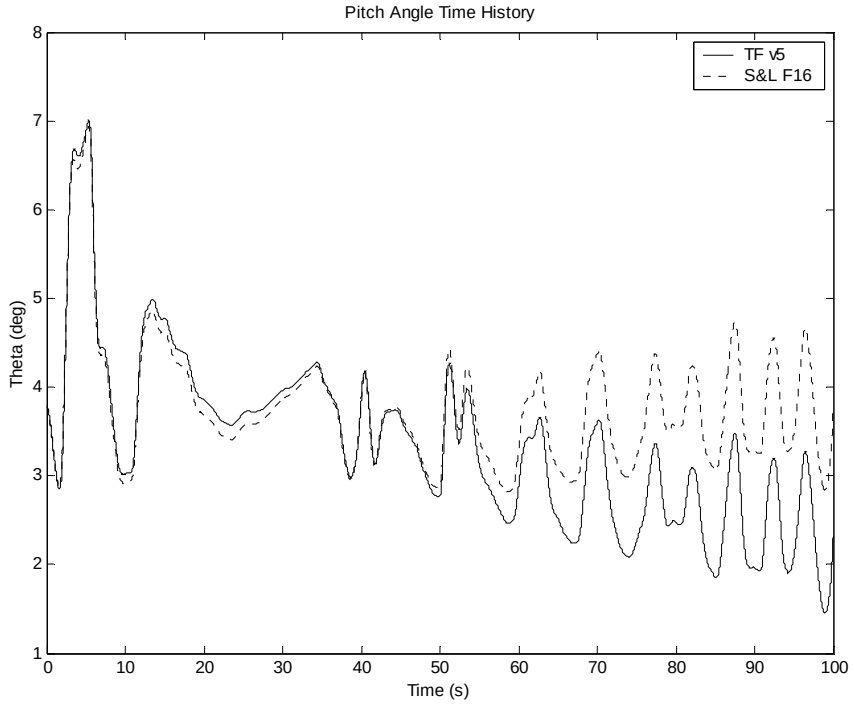

Figure 4.37 – Time History of Pitch Angle for the Formation Flight Test

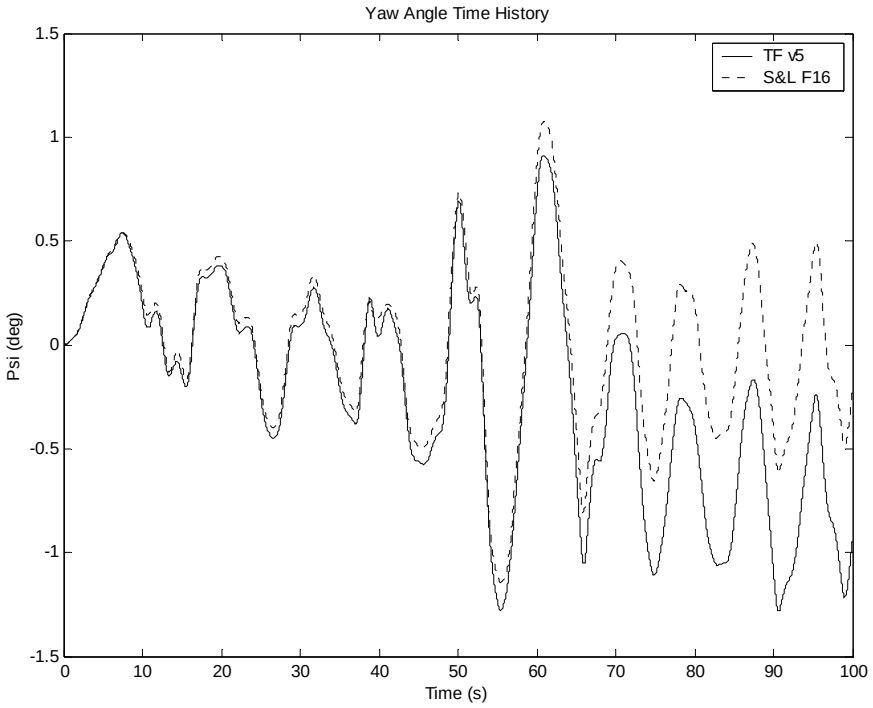

Figure 4.38 – Time History of Yaw Angle for the Formation Flight Test

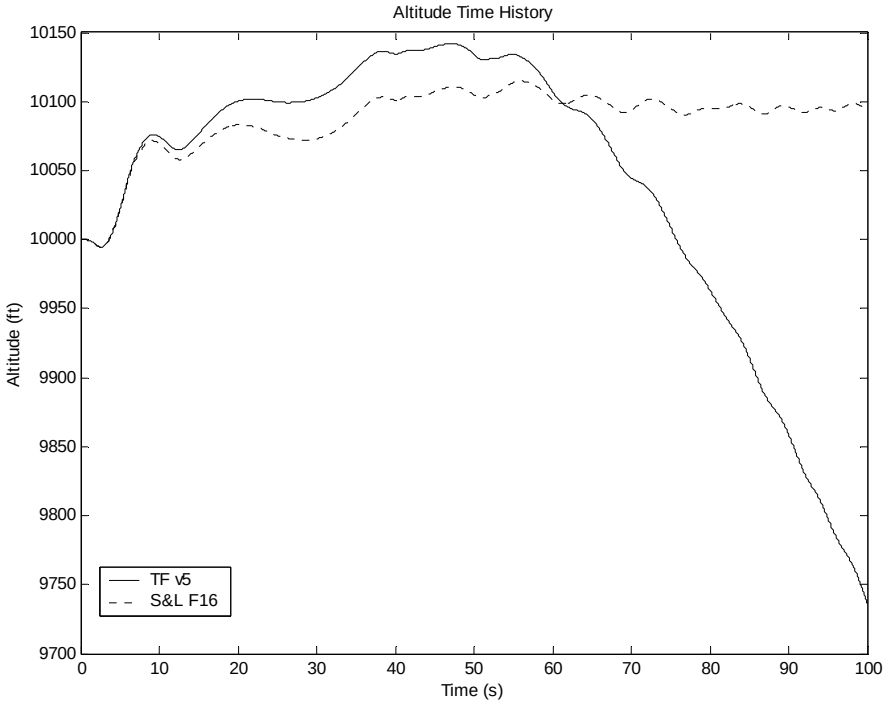

Figure 4.39 – Time History of Altitude for the Formation Flight Test

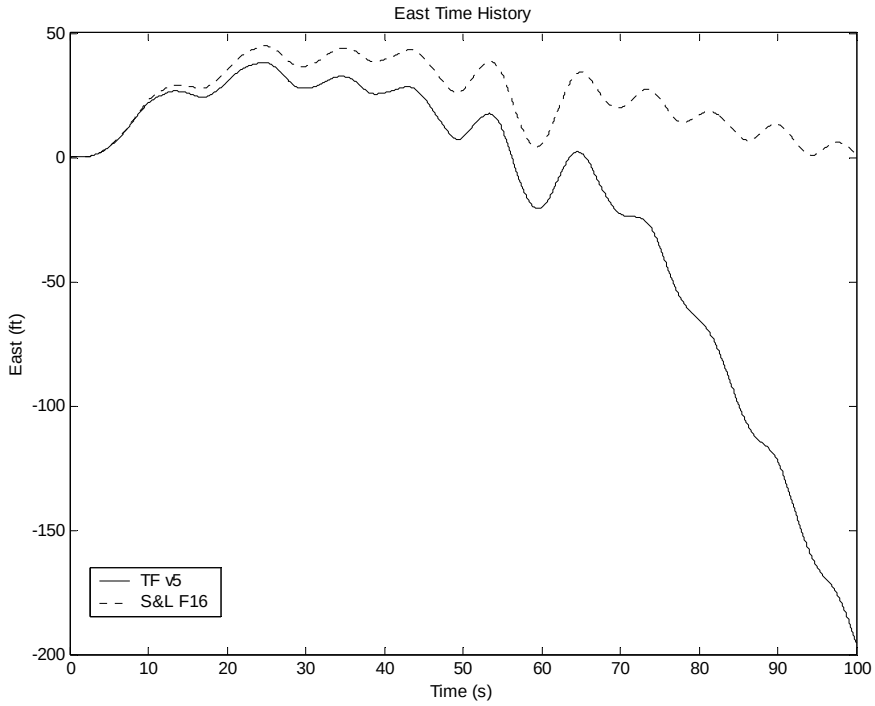

Figure 4.40 – Time History of East Position for the Formation Flight Test

4.3.3 Longitudinal Controls Only

As in Section 4.2.3, this section presents only the longitudinal states because the lateral-directional states' responses to the longitudinal controls were negligible. Velocity responses (Figure 4.41) from the longitudinal controls behaved almost identically to their responses for all four controls. This occurred because the roll angles were small during the fully controlled flight, and therefore, their affect on the aircraft's total velocity was minimal. However, the TFSM's velocity matched better with the lateral-directional controls removed (no bank angle). The angle of attack and pitch-rate time histories (Figures 4.42 and 4.43) also aligned themselves better with even less error than they had originally. By improving the other states and removing the aircraft's roll angle, the TFSM's pitch angle (Figure 4.44) followed the F-16's very closely. The largest error occurred between 10 and 35 seconds, coinciding with the greatest throttle inputs during the flight test. The TFSM's altitude remained below the F-16's due to the slower aircraft speed and smaller pitch angle between 10 and 50 seconds, but followed the same shape as the F-16 with an offset of about 12 ft.

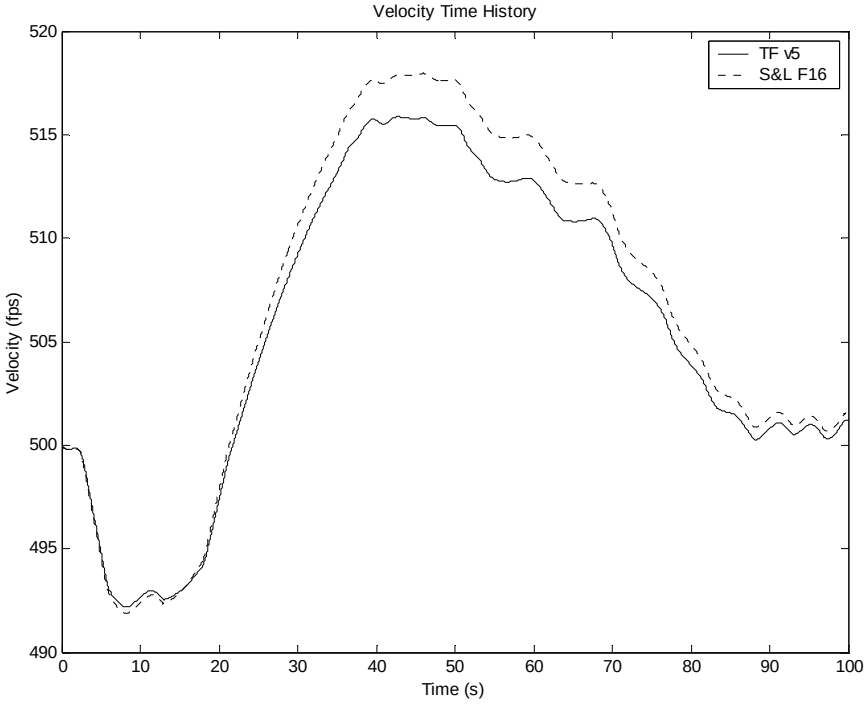

Figure 4.41 – Time History of Total Velocity for the Formation Flight Test's Longitudinal Controls

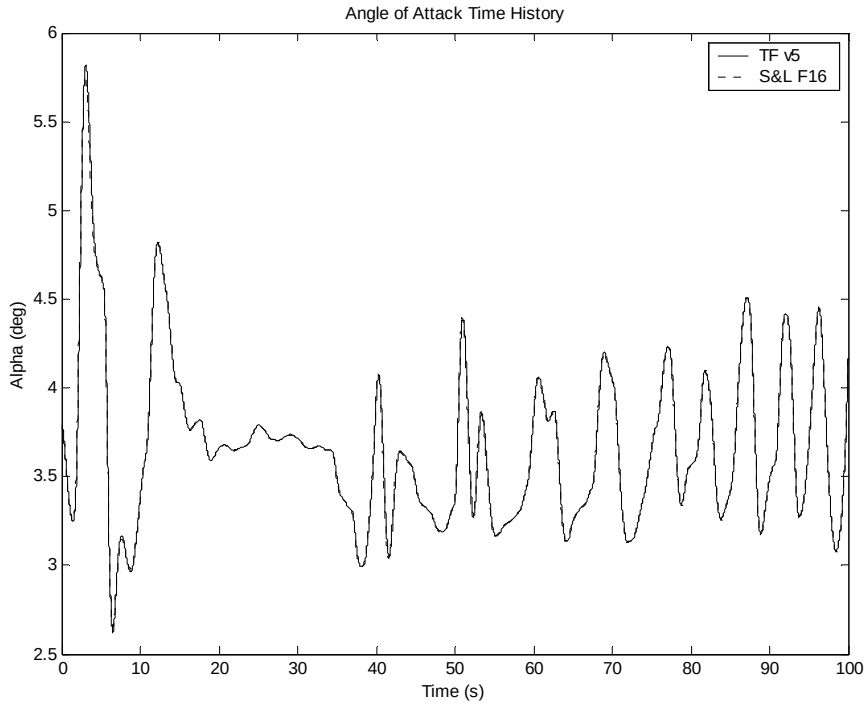

Figure 4.42 – Time History of Angle of Attack for the Formation Flight Test's Longitudinal Controls

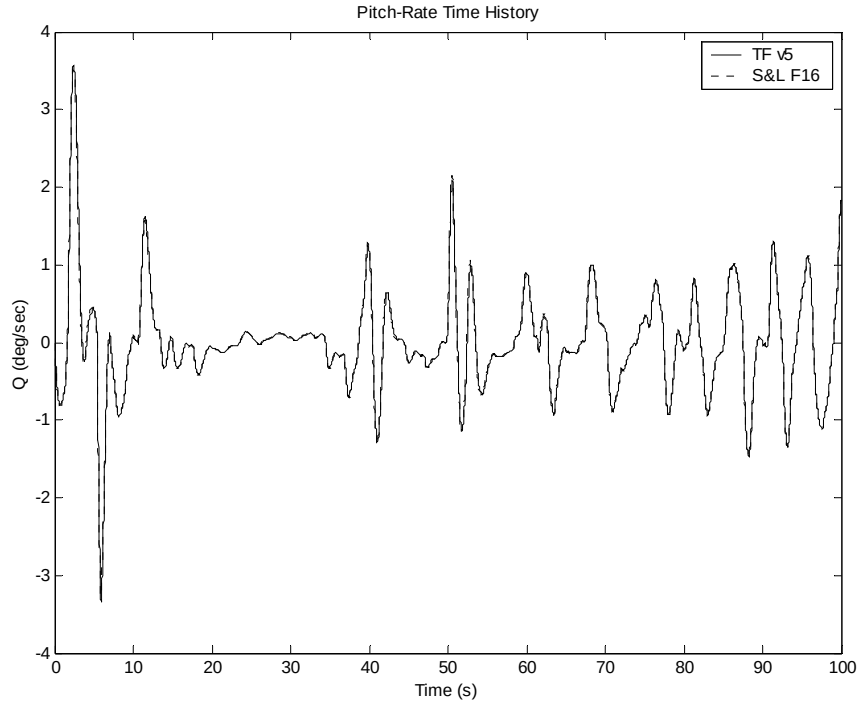

Figure 4.43 – Time History of Pitch-Rate for the Formation Flight Test's Longitudinal Controls

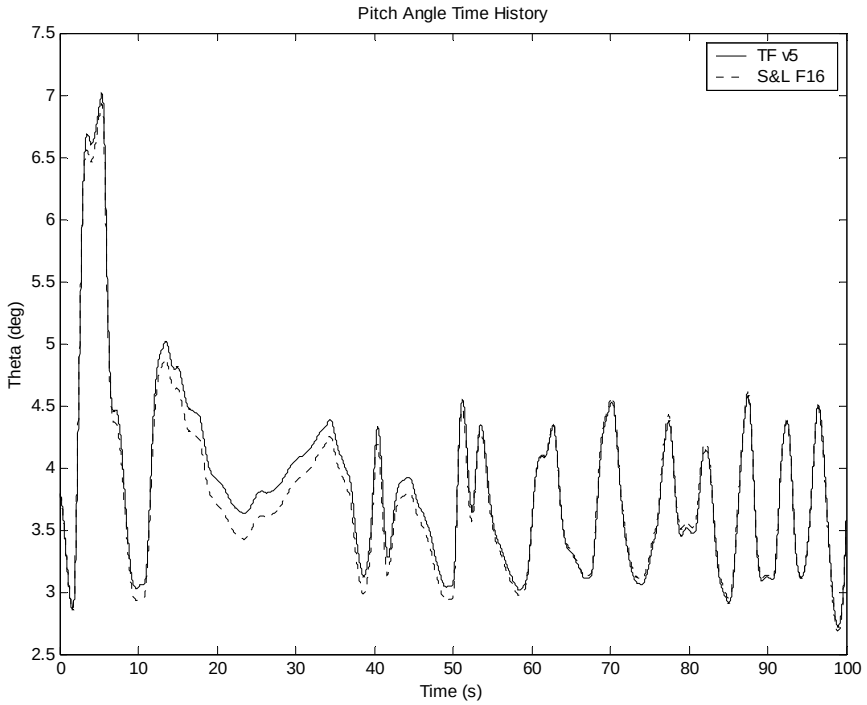

Figure 4.44 – Time History of Pitch Angle for the Formation Flight Test's Longitudinal Controls

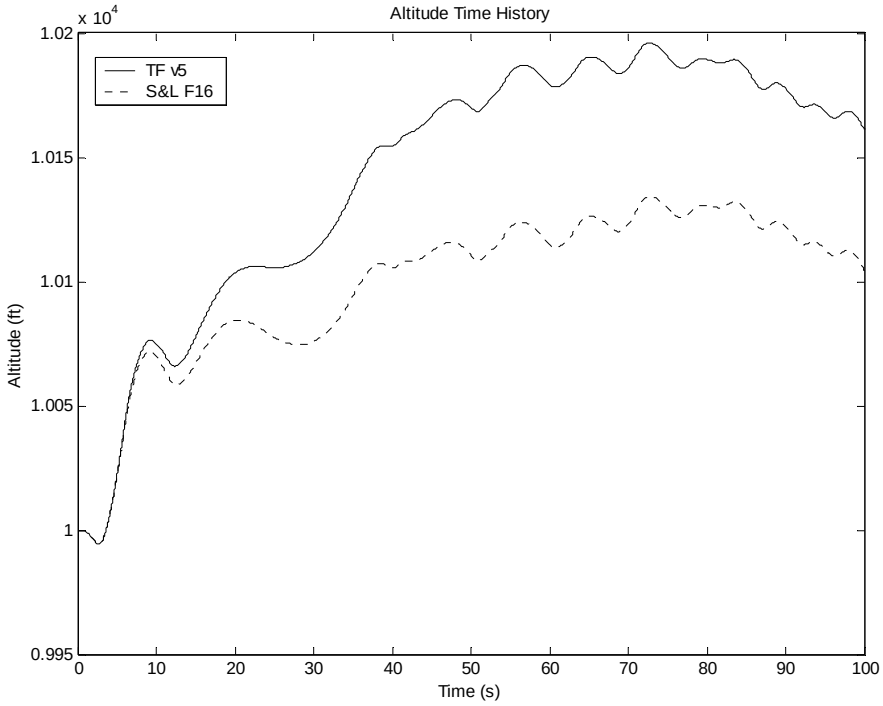

Figure 4.45 – Time History of Altitude for the Formation Flight Test's Longitudinal Controls

4.3.4 Lateral-Directional Controls Only

Only the lateral-directional controls were present during this flight test. Since the rudder deflection was zero, the ailerons controlled this set of simulation runs. All of the lateral-directional states had almost identical responses for aileron inputs alone (Figures 4.46 – 4.50). Similar to the results in Section 4.2.4, the longitudinal responses should not be constant for a simulation with only lateral-directional controls. The F-16's velocity (Figure 4.51) increased slightly as the nose lowered (Figure 4.52) and the aircraft continued to roll back and forth. Unfortunately, the longitudinal states controlled by transfer function realizations in the TFSM remained constant throughout the simulation. Although the F-16's angle of attack did not decrease by more than 0.01 degrees, the response's general trend was important because it was not constant; it varied as the aircraft banked (Figure 4.52). Likewise, in Figure 4.53, the F-16's pitch-rate response was very small, but its shape was very different from the TFSM's constant zero value. The most drastic difference arose in the integrated longitudinal states: pitch angle and altitude. They must obey the laws of flight dynamics; they used the constant longitudinal

states and the changing lateral-directional states to integrate themselves during each time step. Unfortunately, the other longitudinal variables were constant, which means that the pitch angle and altitude integrated all the error from the other states. Figures 4.53 and 4.54 present the time histories for the pitch angle and the aircraft's altitude trajectory; both contain large amounts of error between the two simulations.

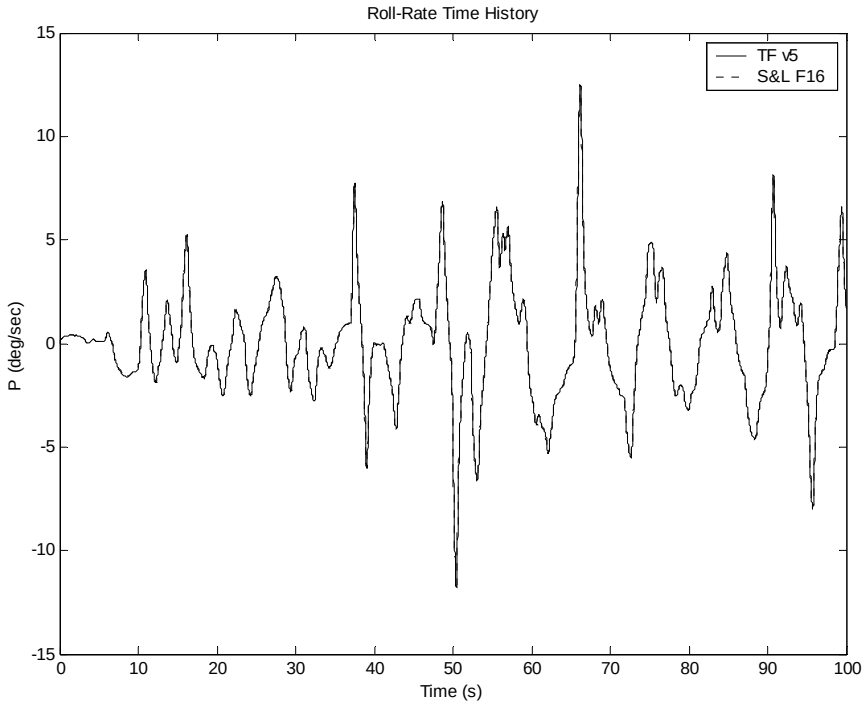

Figure 4.46 – Time History of Roll-Rate for the Formation Flight Test's Lateral-Directional Controls

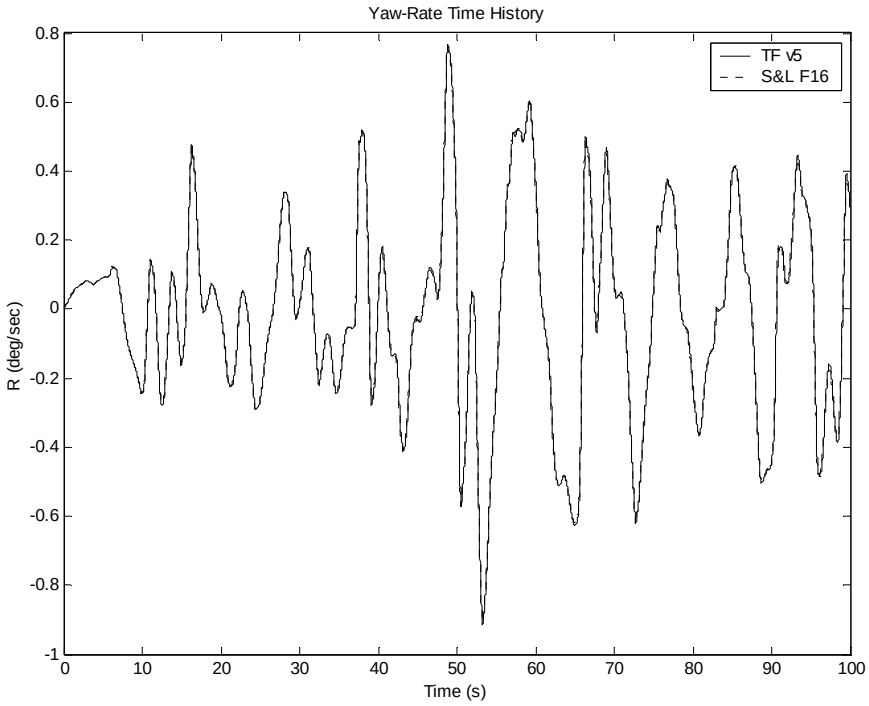

Figure 4.47 – Time History of Yaw-Rate for the Formation Flight Test's Lateral-Directional Controls

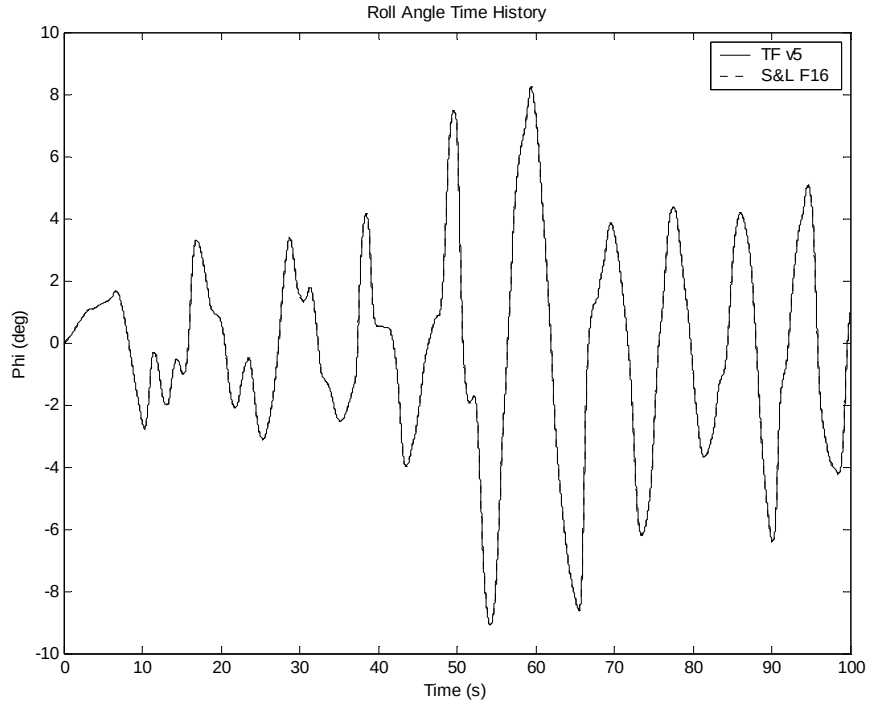

Figure 4.48 – Time History of Roll Angle for the Formation Flight Test's Lateral-Directional Controls

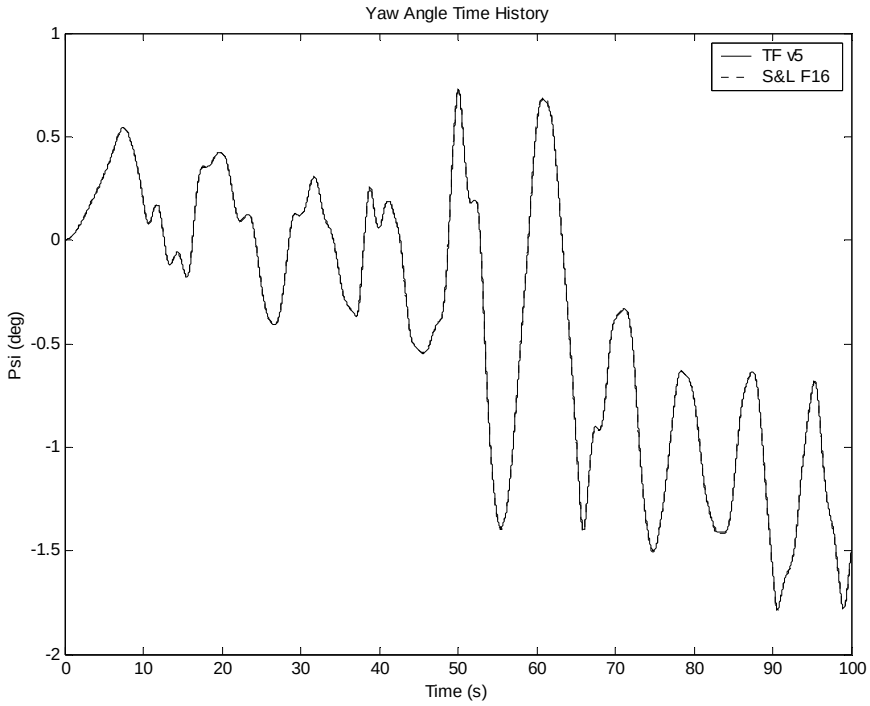

Figure 4.49 – Time History of Yaw Angle for the Formation Flight Test's Lateral-Directional Controls

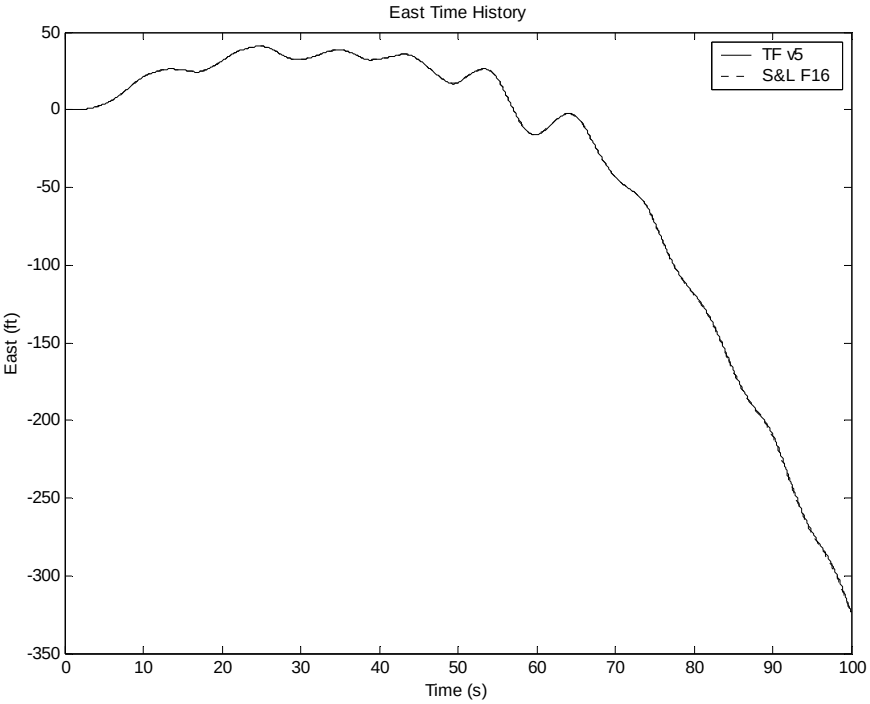

Figure 4.50 – Time History of East Position for the Formation Flight Test's Lateral-Directional Controls

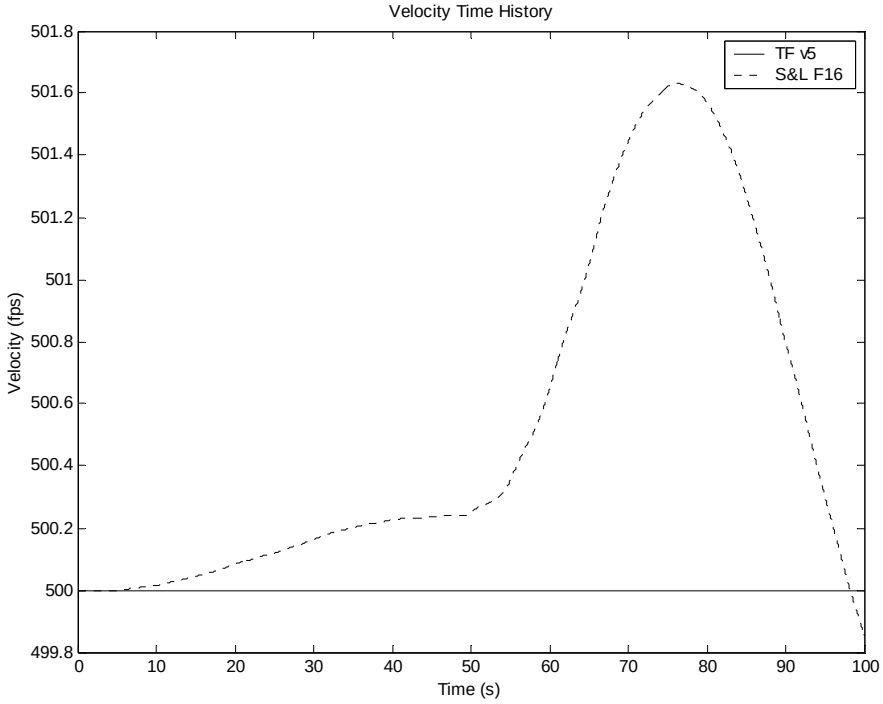

Figure 4.51 – Time History of Total Velocity for the Formation Flight Test's Lateral-Directional Controls

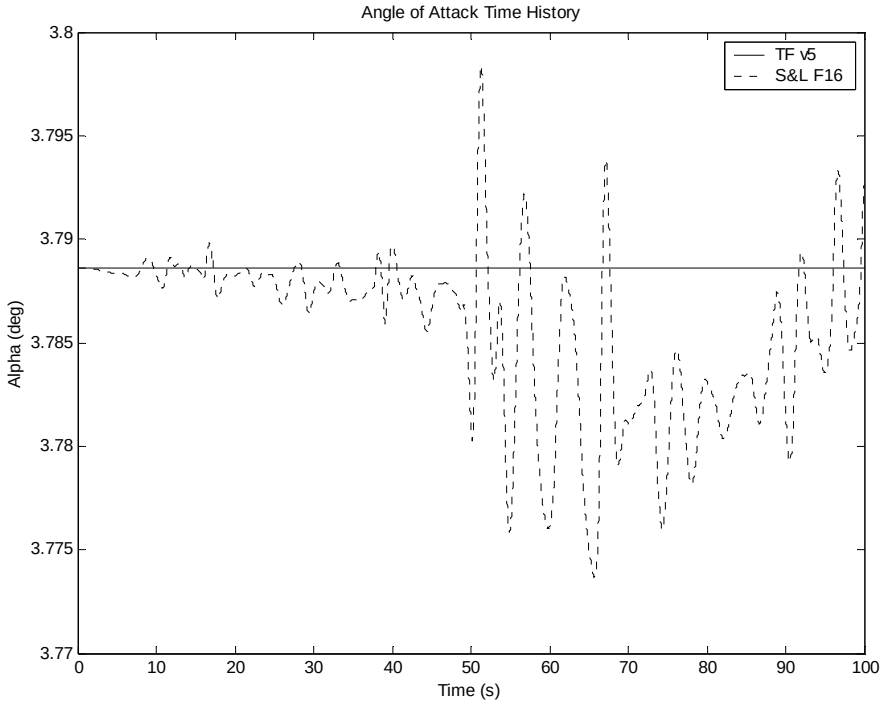

Figure 4.52 – Time History of Angle of Attack for the Formation Flight Test's Lateral-Directional Controls

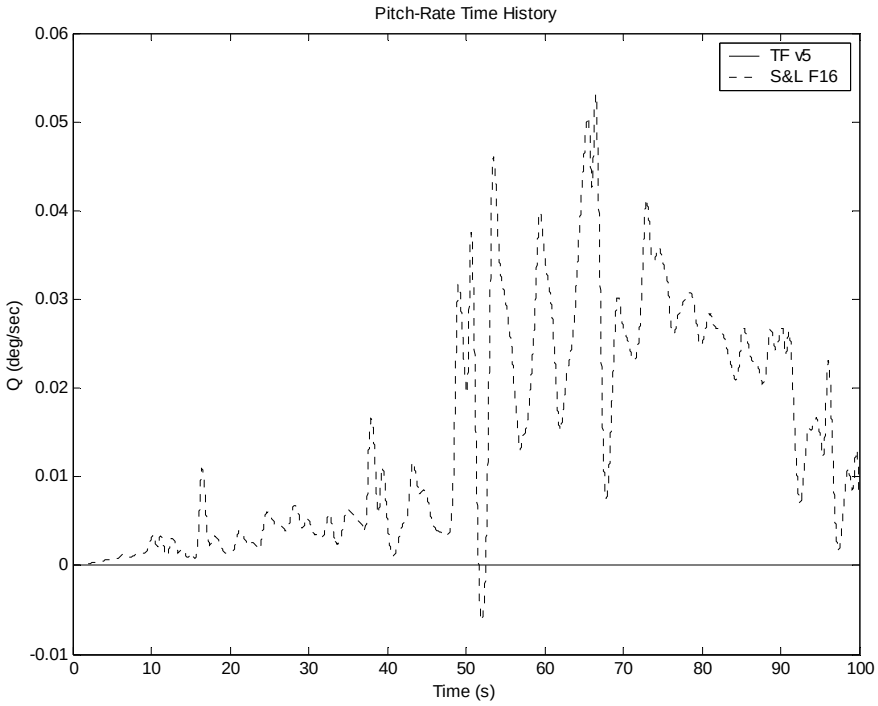

Figure 4.53 – Time History of Pitch-Rate for the Formation Flight Test's Lateral-Directional Controls

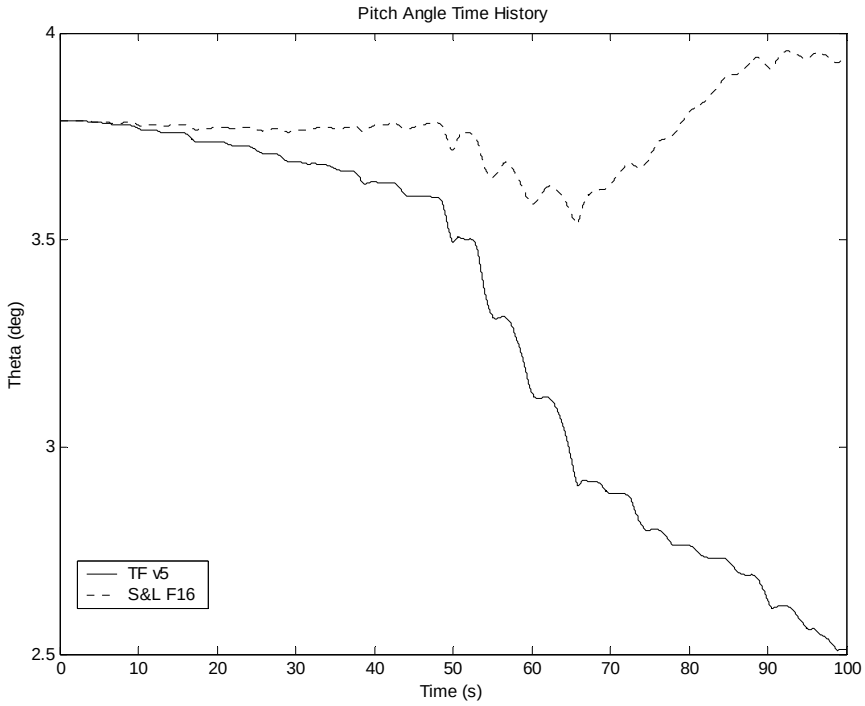

Figure 4.54 – Time History of Pitch Angle for the Formation Flight Test's Lateral-Directional Controls

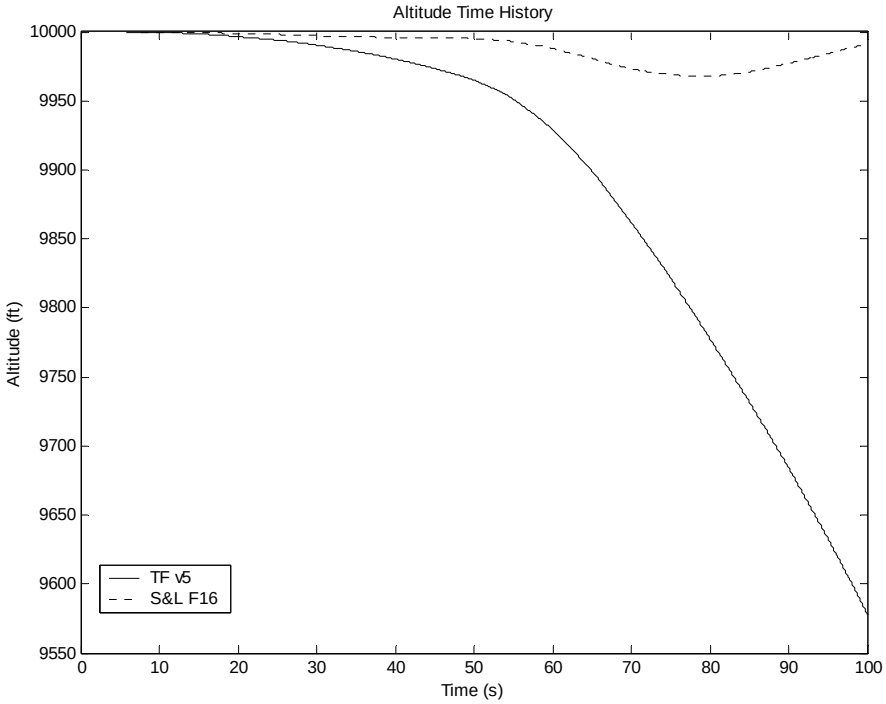

Figure 4.55 – Time History of Altitude for the Formation Flight Test's Lateral-Directional Controls

4.4 Uncontrolled Flight (Banked Initial Condition)

The previous flight tests indicated that the TFSM ignored the effects of gravity when its wings were not level. To test this, the simulation's initial conditions were changed from wing's level flight to an initial bank angle of 60 degrees. Although this is not a small perturbation, it was chosen to amplify the TFSM's response, since the model responds in a similar manner no matter how large or small the disturbance is. The controls remained at their trim deflections to isolate gravity's effects on the aircraft's motion. With no control deflections, this initial condition should cause the aircraft's nose to drop, making it gain speed and lose altitude. As expected, the F-16 performed in the desired manner, and the TFSM did not. All the aircraft states that the TFSM calculated transfer functions for (total velocity, angle of attack, sideslip angle, roll-rate, pitch-rate, and yaw-rate presented in Figures 4.56 – 5.61) remained constant without control deflections. The responses were correct for the state-space representations that generate them,

but their behavior was not the desired one in the area of flight dynamics. Therefore, the state responses dependent on the previous states were not the desired responses. Figures 4.59, 4.60, and 4.61 contain the aircraft's roll, pitch, and yaw angles, which were constant because all the body-axis rates were zero. Aircraft altitude (Figure 4.62) and east position (Figure 4.63) progressed at a constant rate due to the constant bank angle included in the navigation equations. In summary, the F-16 responses were due to the states' initial conditions, and the TFSM responds only to control inputs, which were not present in this flight test. The F-16's longitudinal responses exhibited the Short Period and Phugoid responses, and the lateral-directional states revealed the Dutch Roll.

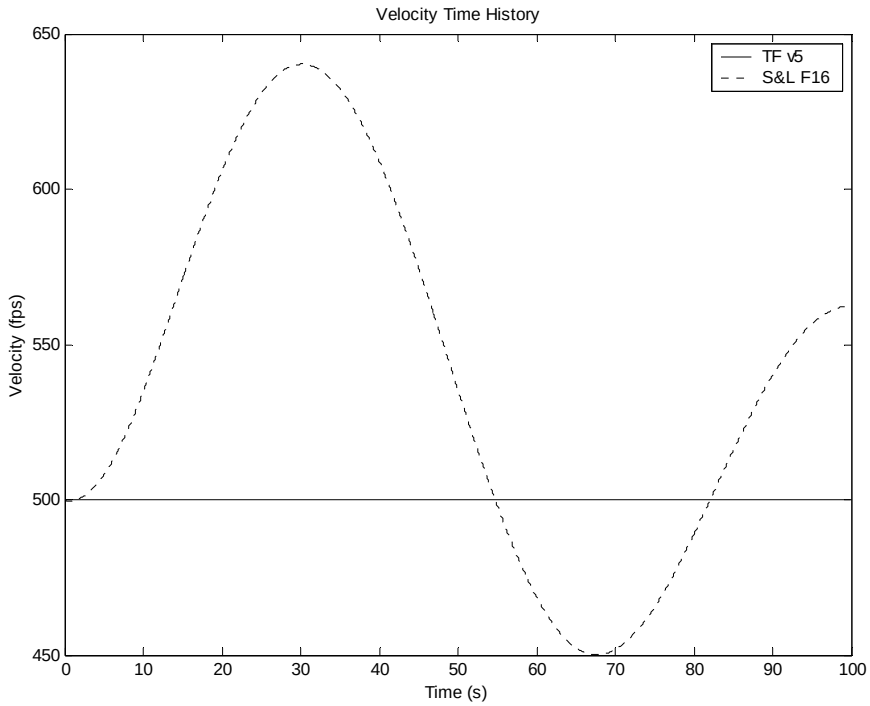

Figure 4.56 – Time History of Velocity for the original Uncontrolled Flight Test

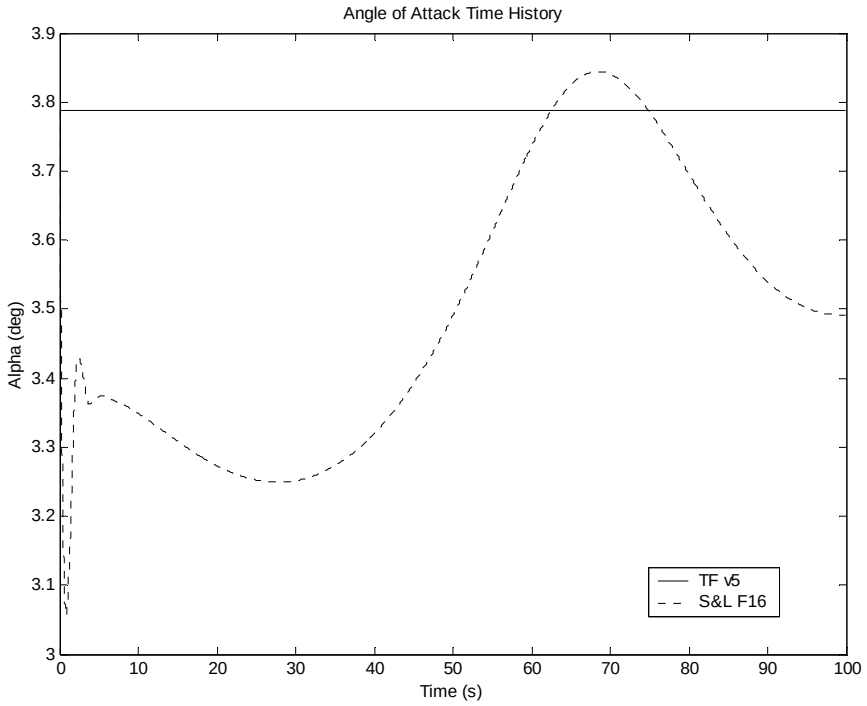

Figure 4.57 – Time History of Angle of Attack for the original Uncontrolled Flight Test

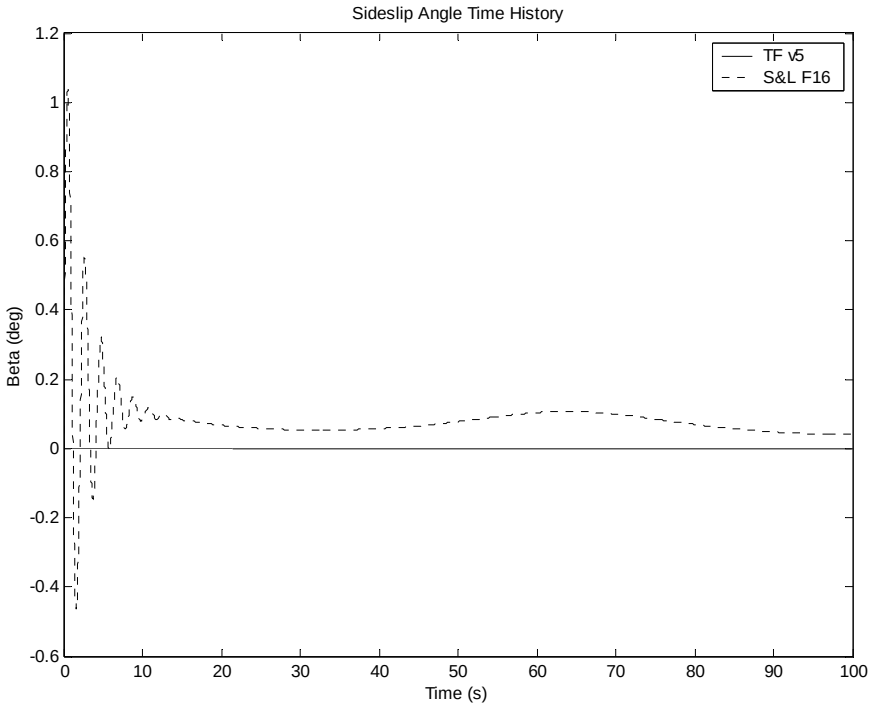

Figure 4.58 – Time History of Sideslip for the original Uncontrolled Flight Test

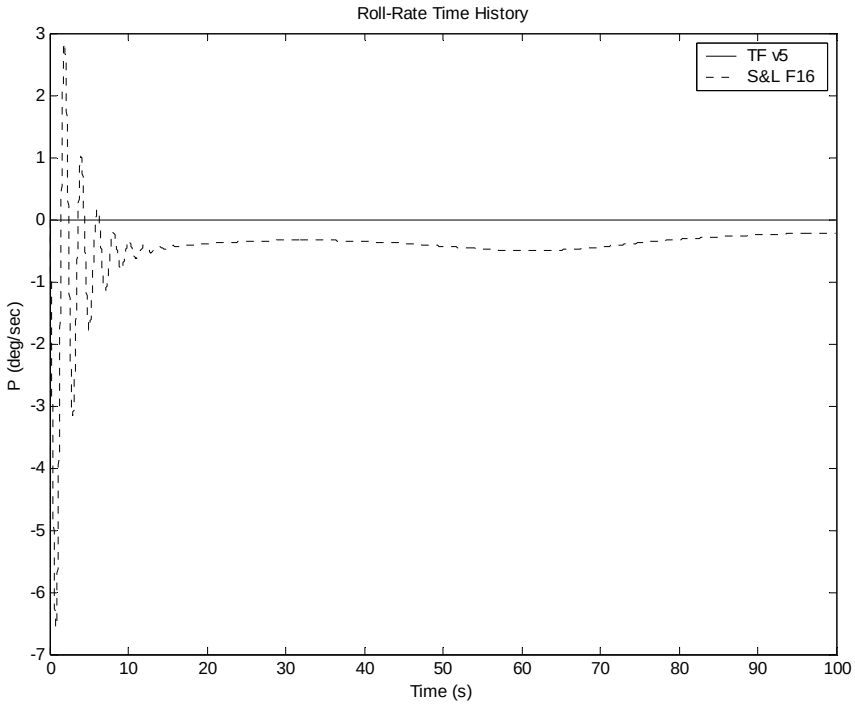

Figure 4.59 – Time History of Roll-Rate for the original Uncontrolled Flight Test

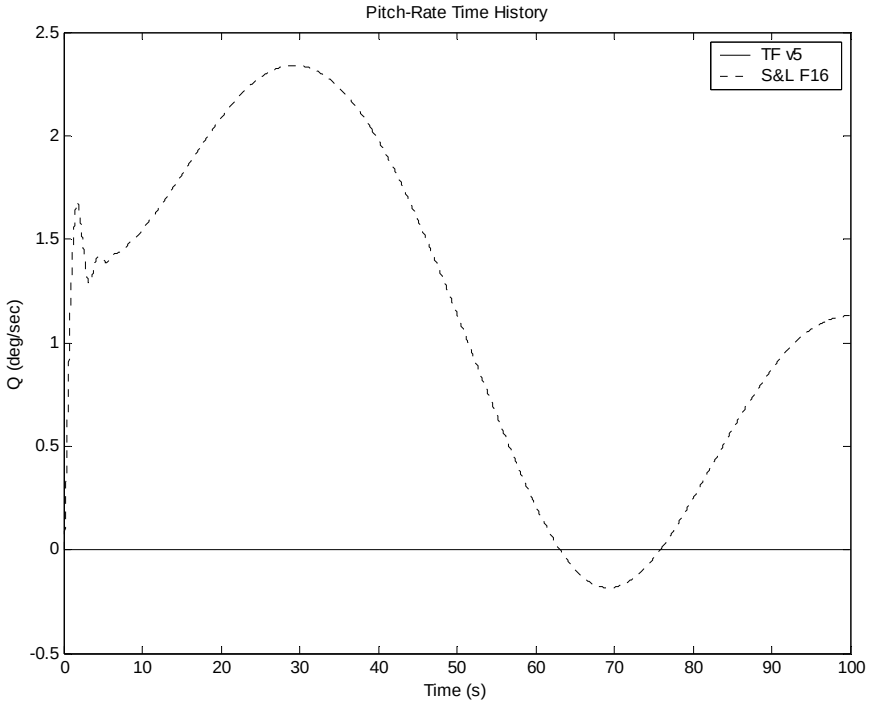

Figure 4.60 – Time History of Pitch-Rate for the original Uncontrolled Flight Test

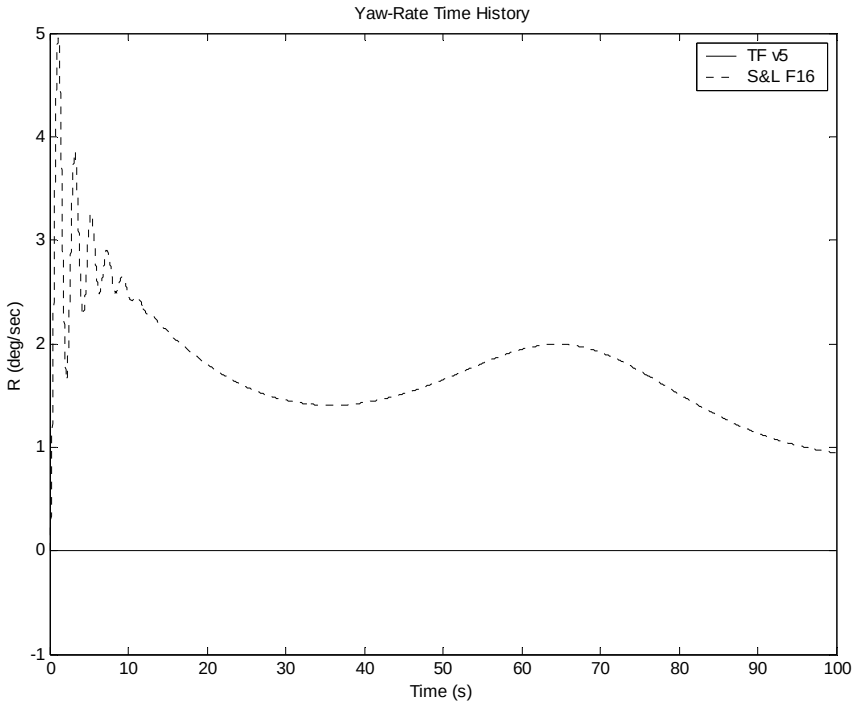

Figure 4.61 – Time History of Yaw-Rate for the original Uncontrolled Flight Test

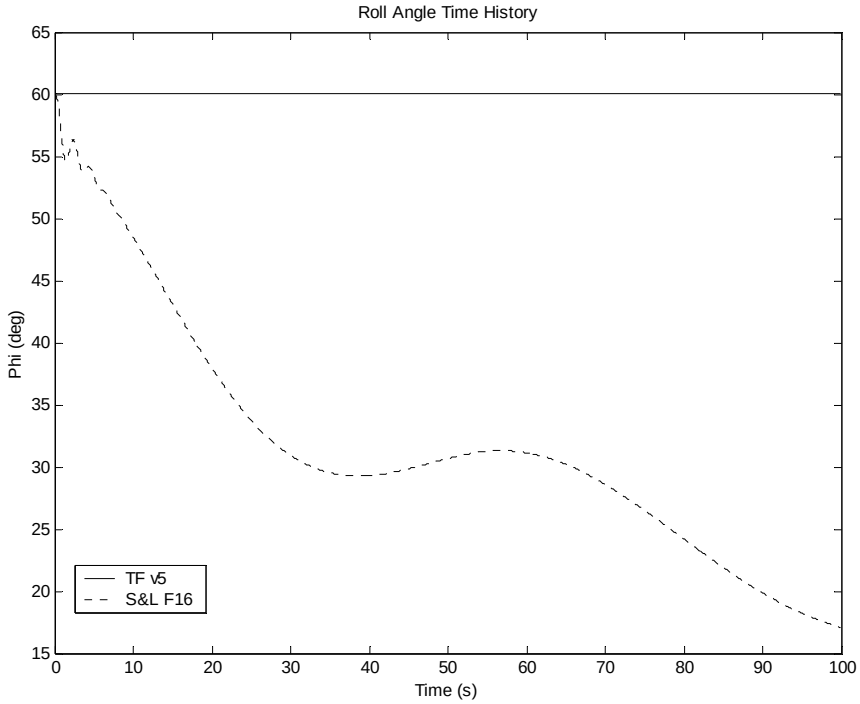

Figure 4.62 – Time History of Roll Angle for the original Uncontrolled Flight Test

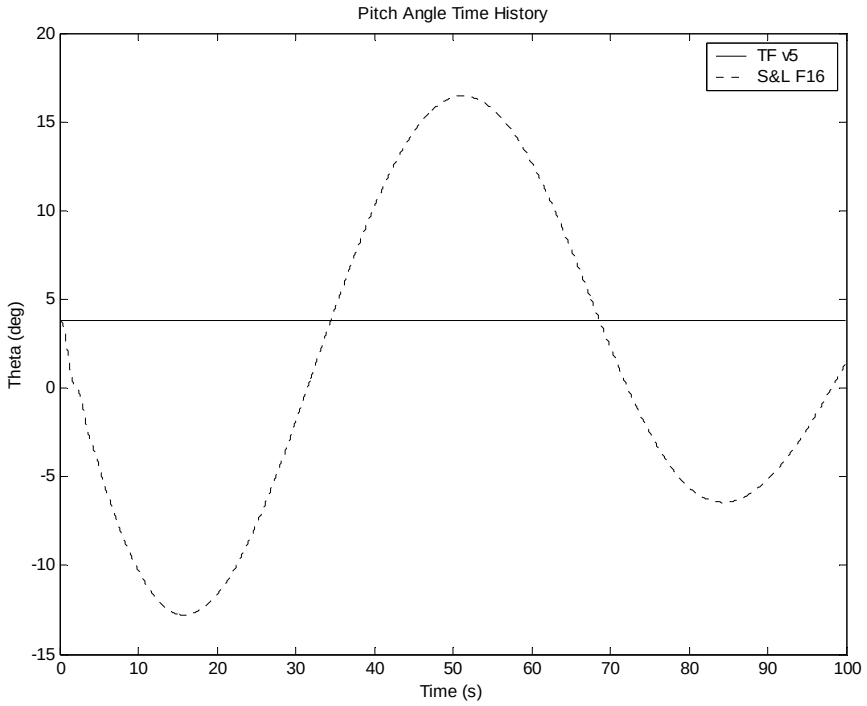

Figure 4.63 – Time History of Pitch Angle for the original Uncontrolled Flight Test

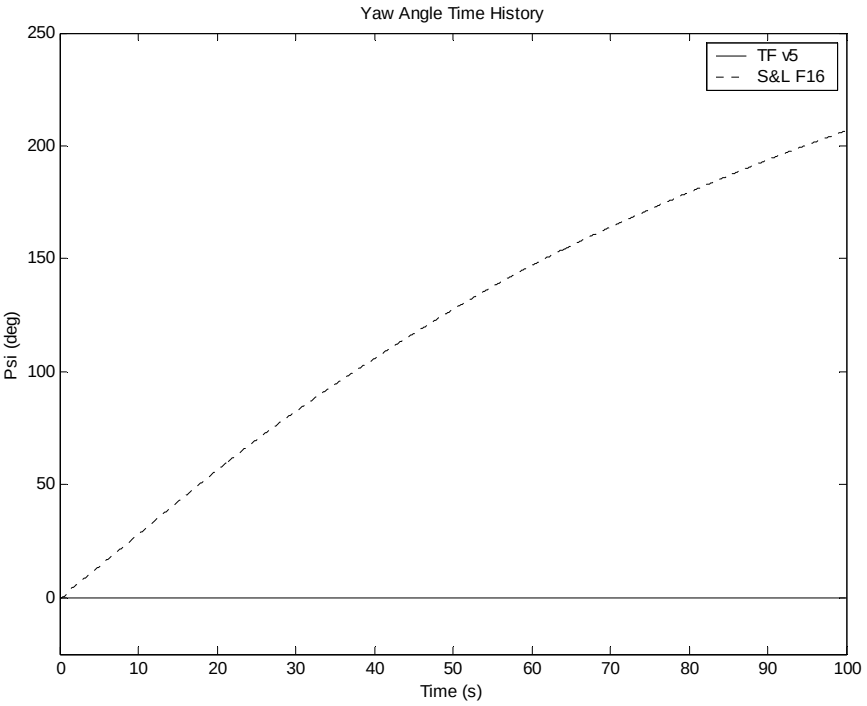

Figure 4.64 – Time History of Yaw Angle for the original Uncontrolled Flight Test

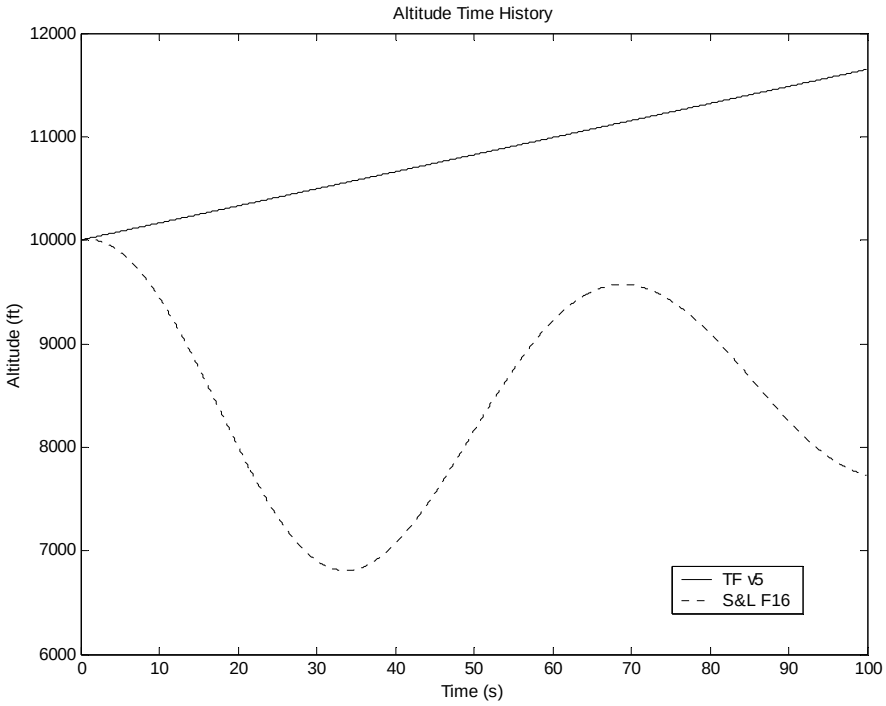

Figure 4.65 – Time History of Altitude for the original Uncontrolled Flight Test

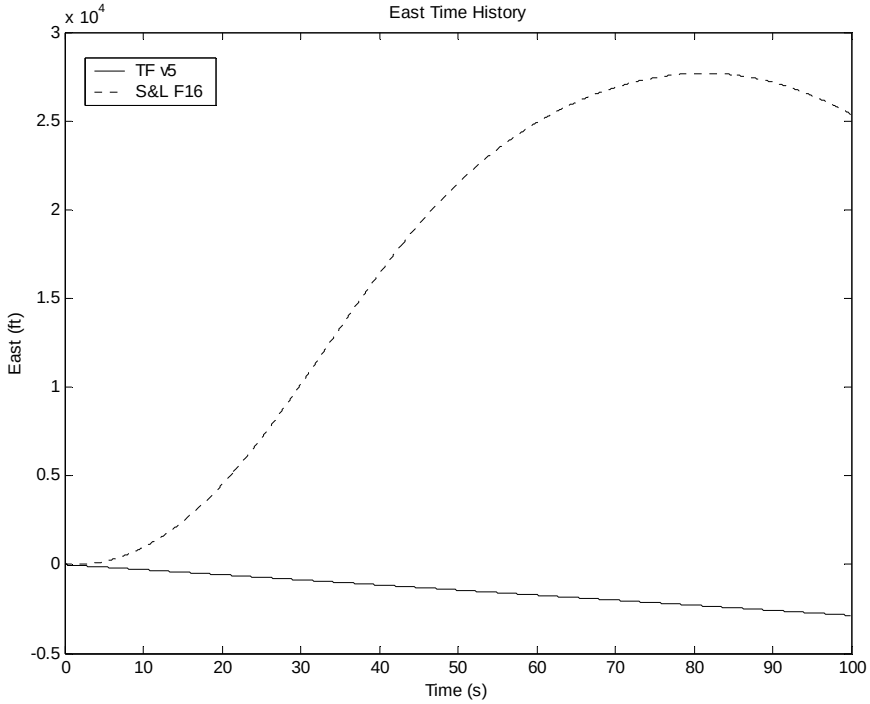

Figure 4.66 – Time History of East Position for the original Uncontrolled Flight Test

4.5 Uncontrolled Flight with Gravity: Banked Initial Condition

This simulation used the same initial conditions and control deflection time history as the Uncontrolled Flight in Section 4.4. The only difference was that the simulation contained an attempt to add gravity to the aircraft's angular rate responses. The wind-axis angular rates shown below have gravity explicitly in them.

$$Q_W = \frac{1}{mV_T}\left[L - mg\cos\mu\cos\gamma - T\sin\left(\varepsilon_T - \alpha\right)\right] \tag{4.1}$$

$$R_W = \frac{1}{mV_T}\left[-C + mg\sin\mu\cos\gamma - T\sin\beta\cos\left(\varepsilon_T - \alpha\right)\right] \tag{4.2}$$

In the wind-axis equations, L is the lift force, D is the drag force, C is the side force, T is the thrust, ε_T is the thrust angle in the aircraft's x-z plane, and μ, γ, and χ are the wind-axis angles. The TFSM incorporates its thrust force into the lift, drag, and side forces, because it does not have an actual thrust model; this causes all of the thrust terms to disappear. The wind-axis kinematic equations calculate the rates for the wind-axis angles (equations 4.3 – 4.5), which are then integrated like the other states in the code.

$$\dot{\mu} = P_W + \left(Q_W\sin\mu + R_W\cos\mu\right)\tan\gamma \tag{4.3}$$

$$\dot{\gamma} = Q_W\cos\mu - R_W\sin\mu \tag{4.4}$$

$$\dot{\chi} = \left(Q_W\sin\mu + R_W\cos\mu\right)\sec\gamma \tag{4.5}$$

Adding the above six equations to the TFSM enabled it to generate new angular rates that contain the effects of gravity.

Directly adding the wind-axis rates to the rates calculated by the transfer functions generated the new angular rates presented in Figures 4.67, 4.68, and 4.69. By no means did the new rates match the F-16's, but their initial responses were much more realistic. The roll-rate started at approximately -0.2 degrees/sec and gradually approached zero at approximately the same rate as the slowly oscillating non-linear model's roll-rate. Similarly, the yaw-rate began at 3.2 degrees/sec before it decreased to zero. The F-16's response took a much longer time to approach zero as it oscillated. The pitch-rate response was the least accurate. Both models' responded with an initial jump in pitch-rate and then decreased afterwards. Yet the TFSM's response was a direct descent to zero without any oscillations and the F-16's long-term oscillation and slower approach to zero had an initial of amplitude of 1.25 degrees/sec. The

responses for total velocity, angle of attack, and sideslip angles were the same as found in Figures 4.56 – 4.58. They did not change because the state-space representations for the transfer functions still calculated them directly; therefore, the wind-axis equations had no influence. The aircraft's remaining states however, changed drastically with the addition of the wind-axis equations. Like the new angular rates, the aircraft angles and positions had appropriate initial responses, although they did not match the F-16 responses. Initially, the roll and pitch angles (Figures 4.70 and 4.71) fell off like the F-16, but then leveled out at a different angle instead of oscillating. Without non-zero pitch- and yaw-rates, the TFSM could not continue its turn. Hence, its heading angle (Figure 4.72) increased initially until about 20 seconds where it leveled off and then began to differ greatly from the F-16's as it continued to turn. Because of these angles and rates, the TFSM's altitude (Figure 4.73) matched initially before it decreased resulting from the large negative pitch angle. The TFSM's east position (Figure 4.74) followed the heading angle closely so that it eventually leveled out instead of increasing continuously.

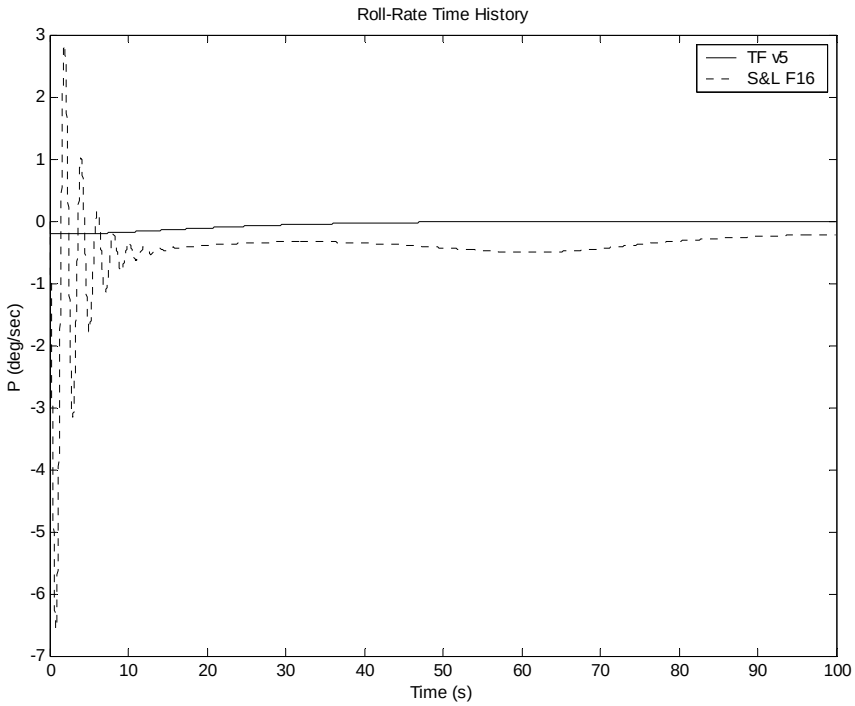

Figure 4.67 – Time History of Roll-Rate for the improved Uncontrolled Flight Test

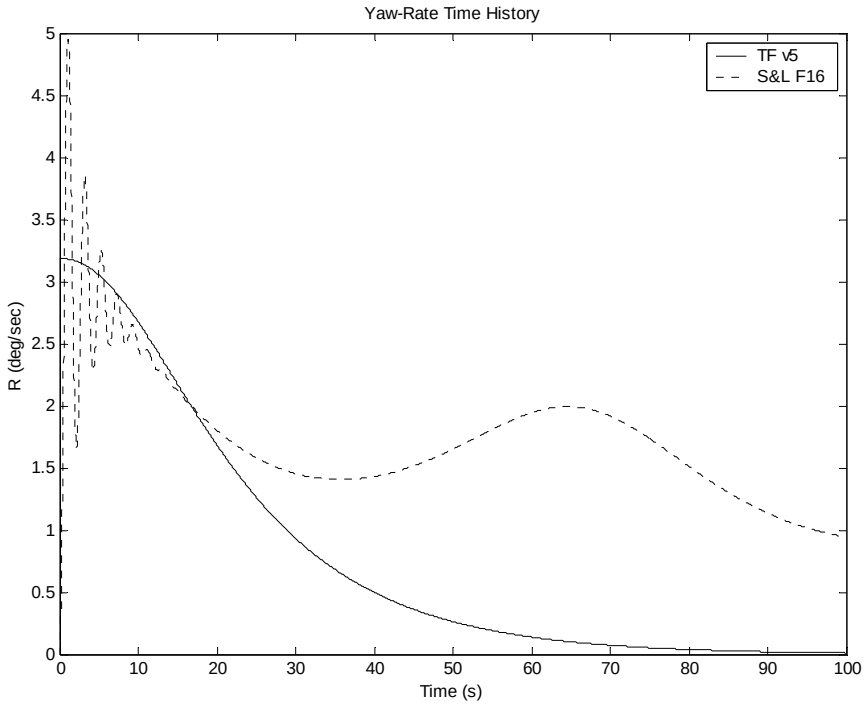

Figure 4.68 – Time History of Yaw-Rate for the improved Uncontrolled Flight Test

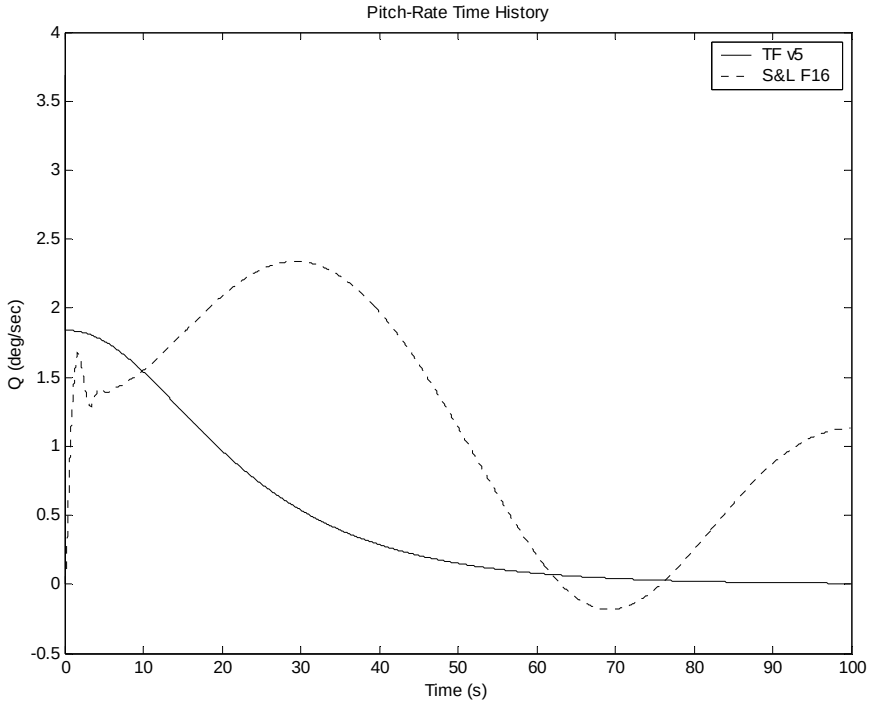

Figure 4.69 – Time History of Pitch-Rate for the improved Uncontrolled Flight Test

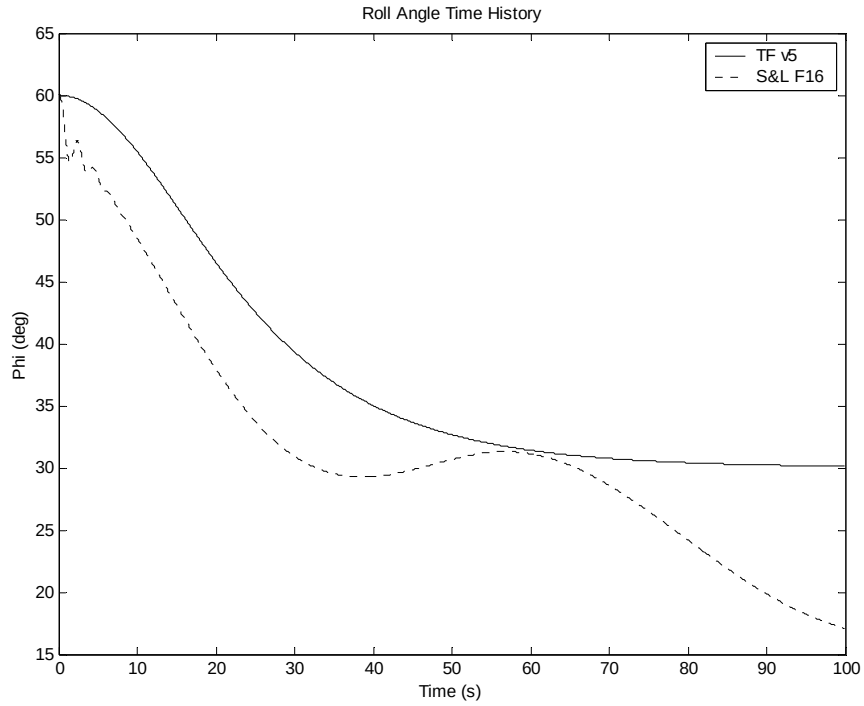

Figure 4.70 – Time History of Roll Angle for the improved Uncontrolled Flight Test

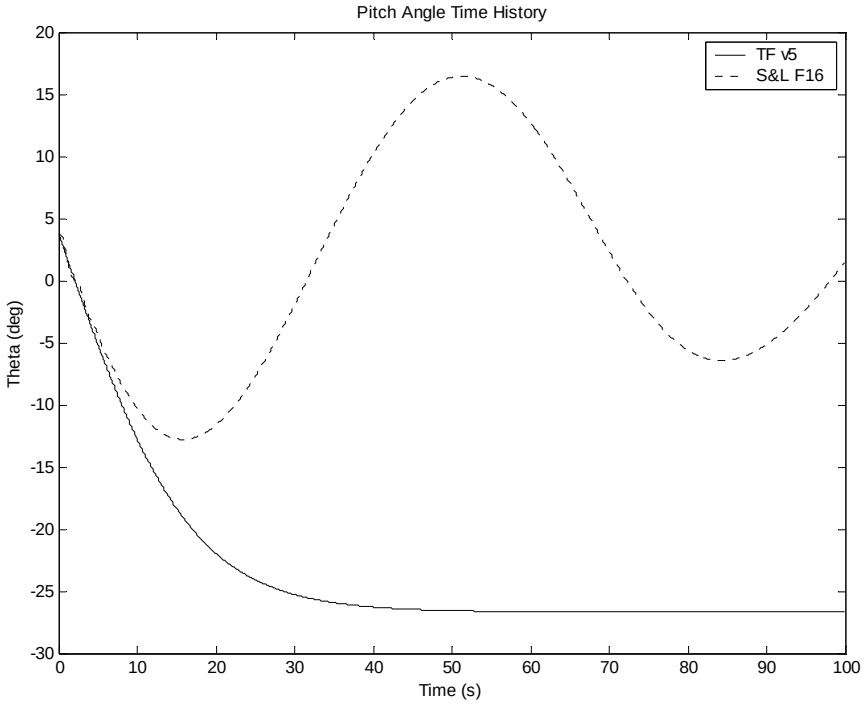

Figure 4.71 – Time History of Pitch Angle for the improved Uncontrolled Flight Test

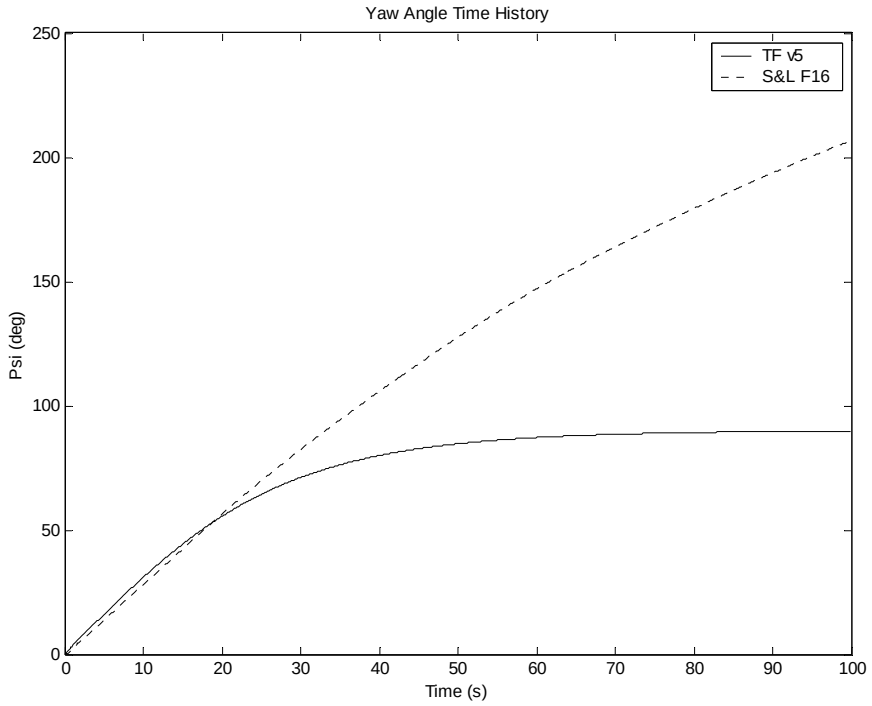

Figure 4.72 – Time History of Yaw Angle for the improved Uncontrolled Flight Test

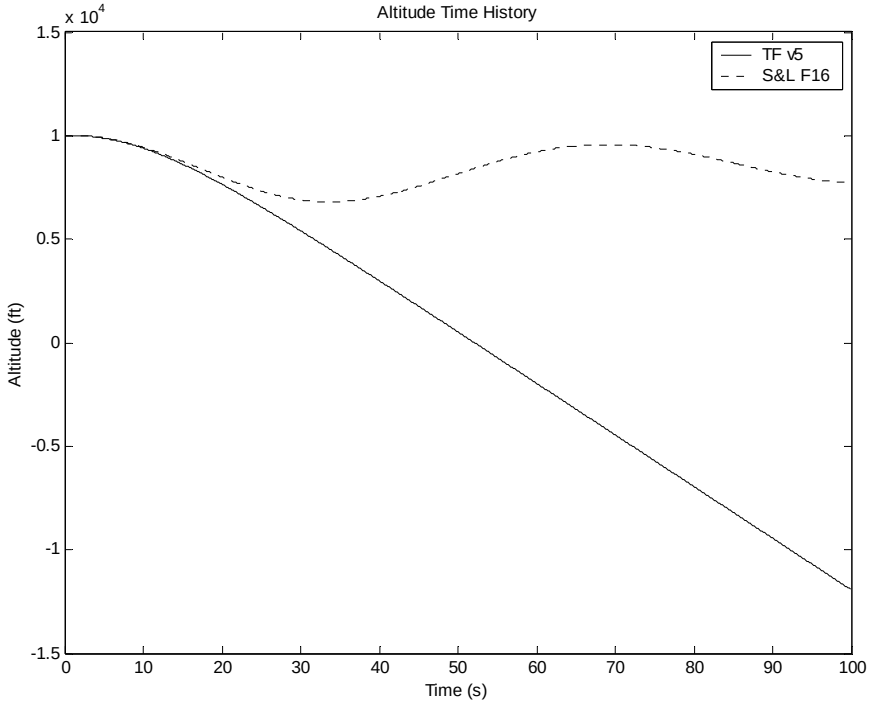

Figure 4.73 – Time History of Altitude for the improved Uncontrolled Flight Test

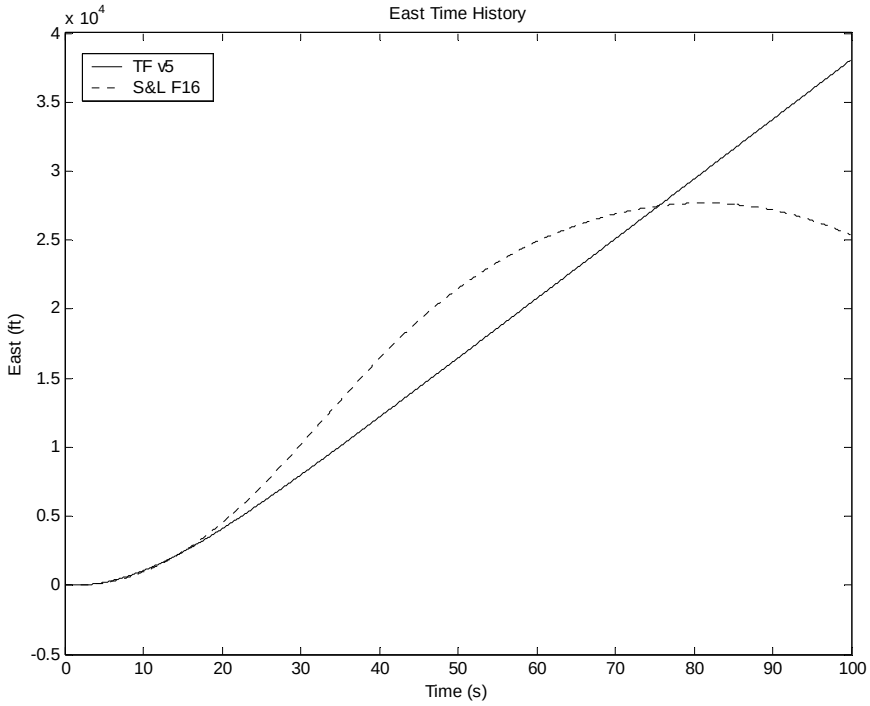

Figure 4.74 – Time History of East Position for the improved Uncontrolled Flight Test

Chapter 5

Simulation Analysis

5.1 Introduction

This chapter discusses the data presented throughout Chapter 4 and its relationship to the linear Transfer Function Simulation Model (TFSM). All comparisons made to validate the TFSM's behavior are with respect to the non-linear Stevens and Lewis F-16 simulation.

5.2 The Analysis

A linear simulation model is expected to have a certain amount of error when compared to the original non-linear model. The responses to throttle and elevator controls presented in Chapter 4 exhibited an acceptable amount of error for a linearized model. All the aircraft state responses were either negligible, in the case of the lateral-directional states, or had the correct shape and similar magnitudes. Those responses show that the TFSM can simulate an aircraft with longitudinal control inputs alone. However, no realistic flight scenario requires solely longitudinal control deflections; there are always at least minor lateral-directional inputs. As stated at the beginning of this thesis, there is a major problem in the linear TFSM – the model

fails to simulate the appropriate longitudinal flight characteristics during lateral-directional maneuvers.

Following each longitudinal control section in Chapter 4, is the corresponding lateral-directional control section. Similar to the longitudinal responses to elevator and throttle inputs, the lateral-directional responses to aileron and rudder deflections were acceptable as well. In most cases, the responses were almost identical to the proof model. The changes in the longitudinal states, with wings level, do not affect the lateral-directional states. However, changes in the lateral-directional states, like bank angle and roll-rate, trigger disturbances in the longitudinal states. Pitching a normal aircraft will not cause it to roll or yaw during all flight conditions; rolling an aircraft will cause the aircraft's nose to drop, causing a loss in altitude and an increase in speed. This phenomenon happens in both lateral-directional control cases presented in Chapter 4. The kinematic, force, moment, and navigation equations found in Chapter 2 explicitly show the relationships between the roll angle and the longitudinal state derivatives (equations 5.1 – 5.5). For example, by introducing a bank angle, the pitch angle's time derivative, $\dot{\theta}$, is no longer equal to the pitch-rate, Q. The pitch-rate's influence decreases and the yaw-rate's influence increases as the bank angle grows (equation 5.1).

$$\dot{\theta} = Q\cos\phi - R\sin\phi \tag{5.1}$$

$$\dot{Q} = \frac{1}{I_{yy}}\left[M - \left(I_{xx} - I_{zz}\right)PR - I_{xz}\left(P^2 - R^2\right)\right] \tag{5.2}$$

$$\dot{U} = RV - QW - g\sin\theta + \frac{X}{m} \tag{5.3}$$

$$\dot{V} = -RU + PW + g\sin\phi\cos\theta + \frac{Y}{m} \tag{5.4}$$

$$\dot{W} = QU - PV + g\cos\phi\cos\theta + \frac{Z}{m} \tag{5.5}$$

Equations 5.2 through 5.5, along with equations 2.8 and 2.9, govern the dynamics of most simulations that calculate the forces and moments, and then generate the states. However, the TFSM uses transfer functions to calculate the aircraft's velocity, angle of attack, sideslip angle, and its roll-, pitch-, and yaw-rates. The reference flight condition decouples the two modes during linearization. Therefore, the TFSM's longitudinal states are unaffected by its lateral-

directional states. Consequently, the pitch-rate does not depend on the roll- and yaw-rates; the roll angle and the roll- and yaw-rates do not affect the total velocity or angle of attack. Since there is no interdependence between the angular rates, the TFSM cannot generate the desired rates for turning flight. Specifically, for turning level flight:

$$\dot{\phi} = P + \tan\theta\left(Q\sin\phi + R\cos\phi\right) = 0 \tag{5.6}$$

$$\dot{\theta} = Q\cos\phi - R\sin\phi = 0 \tag{5.7}$$

$$\dot{\psi} = \frac{Q\sin\phi + R\cos\phi}{\cos\theta} = \text{turn rate} \tag{5.8}$$

Equation 5.8 indicates that some combination of non-zero pitch- and yaw-rates determine the turn rate. To continue a steady level turn, the turn rate must remain constant. However, the steady-state values for the rates in the TFSM are zero, so it cannot maintain a level turn with constant control inputs. The TFSM seeks a new trim condition based on the control deflection as each response goes towards its steady state value. In order to maintain the constant turn rate using the TFSM, the control deflections would have to be increased enough to generate the desired angular rates for the level turn. Unfortunately this is unintuitive to a pilot who expects to get a constant angular rate out of a given control deflection.

The lateral-directional control response for the *S*-turn flight maneuver in Section 4.3.4 illustrates this inherent part of the TFSM. The initiation of the bank angle at 15 seconds immediately causes the aircraft's nose to drop, and in turn, decrease its altitude. The F-16, which obeys the coupled laws of flight dynamics, also picks up speed at this time. The Principle of Conservation of Energy provides a quick analysis for this reaction. A simplified version of the principle states that the potential energy (PE) and the kinetic energy (KE) at one state must equal the potential and kinetic energy at the next state with the addition of work energy (W), which incorporates energy losses such as friction and heat (equation 5.9).

$$PE_0 + KE_0 = PE_1 + KE_1 + W \tag{5.9}$$

$$PE = mgh \tag{5.10}$$

$$KE = \frac{1}{2}mV_T^2 \tag{5.11}$$

The above definition for kinetic energy incorporates only the translational energy and ignores angular rates (rotational kinetic energy). Combining the three equations leads to the relationship that a change in altitude causes a change in velocity, minus the losses represented in W.

Therefore, a stable aircraft continues to lose altitude until it gains enough speed for its restoring pitching moment to increase sufficiently to bring the aircraft's nose up and begin climbing. The increasing altitude causes the aircraft to lose speed until it can no longer maintain the climb, and it begins to dive again. Figures 4.22 and 4.23 show that the F-16 obeys the physics discussed previously, and that the TFSM does not. The TFSM cannot form the desired response because its velocity and pitch-rate are constrained by the transfer functions, which only change with throttle and elevator inputs. Because the velocity cannot increase and there is no restoring pitch-rate, the altitude and pitch angle continue to fall. This energy analysis shows that the longitudinal responses to lateral-directional controls are incorrect, but are inherent in the linearization used to formulate the TFSM.

Section 4.4 shows that the lateral-directional controls do not cause the undesired longitudinal responses, but the lateral-directional states do. Including the initial condition of a 60-degree bank angle in an uncontrolled simulation isolated the states from the controls. The figures in Section 4.4 illustrate that even without lateral-directional controls, the simulation still does not have the desired behavior. One reason for this is that the TFSM does not have any gravity terms in its calculations. Assuming the changes in the forces and moments due to the control surfaces are negligible, the gravitational force's orientation relative to the aircraft's body-axis is the only non-aerodynamic force or moment that changes. The F-16 behaved as expected; its nose dropped, allowing the aircraft to lose altitude and gain speed. However, the TFSM continued to fly with its states remaining at their initial conditions. Its altitude and east position were the only states that did not remain constant, but their rate of change was constant. The navigation equations presented in Section 2.2 show why the bank angle and the other constant states cause the aircraft to climb and drift to the east. The model described in Section 4.5 offers a partial solution to the gravity problem by using the wind-axis angular rates to generate new roll-, pitch-, and yaw-rates in conjunction with the TFSM's existing angular rates. This enabled the TFSM's initial response to behave like the F-16.

The gravity terms helped simulate the initial yaw-rate generation in the banked initial condition flight test. Equation 5.12 presents the yaw-rate's time derivative.

$$\dot{R} = \frac{I_{xz}}{I_{xx}I_{zz} - I_{xz}^2}\left[L + I_{xz}PQ - \left(I_{zz} - I_{yy}\right)QR\right] + \frac{I_{xx}}{I_{xx}I_{zz} - I_{xz}^2}\left[N - I_{xz}QR - \left(I_{yy} - I_{xx}\right)PQ\right] \qquad (5.12)$$

In order to generalize the above equation, the analysis was performed in the Principal Axis coordinate system. Therefore, the non-principal moments of inertia, I_{xy}, I_{xz}, and I_{yz}, were zero, resulting in equation 5.13.

$$\dot{R} = \frac{1}{I_{zz}}\left[N - \left(I_{yy} - I_{xx}\right)PQ\right] \tag{5.13}$$

At the beginning of the flight test, the angular rates (P and Q) were zero, which left the yaw-rate's derivative as purely a function of the yawing moment (N). An aircraft's yawing moment is a function of the aircraft's geometry, the dynamic pressure at the flight condition, and the yawing moment coefficient. The yawing moment coefficient is dependent on the derivative coefficients for the sideslip angle, the non-dimensional roll- and yaw-rates, and the aileron and rudder deflections. [5] As stated previously, the angular rates were zero at the start of the flight test, and therefore, did not affect the yawing moment. The aileron deflection is ignored because its affect on the yawing moment is dependent on the sign of its coefficient in the yawing moment coefficient's buildup. Ailerons generate either proverse or adverse yaw depending on the sign of their coefficient; therefore, to generalize the analysis, it is assumed to be zero. Otherwise, this analysis would become aircraft specific, which is not appropriate for the variable stability TFSM. The rudder deflection is also ignored because it is not required to turn a directionally stable aircraft. An aircraft is directionally stable when its sideslip angle coefficient in the yawing moment coefficient buildup ($C_{n\beta}$) is positive – this means that a positive sideslip angle generates a positive yawing moment. Initially, the yaw-rate's time derivative is a function of the change in the sideslip angle; this is because the derivative of the yaw-rate depends only on the yawing moment coefficient and is solely a function of the sideslip angle. The component of gravity in the direction of the y-axis (body-axis coordinate system), created by the bank angle, generated a y-directional acceleration because gravity's component in the y-direction was not balanced by the aerodynamic and propulsive forces. When integrated, gravity's acceleration along the y-axis created a sideways velocity that caused the change in the sideslip angle. Therefore, the change in sideslip angle produced the change in the yaw-rate's time derivative.

At the initial bank angle, gravity caused a change in the sideslip angle, which in turn generated the yaw-rate experienced by the F-16. The TFSM did not produce the change in sideslip angle during the uncontrolled flight because the sideslip angle was only affected by the control deflections and not by the other states. As seen in the plots in Section 4.5, adding the

wind-axis rates to the TFSM helped it emulate the initial change in yaw-rate. Unfortunately, after the first five to ten seconds, the errors in total velocity, angle of attack, and sideslip angle, (of which the wind-axis equations did not affect) became very large. Even with the changes to the angular rates, the simulation failed to perform as desired. In another pilot-in-the-loop flight test, the TFSM's simulation seemed, realistic although it did not follow the exact trajectory of the non-linear model. In summary, the changes to the initial responses made by the wind-axis rates indicate that adding gravity explicitly to the simulation would help its fidelity.

The results in Chapter 4 and the discussion here demonstrate that the TFSM can simulate the flight of an aircraft with longitudinal controls alone, but not with lateral-directional controls. The transfer function realizations within the TFSM do not allow for the desired coupling between the lateral-directional states and the longitudinal states. For small lateral-directional rates and angles, the transfer functions provide a decent approximation to the aircraft's flight characteristics. Unfortunately, the pitch angle and altitude calculations integrate the approximation's error into themselves at each time step. Even when the responses were very good, the error still built up during the simulation (see Figures 4.37 and 4.39). Obviously, limiting the amount of lateral-directional control improves the simulation, but the TFSM still fails to emulate the basis aircraft fully when lateral-directional controls are used.

Chapter 6

Summary and Conclusions

This thesis investigated the lateral-directional coupling in the longitudinal responses of the linear variable stability Transfer Function Simulation Model (TFSM). The TFSM can represent any aircraft by changing the gain constants, time constants, damping ratios, and natural frequencies that model a linearized aircraft. Previous flight tests using the TFSM indicated undesired coupling between the longitudinal and lateral-directional modes after the initiation of lateral-directional maneuvers. The purpose of this thesis was to examine this phenomenon and identify the problems within the TFSM that caused the undesired coupling. The Stevens and Lewis F-16 simulation model was the verification model used during the investigation. An *S*-turn maneuver, formation flight test, and an uncontrolled simulation with an initial bank angle of 60 degrees were the foundation for the analysis previously presented, to pinpoint the TFSM's errors. The flight tests and subsequent analysis showed that although this model is highly versatile, it has three fundamental problems.

First, the original creation of the TFSM assumed that the time rate of change for the pitch angle was equal to the body-axis pitch-rate. This is only correct for wings level flight; once a bank angle is present, this assumption is invalid. The aircraft body-axis angles (roll, pitch, and yaw) were integrated using the flight dynamics' kinematic equations instead of assuming the angles' time derivatives equaled the body-axis angular rates.

Second, the TFSM does not have gravity terms. Linearizing the model about wings level flight removes the gravity terms from all lateral-directional states, and therefore, effectively removes it from the longitudinal states as well. Adding gravity terms to the angular rates by using wind-axis angular rates showed promise during the initial testing, and would be a good place to start for future work to improve the TFSM.

Third, the TFSM cannot generate the angular rates needed in a turn. Linearizing about the given reference conditions removed the interconnections between the individual angular rates as seen in the body-axis moment equations. Particularly, the steady state values for the rates generated from transfer functions are always zero, but turning level flight requires non-zero rates. Because this problem is inherent in the transfer functions, taking the angular rate transfer function realizations out, and thus destroying the TFSM, is the only way to fix this problem.

Even though the TFSM's longitudinal responses are flawed due to the lack of influence by the lateral-directional states, it still has promise for future work. Adding fixes like the addition of the wind-axis rates could improve its accuracy and behavior at flight conditions away from the reference. Another possible solution would be to linearize the non-linear aircraft about several flight conditions (like turning flight) and have the TFSM interpolate between the different linearizations based on the current flight condition. In addition, it could investigate each mode individually by splitting the TFSM into two models used to simulate either the longitudinal or the lateral-directional dynamics. For instance, the model analyzing the longitudinal dynamics would have normal non-linear lateral-directional dynamics and transfer functions representing the longitudinal variables. This would allow unlimited versatility in the longitudinal modes without affecting the lateral-directional modes, and vice versa for the lateral-directional model.

Bibliography

[1] Pettersson, Henrik B., *Variable Stability Transfer Function Simulation*, M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA 2002.

[2] Minor, John, Thurling, Andrew J., Ohmit, Eric, "VISTA-A 21st Century UAV Testbed", *United States Air Force Test Pilot School website*, 10 May 2001, <http://www.edwards.af.mil/tps/vista/VISTA-21st%20Century%20UAV%20Testbed.pdf>, (27 October 2003).

[3] Anon, Military Specification, Flying Qualities of Piloted Airplanes, MIL-F-8785C, November 1980.

[4] Johnson, Donald A., Captain, USAF, *Suppression of Pilot-Induced Oscillations (PIO)*, M.S. Thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio 2002.

[5] Durham, W. C., "AOE 5214: Aircraft Dynamics and Control", *Class notes from Fall 2002*, Aerospace and Ocean Engineering Department, Virginia Polytechnic Institute and State University, 2002.

[6] Stevens, Brian L. and Frank L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, Inc., New York, 1992.

[7] Ogata, Katsuhiko, *System Dynamics, Second Edition*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1992, pp. 175-177.

[8] Nichols, James H., Magyar, Thomas J. and Schug, Eric C., "The Platform-Independent Aircraft Simulation Environment at Manned Flight Simulator", AIAA-98-4179, 1998.

[9] Scalera, Kevin R., and Durham, W. C., "Modification of a Surplus Navy 2F122A A-6E OFT for Flight Dynamics Research and Instruction", AIAA-98-4180, 1998.

[10] "Research – Facilities – Full Motion Based Piloted Flight Simulator", *Virginia Tech Department of Aerospace and Ocean Engineering Website*, 14 July 2003, <http://www.aoe.vt.edu/research/facilities/simfacility.php>, (7 November 2003).

[11] Etkin, Bernard, *Dynamics of Atmospheric Flight*, John Wiley & Sons, Inc., New York, 1972.

Appendix A

MATLAB TFSM Generation Code

A.1 Description

This appendix includes two sets of MATLAB code – one to calculate the longitudinal and the other to determine the lateral-directional characteristics. The two sets of code are fundamentally the same, but have different variables associated with the different modes, so only the longitudinal code is fully explained. LINEARIZE (a Stevens and Lewis F-16 subroutine) automatically generates the Jacobian matrices and outputs them to a MATLAB file similar to the ones shown in Sections B.2 and B.4. The rest of the code processes the constant A, B, C, and D matrices. MATLAB builds the state-space equations, generates the corresponding transfer functions, and then creates a new set of state-space equations that map the original outputs to the new internal states associated with either the longitudinal or the lateral-direction modes. The second section of code uses the MATLAB command *SS* to recreate the state-space equations in MATLAB. SS takes the system matrices A, B, C, and D as arguments and creates a state-space system in MATLAB's notation. The command *TF* takes a state-space system as input and generates the transfer function matrix associated with that system. Giving *TF* this system creates all of the transfer functions relating the outputs to the inputs. Over the next several lines, the transfer functions within the matrix are given individual names to make the code easier to read. Next, each transfer function's numerator and denominator are stored as separate variables so

MATLAB can calculate the associated poles and zeros. The *DAMP* command determines the poles in the denominator of every longitudinal transfer function since *DLONG* is their common denominator. *DAMP* then calls each numerator to get the zeros for the longitudinal transfer functions. After calculating all of the poles and zeros, the each transfer function recreates a corresponding state-space system representation. *TF2SS* is the MATLAB command that takes a transfer function's numerator and denominator, in polynomial form, and returns the A, B, C, and D matrices. The longitudinal systems for each aircraft state transfer function have identical A and B matrices when the state-space equations are recreated. The only difference between each system is the C matrix, since all D matrices are zero. The C matrix ends up being the coefficients of the numerator for each transfer function (a one by four matrix). Therefore, only one such *TF2SS* call is made for all the longitudinal transfer functions.

A.2 Longitudinal Code

```
Clear; clc;
% States: V, alpha, theta, Q (all angles in radians)
A = [ -0.171623E-01 -0.388575E+01 -0.321696E+02 -0.110962E+01
 -0.255092E-03 -0.750614E+00  0.000000E+00  0.927837E+00
  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
 -0.742464E-12 -0.427826E+01  0.000000E+00 -0.126121E+01
];
% Controls: throttle, elevator
B = [  0.197222E+02  0.984902E-01
 -0.261203E-02 -0.158954E-02
  0.000000E+00  0.000000E+00
  0.000000E+00 -0.138577E+00
];
% Outputs: V, alpha, Q
C = [  0.100000E+01  0.000000E+00  0.000000E+00  0.000000E+00
  0.000000E+00  0.100000E+01  0.000000E+00  0.000000E+00
  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
];
D = [  0.000000E+00  0.000000E+00
  0.000000E+00  0.000000E+00
  0.000000E+00  0.000000E+00
];

sysss=ss(A,B,C,D);    % state-space system
systf=tf(sysss)       % transfer functions
UDT = systf(1,1);
ADT = systf(2,1);
QDT = systf(3,1);
UDE = systf(1,2);
ADE = systf(2,2);
QDE = systf(3,2);

[NUDT,DLONG] = tfdata(UDT,'v');
[NADT,DLONG] = tfdata(ADT,'v');
[NQDT,DLONG] = tfdata(QDT,'v');
[NUDE,DLONG] = tfdata(UDE,'v');
[NADE,DLONG] = tfdata(ADE,'v');
[NQDE,DLONG] = tfdata(QDE,'v');

fprintf('\n\n\nLongitudinal Denominator')
damp(DLONG)
fprintf('\n\n\nVelocity to Throttle Numerator')
damp(NUDT)
fprintf('\n\n\nAlpha to Throttle Numerator')
damp(NADT)
fprintf('\n\n\nQ to Throttle Numerator')
damp(NQDT)
fprintf('\n\n\nVelocity to Elevator Numerator')
damp(NUDE)
fprintf('\n\n\nAlpha to Elevator Numerator')
damp(NADE)
fprintf('\n\n\nQ to Elevator Numerator')
damp(NQDE)

[Ap, Bp, Cp, Dp] = tf2ss(NUDE,DLONG)
```

A.3 Sample Longitudinal Output (Formatted)

```
Transfer function from input 1 to output...
          19.72 s^3 + 39.69 s^2 + 96.96 s - 0.3595
 #1:  ---------------------------------------------
      s^4 + 2.029 s^3 + 4.95 s^2 + 0.08433 s + 0.03511

      -0.002612 s^3 - 0.00837 s^2 - 0.006402 s + 6.239e-014
 #2:  -----------------------------------------------------
        s^4 + 2.029 s^3 + 4.95 s^2 + 0.08433 s + 0.03511

             0.01117 s^2 + 0.02172 s - 7.534e-020
 #3:  ---------------------------------------------
      s^4 + 2.029 s^3 + 4.95 s^2 + 0.08433 s + 0.03511

Transfer function from input 2 to output...
         0.09849 s^3 + 0.3581 s^2 + 5.557 s + 3.127
 #1:  ---------------------------------------------
      s^4 + 2.029 s^3 + 4.95 s^2 + 0.08433 s + 0.03511

      -0.00159 s^3 - 0.1306 s^2 - 0.002312 s - 0.001137
 #2:  -------------------------------------------------
      s^4 + 2.029 s^3 + 4.95 s^2 + 0.08433 s + 0.03511

      -0.1386 s^3 - 0.0996 s^2 - 0.001424 s + 2.552e-019
 #3:  --------------------------------------------------
       s^4 + 2.029 s^3 + 4.95 s^2 + 0.08433 s + 0.03511


Longitudinal Denominator
          Zeros               Damping     Freq. (rad/s)
 -7.12e-003 + 8.42e-002i     8.42e-002      8.45e-002
 -7.12e-003 - 8.42e-002i     8.42e-002      8.45e-002
 -1.01e+000 + 1.97e+000i     4.54e-001      2.22e+000
 -1.01e+000 - 1.97e+000i     4.54e-001      2.22e+000

Velocity to Throttle Numerator
          Zeros               Damping     Freq. (rad/s)
  3.70e-003                 -1.00e+000      3.70e-003
 -1.01e+000 + 1.98e+000i     4.54e-001      2.22e+000
 -1.01e+000 - 1.98e+000i     4.54e-001      2.22e+000


Alpha to Throttle Numerator
    Zeros        Damping     Freq. (rad/s)
  9.75e-012    -1.00e+000      9.75e-012
 -1.26e+000     1.00e+000      1.26e+000
 -1.94e+000     1.00e+000      1.94e+000


Q to Throttle Numerator
    Zeros        Damping     Freq. (rad/s)
  3.47e-018    -1.00e+000      3.47e-018
 -1.94e+000     1.00e+000      1.94e+000
```

```
Velocity to Elevator Numerator
          Zeros                Damping     Freq. (rad/s)
 -5.81e-001                  1.00e+000      5.81e-001
 -1.53e+000 + 7.23e+000i     2.07e-001      7.39e+000
 -1.53e+000 – 7.23e+000i     2.07e-001      7.39e+000


Alpha to Elevator Numerator
          Zeros                Damping     Freq. (rad/s)
 -8.80e-003 + 9.29e-002i     9.43e-002      9.33e-002
 -8.80e-003 – 9.29e-002i     9.43e-002      9.33e-002
 -8.22e+001                  1.00e+000      8.22e+001


Q to Elevator Numerator
    Zeros        Damping     Freq. (rad/s)
  1.79e-016    -1.00e+000      1.79e-016
 -1.46e-002     1.00e+000      1.46e-002
 -7.04e-001     1.00e+000      7.04e-001


Ap = -2.0290   -4.9497   -0.0843   -0.0351
      1.0000         0         0         0
           0    1.0000         0         0
           0         0    1.0000         0

Bp = 1
     0
     0
     0

Cp = 0.0985    0.3581    5.5574    3.1274

Dp = 0
```

A.4 Lateral-Directional Code

```
clear
clc
% States: beta, phi, P, R (all angles in radians)
A = [ -0.237179E+00  0.641986E-01  0.663105E-01 -0.991989E+00
  0.000000E+00  0.000000E+00  0.100000E+01  0.662205E-01
 -0.255333E+02  0.000000E+00  -0.266339E+01  0.590571E+00
  0.769230E+01  0.000000E+00  -0.394986E-01 -0.385094E+00
];
% Controls: aileron, rudder
B = [  0.217329E-03  0.593344E-03
  0.000000E+00  0.000000E+00
 -0.541536E+00  0.938229E-01
 -0.241491E-01 -0.489280E-01
];
% Outputs: beta, P, R
C = [  0.100000E+01  0.000000E+00  0.000000E+00  0.000000E+00
  0.000000E+00  0.000000E+00  0.100000E+01  0.000000E+00
  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
];
D = [  0.000000E+00  0.000000E+00
  0.000000E+00  0.000000E+00
  0.000000E+00  0.000000E+00
];

sysss=ss(A,B,C,D);      % state-space system
systf=tf(sysss)         % transfer functions
BDA = systf(1,1);
PDA = systf(2,1);
RDA = systf(3,1);
BDR = systf(1,2);
PDR = systf(2,2);
RDR = systf(3,2);

[NBDA,DLAT] = tfdata(BDA,'v');
[NPDA,DLAT] = tfdata(PDA,'v');
[NRDA,DLAT] = tfdata(RDA,'v');
[NBDR,DLAT] = tfdata(BDR,'v');
[NPDR,DLAT] = tfdata(PDR,'v');
[NRDR,DLAT] = tfdata(RDR,'v');
fprintf('\n\n\nLateral-Directional Denominator')
damp(DLAT)
fprintf('\n\n\nBeta to Aileron Numerator')
damp(NBDA)
fprintf('\n\n\nP to Aileron Numerator')
damp(NPDA)
fprintf('\n\n\nR to Aileron Numerator')
damp(NRDA)
fprintf('\n\n\nBeta to Rudder Numerator')
damp(NBDR)
fprintf('\n\n\nP to Rudder Numerator')
damp(NPDR)
fprintf('\n\n\nR to Rudder Numerator')
damp(NRDR)

[Ap, Bp, Cp, Dp] = tf2ss(NBDA,DLAT)
```

A.5 Sample Lateral-Directional Output (Formatted)

```
Transfer function from input 1 to output...
      0.0002173 s^3 – 0.01129 s^2 – 0.00683 s – 0.01449
 #1:  ---------------------------------------------
        s^4 + 3.286 s^3 + 11.1 s^2 + 23.53 s + 0.2482

      -0.5415 s^3 – 0.3568 s^2 – 4.798 s + 0.02033
 #2:  -----------------------------------------
      s^4 + 3.286 s^3 + 11.1 s^2 + 23.53 s + 0.2482

      -0.02415 s^3 – 0.04698 s^2 – 0.3226 s – 0.307
 #3:  -----------------------------------------
      s^4 + 3.286 s^3 + 11.1 s^2 + 23.53 s + 0.2482

Transfer function from input 2 to output...
      0.0005933 s^3 + 0.05657 s^2 + 0.1399 s – 0.0001053
 #1:  -------------------------------------------------
        s^4 + 3.286 s^3 + 11.1 s^2 + 23.53 s + 0.2482

      0.09382 s^3 + 0.01434 s^2 – 0.5248 s + 0.002243
 #2:  -----------------------------------------
       s^4 + 3.286 s^3 + 11.1 s^2 + 23.53 s + 0.2482

      -0.04893 s^3 – 0.1411 s^2 – 0.05402 s – 0.03387
 #3:  -----------------------------------------
       s^4 + 3.286 s^3 + 11.1 s^2 + 23.53 s + 0.2482


Lateral-Directional Denominator
          Zeros               Damping     Freq. (rad/s)
 -1.06e-002                 1.00e+000      1.06e-002
 -3.65e-001 + 3.01e+000i    1.20e-001      3.03e+000
 -3.65e-001 – 3.01e+000i    1.20e-001      3.03e+000
 -2.54e+000                 1.00e+000      2.54e+000


Beta to Aileron Numerator
          Zeros               Damping     Freq. (rad/s)
  5.26e+001                -1.00e+000      5.26e+001
 -3.11e-001 + 1.08e+000i    2.76e-001      1.13e+000
 -3.11e-001 – 1.08e+000i    2.76e-001      1.13e+000


P to Aileron Numerator
          Zeros               Damping     Freq. (rad/s)
  4.24e-003                -1.00e+000      4.24e-003
 -3.32e-001 + 2.96e+000i    1.11e-001      2.98e+000
 -3.32e-001 – 2.96e+000i    1.11e-001      2.98e+000


R to Aileron Numerator
          Zeros               Damping     Freq. (rad/s)
 -4.61e-001 + 3.49e+000i    1.31e-001      3.52e+000
 -4.61e-001 – 3.49e+000i    1.31e-001      3.52e+000
 -1.02e+000                 1.00e+000      1.02e+000
```

```
Beta to Rudder Numerator
    Zeros        Damping     Freq. (rad/s)
  7.52e-004    -1.00e+000      7.52e-004
 -2.54e+000     1.00e+000      2.54e+000
 -9.28e+001     1.00e+000      9.28e+001


P to Rudder Numerator
    Zeros        Damping     Freq. (rad/s)
  2.29e+000    -1.00e+000      2.29e+000
  4.27e-003    -1.00e+000      4.27e-003
 -2.44e+000     1.00e+000      2.44e+000


R to Rudder Numerator
          Zeros               Damping     Freq. (rad/s)
 -1.63e-001 + 4.94e-001i     3.13e-001      5.20e-001
 -1.63e-001 – 4.94e-001i     3.13e-001      5.20e-001
 -2.56e+000                  1.00e+000      2.56e+000


Ap = -3.2857  -11.0958  -23.5300   -0.2482
      1.0000         0         0         0
           0    1.0000         0         0
           0         0    1.0000         0

Bp = 1
     0
     0
     0


Cp = 0.0002   -0.0113   -0.0068   -0.0145


Dp = 0
```

Appendix B

F-16 Linearization Results

B.1 Description

This appendix presents the results from the linearization of the Stevens and Lewis F-16 simulation at the given trim conditions. The longitudinal results are located in Section B.2.2 and the lateral-directional results are in Section B.2.3. In both sections, the first part contains the original state-space system that relates the aircraft states (outputs and state variables) to the control inputs. Next, the poles and zeros for each transfer function are listed and the new internal state-space system is presented. Lastly, the transfer functions themselves are presented in numeric form. The numerators are factored symbolically with the stability parameters and then the output equation from the internal state-space system is given. This appendix provides the supplemental information for Chapter 2 Section 4.

B.2 Trim Conditions

Flight Condition: Straight and Level flight $(\beta, \phi = 0.0°)$

$V_T = 500.0\, ft/\sec$

$alt = 10000.0\, ft$

$XCG = 0.25\bar{c}$

Resulting Angles and Control Deflections:

$\alpha = 3.788625°$ $\quad$ $\delta_T = 0.1962366$

$\theta = 3.788625°$ $\quad$ $\delta_e = -3.851317°$

$\psi = 0.0°$ $\quad$ $\delta_a = 0.0°$

$\delta_r = 0.0°$

B.3 Longitudinal Results

B.3.1 Stevens and Lewis Linearized Longitudinal State-Space System

$$\dot{x}_{long} = A_{long} x_{long} + B_{long} u_{long}$$
$$y_{long} = C_{long} x_{long} + D_{long} u_{long}$$

where $x_{long} = \begin{Bmatrix} V_T \\ \alpha \\ \theta \\ Q \end{Bmatrix}$, $y_{long} = \begin{Bmatrix} V_T \\ \alpha \\ Q \end{Bmatrix}$, and $u_{long} = \begin{Bmatrix} \delta_T \\ \delta_e \end{Bmatrix}$

$$A_{long} = \begin{bmatrix} -0.0171623 & -3.88575 & -32.1696 & -1.10962 \\ -0.00255092 & -0.750614 & 0.0 & 0.927837 \\ 0.0 & 0.0 & 0.0 & 1.0 \\ -0.742464 \times 10^{-12} & -4.27826 & 0.0 & -1.26121 \end{bmatrix}$$

$$B_{long} = \begin{bmatrix} 19.7222 & 0.0984902 \\ -0.00261203 & -0.00158954 \\ 0.0 & 0.0 \\ 0.0 & -0.138577 \end{bmatrix}$$

$$C_{long} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \quad D_{long} = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix}$$

B.3.2 Longitudinal Poles

Short Period: $\omega_{sp} = 2.22$

$\zeta_{sp} = 0.454$

Phugoid Mode: $\omega_{p} = 0.0845$

$\zeta_{p} = 0.0842$

Longitudinal Transfer Function Denominator:

$$DENOM_{long} = -\left(s^4 + C_{3long}s^3 + C_{2long}s^2 + C_{1long}s^1 + C_{0long}\right)$$

B.3.3 Longitudinal Zeros

Velocity to Throttle: $T_{U\delta_T} = -270.27027$

$\omega_{U\delta_T} = 2.22$

$\zeta_{U\delta_T} = 0.454$

Alpha to Throttle: $T_{\alpha\delta_{T1}} = 0.51546$

$T_{\alpha\delta_{T2}} = 0.79365$

$T_{\alpha\delta_{T3}} = -1.026\times10^{11}$

Q to Throttle: $T_{Q\delta_{T1}} = 0.51546$

$T_{Q\delta_{T2}} = -2.882\times10^{17}$

Velocity to Elevator: $T_{U\delta_e} = 1.72117$

$\omega_{U\delta_e} = 7.39$

$\zeta_{U\delta_e} = 0.207$

Alpha to Elevator: $T_{\alpha\delta_e} = 0.012165$

$\omega_{\alpha\delta_e} = 0.0933$

$\zeta_{\alpha\delta_e} = 0.0943$

Q to Elevator: $T_{Q\delta_{e1}} = 1.42045$

$T_{Q\delta_{e2}} = 68.49315$

$T_{Q\delta_{e3}} = -5.587\times10^{15}$

B.3.4 Internal Longitudinal States

$$\left\{\dot{x}_{\delta T,e}(i)\right\} = \begin{bmatrix} C_{3long} & C_{2long} & C_{1long} & C_{0long} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left\{x_{\delta T,e}(i)\right\} + \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \delta_{T,e} \qquad \text{for } i \in \{1,2,3,4\}$$

where

$$\begin{aligned} C_{3long} &= -2\left(\zeta_{sp}\omega_{sp} + \zeta_p\omega_p\right) \\ C_{2long} &= -\left(\omega_p^2 + 4\zeta_{sp}\omega_{sp}\zeta_p\omega_{sp} + \omega_{sp}^2\right) \\ C_{1long} &= -2\left(\zeta_{sp}\omega_{sp}\omega_p^2 + \zeta_p\omega_p\omega_{sp}^2\right) \\ C_{0long} &= -\omega_{sp}^2\omega_p^2 \end{aligned}$$

B.3.5 Total Velocity Numeric and Symbolic Representation

$$\frac{V_T(s)}{\delta_T(s)} = \frac{19.72s^3 + 39.69s^2 + 96.96s - 0.3595}{s^4 + 2.029s^3 + 4.95s^2 + 0.08433s + 0.03511}$$

$$K_{U\delta_T} = 19.72$$
$$T_{U\delta_T} = -270.27027$$
$$\omega_{U\delta_T} = 2.22$$
$$\zeta_{U\delta_T} = 0.454$$

$$\frac{V_T(s)}{\delta_e(s)} = \frac{0.09849s^3 + 0.3581s^2 + 5.557s + 3.127}{s^4 + 2.029s^3 + 4.95s^2 + 0.08433s + 0.03511}$$

$$K_{U\delta_e} = 0.09849$$
$$T_{U\delta_e} = 1.72117$$
$$\omega_{U\delta_e} = 7.39$$
$$\zeta_{U\delta_e} = 0.207$$

$$V_T(s) = K_{U\delta_T}\left(s + \frac{1}{T_{U\delta_T}}\right)\left(s^2 + 2\zeta_{U\delta_T}\omega_{U\delta_T}s + \omega_{U\delta_T}^2\right) + K_{U\delta_e}\left(s + \frac{1}{T_{U\delta_e}}\right)\left(s^2 + 2\zeta_{U\delta_e}\omega_{U\delta_e}s + \omega_{U\delta_e}^2\right)$$

$$V_T(s) = K_{U\delta_T}\left[s^3 + \left(2\zeta_{U\delta_T}\omega_{U\delta_T} + \frac{1}{T_{U\delta_T}}\right)s^2 + \left(\frac{2\zeta_{U\delta_T}\omega_{U\delta_T}}{T_{U\delta_T}} + \omega_{U\delta_T}^2\right)s + \frac{\omega_{U\delta_T}^2}{T_{U\delta_T}}\right]$$
$$+ K_{U\delta_e}\left[s^3 + \left(2\zeta_{U\delta_e}\omega_{U\delta_e} + \frac{1}{T_{U\delta_e}}\right)s^2 + \left(\frac{2\zeta_{U\delta_e}\omega_{U\delta_e}}{T_{U\delta_e}} + \omega_{U\delta_e}^2\right)s + \frac{\omega_{U\delta_e}^2}{T_{U\delta_e}}\right]$$

$$V_T(\bar{x}) = K_{U\delta_T}\left[x(1) + \left(2\zeta_{U\delta_T}\omega_{U\delta_T} + \frac{1}{T_{U\delta_T}}\right)x(2) + \left(\frac{2\zeta_{U\delta_T}\omega_{U\delta_T}}{T_{U\delta_T}} + \omega_{U\delta_T}^2\right)x(3) + \frac{\omega_{U\delta_T}^2}{T_{U\delta_T}}x(4)\right]$$
$$+ K_{U\delta_e}\left[x(5) + \left(2\zeta_{U\delta_e}\omega_{U\delta_e} + \frac{1}{T_{U\delta_e}}\right)x(6) + \left(\frac{2\zeta_{U\delta_e}\omega_{U\delta_e}}{T_{U\delta_e}} + \omega_{U\delta_e}^2\right)x(7) + \frac{\omega_{U\delta_e}^2}{T_{U\delta_e}}x(8)\right]$$

B.3.6 Angle of Attack Numeric and Symbolic Representation

$$\frac{\alpha(s)}{\delta_T(s)} = \frac{-0.002612s^3 - 0.00837s^2 - 0.006402s + 6.239\times10^{-14}}{s^4 + 2.029s^3 + 4.95s^2 + 0.08433s + 0.03511}$$

$$\begin{aligned}
K_{\alpha\delta_T} &= -0.002612 \\
T_{\alpha\delta_{T1}} &= 0.51546 \\
T_{\alpha\delta_{T2}} &= 0.79365 \\
T_{\alpha\delta_{T3}} &= -1.02564\times10^{11} \qquad \text{Ignored because } T_{\alpha\delta_{T3}}^{-1} \approx 0
\end{aligned}$$

$$\frac{\alpha(s)}{\delta_e(s)} = \frac{-0.00159s^3 - 0.1306s^2 - 0.002312s - 0.001137}{s^4 + 2.029s^3 + 4.95s^2 + 0.08433s + 0.03511}$$

$$\begin{aligned}
K_{\alpha\delta_e} &= -0.00159 \\
T_{\alpha\delta_e} &= 0.012165 \\
\omega_{\alpha\delta_e} &= 0.0933 \\
\zeta_{\alpha\delta_e} &= 0.0943
\end{aligned}$$

$$\alpha(s) = K_{\alpha\delta_T} s\left(s + \frac{1}{T_{\alpha\delta_{T1}}}\right)\left(s + \frac{1}{T_{\alpha\delta_{T2}}}\right) + K_{\alpha\delta_e}\left(s + \frac{1}{T_{\alpha\delta_e}}\right)\left(s^2 + 2\zeta_{U\delta_e}\omega_{U\delta_e}s + \omega_{U\delta_e}^2\right)$$

$$\begin{aligned}
\alpha(s) = {} & K_{\alpha\delta_T}\left[s^3 + \left(\frac{1}{T_{\alpha\delta_{T1}}} + \frac{1}{T_{\alpha\delta_{T2}}}\right)s^2 + \frac{1}{T_{\alpha\delta_{T1}}T_{\alpha\delta_{T2}}}s\right] \\
& + K_{\alpha\delta_e}\left[s^3 + \left(2\zeta_{\alpha\delta_e}\omega_{\alpha\delta_e} + \frac{1}{T_{\alpha\delta_e}}\right)s^2 + \left(\frac{2\zeta_{\alpha\delta_e}\omega_{\alpha\delta_e}}{T_{\alpha\delta_e}} + \omega_{\alpha\delta_e}^2\right)s + \frac{\omega_{\alpha\delta_e}^2}{T_{\alpha\delta_e}}\right]
\end{aligned}$$

$$\begin{aligned}
\alpha(\bar{x}) = {} & K_{\alpha\delta_T}\left[x(1) + \left(\frac{1}{T_{\alpha\delta_{T1}}} + \frac{1}{T_{\alpha\delta_{T2}}}\right)x(2) + \frac{1}{T_{\alpha\delta_{T1}}T_{\alpha\delta_{T2}}}x(3)\right] \\
& + K_{\alpha\delta_e}\left[x(5) + \left(2\zeta_{\alpha\delta_e}\omega_{\alpha\delta_e} + \frac{1}{T_{\alpha\delta_e}}\right)x(6) + \left(\frac{2\zeta_{\alpha\delta_e}\omega_{\alpha\delta_e}}{T_{\alpha\delta_e}} + \omega_{\alpha\delta_e}^2\right)x(7) + \frac{\omega_{\alpha\delta_e}^2}{T_{\alpha\delta_e}}x(8)\right]
\end{aligned}$$

B.3.7 Pitch-Rate Numeric and Symbolic Representation

$$\frac{Q(s)}{\delta_T(s)} = \frac{0.01117s^2 + 0.02172s - 7.534\times10^{-20}}{s^4 + 2.029s^3 + 4.95s^2 + 0.08433s + 0.03511}$$

$$K_{Q\delta_T} = 0.01117$$
$$T_{Q\delta_T} = 0.51546$$
$$T_{Q\delta_{T2}} = -2.8818\times10^{17} \qquad \text{Ignored because } T_{Q\delta_{T2}}^{-1} \approx 0$$

$$\frac{Q(s)}{\delta_e(s)} = \frac{-0.1386s^3 - 0.0996s^2 - 0.001424s + 2.552\times10^{-19}}{s^4 + 2.029s^3 + 4.95s^2 + 0.08433s + 0.03511}$$

$$K_{Q\delta_e} = -0.1386$$
$$T_{Q\delta_{e1}} = 1.42045$$
$$T_{Q\delta_{e2}} = 68.49315$$
$$T_{Q\delta_{e3}} = -5.58659\times10^{15} \qquad \text{Ignored because } T_{Q\delta_{e3}}^{-1} \approx 0$$

$$Q(s) = K_{Q\delta_T} s\left(s + \frac{1}{T_{Q\delta_T}}\right) + K_{Q\delta_e} s\left(s + \frac{1}{T_{Q\delta_{e1}}}\right)\left(s + \frac{1}{T_{Q\delta_{e2}}}\right)$$

$$Q(s) = K_{Q\delta_T}\left(s^2 + \frac{1}{T_{Q\delta_T}}s\right) + K_{Q\delta_e}\left[s^3 + \left(\frac{1}{T_{Q\delta_{e1}}} + \frac{1}{T_{Q\delta_{e2}}}\right)s^2 + \frac{1}{T_{Q\delta_{e1}}T_{Q\delta_{e2}}}s\right]$$

$$Q(\bar{x}) = K_{Q\delta_T}\left(x(2) + \frac{1}{T_{Q\delta_T}}x(3)\right) + K_{Q\delta_e}\left[x(5) + \left(\frac{1}{T_{Q\delta_{e1}}} + \frac{1}{T_{Q\delta_{e2}}}\right)x(6) + \frac{1}{T_{Q\delta_{e1}}T_{Q\delta_{e2}}}x(7)\right]$$

B.4 Lateral-Directional Results

B.4.1 Stevens and Lewis Linearized Lateral-Directional State-Space System

$$\dot{x}_{lat-dir} = A_{lat-dir}x_{lat-dir} + B_{lat-dir}u_{lat-dir}$$
$$y_{lat-dir} = C_{lat-dir}x_{lat-dir} + D_{lat-dir}u_{lat-dir}$$

where $x_{lat-dir} = \begin{Bmatrix} \beta \\ \phi \\ P \\ R \end{Bmatrix}$, $y_{lat-dir} = \begin{Bmatrix} \beta \\ P \\ R \end{Bmatrix}$, and

$$u_{lat-dir} = \begin{Bmatrix} \delta_a \\ \delta_r \end{Bmatrix}$$

$$A_{lat-dir} = \begin{bmatrix} -0.237179 & 0.0641986 & 0.0663105 & -0.991989 \\ 0.0 & 0.0 & 1.0 & 0.0662205 \\ -25.5333 & 0.0 & -2.66339 & 0.590571 \\ 7.69230 & 0.0 & -0.0394986 & -0.385094 \end{bmatrix}$$

$$B_{lat-dir} = \begin{bmatrix} 0.000217329 & 0.000593344 \\ 0.0 & 0.0 \\ -0.541536 & 0.0938229 \\ -0.0241491 & -0.0489280 \end{bmatrix}$$

$$C_{lat-dir} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \qquad D_{lat-dir} = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix}$$

B.4.2 Lateral-Directional Poles

Dutch Roll: $\omega_{dr} = 3.03$
$\zeta_{dr} = 0.12$

Roll Mode: $T_r = 0.3937$

Spiral Mode: $T_s = 94.33962$

Lateral-Directional Transfer Function Denominator:

$$DENOM_{lat-dir} - \left(s^4 + C_{3lat-dir}s^3 + C_{2lat-dir}s^2 + C_{1lat-dir}s^1 + C_{0lat-dir}\right)$$

B.4.3 Lateral-Directional Zeros

Beta to Aileron:	$T_{\beta\delta_a} = -0.019011$	Beta to Rudder:	$T_{\beta\delta_{r1}} = 0.010776$
	$\omega_{\beta\delta_a} = 1.13$		$T_{\beta\delta_{r2}} = 0.3937$
	$\zeta_{\beta\delta_a} = 0.276$		$T_{\beta\delta_{r3}} = -1329.78723$
P to Aileron:	$T_{P\delta_a} = -235.84906$	P to Rudder:	$T_{P\delta_{r1}} = 0.40984$
	$\omega_{P\delta_a} = 2.98$		$T_{P\delta_{r2}} = -0.43668$
	$\zeta_{P\delta_a} = 0.111$		$T_{P\delta_{r3}} = -234.19204$
R to Aileron:	$T_{R\delta_a} = 0.98039$	R to Rudder:	$T_{R\delta_r} = 0.39063$
	$\omega_{R\delta_a} = 3.52$		$\omega_{R\delta_r} = 0.520$
	$\zeta_{R\delta_a} = 0.131$		$\zeta_{R\delta_r} = 0.313$

B.4.4 Internal Lateral-Directional States

$$\{\dot{x}_{a,r}(i)\} = \begin{bmatrix} C_{3lat-dir} & C_{2lat-dir} & C_{1lat-dir} & C_{0lat-dir} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \{x_{a,r}(i)\} + \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \delta_{a,r} \quad \text{for } i \in \{1,2,3,4\}$$

where

$$C_{3lat-dir} = -\left(2\zeta_{dr}\omega_{dr} + \frac{1}{T_r} + \frac{1}{T_s} \right) = -3.286$$

$$C_{2lat-dir} = -\left(\omega_{dr}^2 + 2\zeta_{dr}\omega_{dr}\left(\frac{1}{T_r} + \frac{1}{T_s} \right) + \frac{1}{T_r T_s} \right) = -11.1$$

$$C_{1lat-dir} = -\left(\omega_{dr}^2\left(\frac{1}{T_r} + \frac{1}{T_s} \right) + \frac{2\zeta_{dr}\omega_{dr}}{T_r T_s} \right) = -23.53$$

$$C_{0lat-dir} = -\left(\frac{\omega_{dr}^2}{T_r T_s} \right) = -0.2482$$

B.4.5 Sideslip Angle Numeric and Symbolic Representation

$$\frac{\beta(s)}{\delta_a(s)} = \frac{0.0002173s^3 - 0.01129s^2 - 0.00683s - 0.01449}{s^4 + 3.286s^3 + 11.1s^2 + 23.53s + 0.2482}$$

$$K_{\beta\delta_a} = 0.0002173$$
$$T_{\beta\delta_a} = -0.019011$$
$$\omega_{\beta\delta_a} = 1.13$$
$$\zeta_{\beta\delta_a} = 0.276$$

$$\frac{\beta(s)}{\delta_r(s)} = \frac{0.0005933s^3 + 0.05657s^2 + 0.1399s - 0.0001053}{s^4 + 3.286s^3 + 11.1s^2 + 23.53s + 0.2482}$$

$$K_{\beta\delta_r} = 0.0005933$$
$$T_{\beta\delta_{r1}} = 0.010776$$
$$T_{\beta\delta_{r2}} = 0.3937$$
$$T_{\beta\delta_{r3}} = -1329.78723$$

$$\beta(s) = K_{\beta\delta_a}\left(s + \frac{1}{T_{\beta\delta_a}}\right)\left(s^2 + 2\zeta_{\beta\delta_a}\omega_{\beta\delta_a}s + \omega_{\beta\delta_a}^2\right) + K_{\beta\delta_r}\left(s + \frac{1}{T_{\beta\delta_{r1}}}\right)\left(s + \frac{1}{T_{\beta\delta_{r2}}}\right)\left(s + \frac{1}{T_{\beta\delta_{r3}}}\right)$$

$$\beta(s) = K_{\beta\delta_a}\left[s^3 + \left(2\zeta_{\beta\delta_a}\omega_{\beta\delta_a} + \frac{1}{T_{\beta\delta_a}}\right)s^2 + \left(\frac{2\zeta_{\beta\delta_a}\omega_{\beta\delta_a}}{T_{\beta\delta_a}} + \omega_{\beta\delta_a}^2\right)s + \frac{\omega_{\beta\delta_a}^2}{T_{\beta\delta_a}}\right]$$

$$+ K_{\beta\delta_r}\left[s^3 + \left(\frac{1}{T_{\beta\delta_{r1}}} + \frac{1}{T_{\beta\delta_{r2}}} + \frac{1}{T_{\beta\delta_{r3}}}\right)s^2 + \left(\frac{1}{T_{\beta\delta_{r1}}T_{\beta\delta_{r2}}} + \frac{1}{T_{\beta\delta_{r1}}T_{\beta\delta_{r3}}} + \frac{1}{T_{\beta\delta_{r2}}T_{\beta\delta_{r3}}}\right)s + \frac{1}{T_{\beta\delta_{r1}}T_{\beta\delta_{r2}}T_{\beta\delta_{r3}}}\right]$$

$$\beta(\bar{x}) = K_{\beta\delta_a}\left[x(9) + \left(2\zeta_{\beta\delta_a}\omega_{\beta\delta_a} + \frac{1}{T_{\beta\delta_a}}\right)x(10) + \left(\frac{2\zeta_{\beta\delta_a}\omega_{\beta\delta_a}}{T_{\beta\delta_a}} + \omega_{\beta\delta_a}^2\right)x(11) + \frac{\omega_{\beta\delta_a}^2}{T_{\beta\delta_a}}x(12)\right]$$
$$+ K_{\beta\delta_r}\left[x(13) + \left(\frac{1}{T_{\beta\delta_{r1}}} + \frac{1}{T_{\beta\delta_{r2}}} + \frac{1}{T_{\beta\delta_{r3}}}\right)x(14)\right]$$
$$+ K_{\beta\delta_r}\left[\left(\frac{1}{T_{\beta\delta_{r1}}T_{\beta\delta_{r2}}} + \frac{1}{T_{\beta\delta_{r1}}T_{\beta\delta_{r3}}} + \frac{1}{T_{\beta\delta_{r2}}T_{\beta\delta_{r3}}}\right)x(15) + \frac{1}{T_{\beta\delta_{r1}}T_{\beta\delta_{r2}}T_{\beta\delta_{r3}}}x(16)\right]$$

B.4.6 Roll-Rate Numeric and Symbolic Representation

$$\frac{P(s)}{\delta_a(s)} = \frac{-0.5415s^3 - 0.3568s^2 - 4.798s + 0.02033}{s^4 + 3.286s^3 + 11.1s^2 + 23.53s + 0.2482}$$

$$K_{P\delta_a} = -0.5415$$
$$T_{P\delta_a} = \text{-}235.84906$$
$$\omega_{P\delta_a} = 2.98$$
$$\zeta_{P\delta_a} = 0.111$$

$$\frac{P(s)}{\delta_r(s)} = \frac{0.09382s^3 + 0.01434s^2 - 0.5248s + 0.002243}{s^4 + 3.286s^3 + 11.1s^2 + 23.53s + 0.2482}$$

$$K_{P\delta_r} = 0.09382$$
$$T_{P\delta_{r1}} = 0.40984$$
$$T_{P\delta_{r2}} = -0.43668$$
$$T_{P\delta_{r3}} = -234.19204$$

$$P(s) = K_{P\delta_a}\left(s + \frac{1}{T_{P\delta_a}}\right)\left(s^2 + 2\zeta_{P\delta_a}\omega_{P\delta_a}s + \omega_{P\delta_a}^2\right) + K_{P\delta_r}\left(s + \frac{1}{T_{P\delta_{r1}}}\right)\left(s + \frac{1}{T_{P\delta_{r2}}}\right)\left(s + \frac{1}{T_{P\delta_{r3}}}\right)$$

$$P(s) = K_{P\delta_a}\left[s^3 + \left(2\zeta_{P\delta_a}\omega_{P\delta_a} + \frac{1}{T_{P\delta_a}}\right)s^2 + \left(\frac{2\zeta_{P\delta_a}\omega_{P\delta_a}}{T_{P\delta_a}} + \omega_{P\delta_a}^2\right)s + \frac{\omega_{P\delta_a}^2}{T_{P\delta_a}}\right]$$
$$+ K_{P\delta_r}\left[s^3 + \left(\frac{1}{T_{P\delta_{r1}}} + \frac{1}{T_{P\delta_{r2}}} + \frac{1}{T_{P\delta_{r3}}}\right)s^2 + \left(\frac{1}{T_{P\delta_{r1}}T_{P\delta_{r2}}} + \frac{1}{T_{P\delta_{r1}}T_{P\delta_{r3}}} + \frac{1}{T_{P\delta_{r2}}T_{P\delta_{r3}}}\right)s + \frac{1}{T_{P\delta_{r1}}T_{P\delta_{r2}}T_{P\delta_{r3}}}\right]$$

$$P(\bar{x}) = K_{P\delta_a}\left[x(9) + \left(2\zeta_{P\delta_a}\omega_{P\delta_a} + \frac{1}{T_{P\delta_a}}\right)x(10) + \left(\frac{2\zeta_{P\delta_a}\omega_{P\delta_a}}{T_{P\delta_a}} + \omega_{P\delta_a}^2\right)x(11) + \frac{\omega_{P\delta_a}^2}{T_{P\delta_a}}x(12)\right]$$
$$+ K_{P\delta_r}\left[x(13) + \left(\frac{1}{T_{P\delta_{r1}}} + \frac{1}{T_{P\delta_{r2}}} + \frac{1}{T_{P\delta_{r3}}}\right)x(14)\right]$$
$$+ K_{P\delta r}\left[\left(\frac{1}{T_{P\delta_{r1}}T_{P\delta_{r2}}} + \frac{1}{T_{P\delta_{r1}}T_{P\delta_{r3}}} + \frac{1}{T_{P\delta_{r2}}T_{P\delta_{r3}}}\right)x(15) + \frac{1}{T_{P\delta_{r1}}T_{P\delta_{r2}}T_{P\delta_{r3}}}x(16)\right]$$

B.4.7 Yaw-Rate Numeric and Symbolic Representation

$$\frac{R(s)}{\delta_a(s)} = \frac{-0.02415s^3 - 0.04698s^2 - 0.3226s - 0.307}{s^4 + 3.286s^3 + 11.1s^2 + 23.53s + 0.2482}$$

$$K_{R\delta_a} = -0.02415$$
$$T_{R\delta_a} = 0.98039$$
$$\omega_{R\delta_a} = 3.52$$
$$\zeta_{R\delta_a} = 0.131$$

$$\frac{R(s)}{\delta_r(s)} = \frac{-0.04893s^3 - 0.1411s^2 - 0.05402s - 0.03387}{s^4 + 3.286s^3 + 11.1s^2 + 23.53s + 0.2482}$$

$$K_{R\delta_r} = -0.04893$$
$$T_{R\delta_r} = 0.39063$$
$$\omega_{R\delta_r} = 0.520$$
$$\zeta_{R\delta_r} = 0.313$$

$$R(s) = K_{R\delta_a}\left(s + \frac{1}{T_{R\delta_a}}\right)\left(s^2 + 2\zeta_{R\delta_a}\omega_{R\delta_a}s + \omega_{R\delta_a}^2\right) + K_{R\delta_r}\left(s + \frac{1}{T_{R\delta_r}}\right)\left(s^2 + 2\zeta_{R\delta_r}\omega_{R\delta_r}s + \omega_{R\delta_r}^2\right)$$

$$R(s) = K_{R\delta_a}\left[s^3 + \left(2\zeta_{R\delta_a}\omega_{R\delta_a} + \frac{1}{T_{R\delta_a}}\right)s^2 + \left(\frac{2\zeta_{R\delta_a}\omega_{R\delta_a}}{T_{R\delta_a}} + \omega_{R\delta_a}^2\right)s + \frac{\omega_{R\delta_a}^2}{T_{R\delta_a}}\right]$$
$$+ K_{R\delta_r}\left[s^3 + \left(2\zeta_{R\delta_r}\omega_{R\delta_r} + \frac{1}{T_{R\delta_r}}\right)s^2 + \left(\frac{2\zeta_{R\delta_r}\omega_{R\delta_r}}{T_{R\delta_r}} + \omega_{R\delta_r}^2\right)s + \frac{\omega_{R\delta_r}^2}{T_{R\delta_r}}\right]$$

$$R(\bar{x}) = K_{R\delta_a}\left[x(9) + \left(2\zeta_{R\delta_a}\omega_{R\delta_a} + \frac{1}{T_{R\delta_a}}\right)x(10) + \left(\frac{2\zeta_{R\delta_a}\omega_{R\delta_a}}{T_{R\delta_a}} + \omega_{R\delta_a}^2\right)x(11) + \frac{\omega_{R\delta_a}^2}{T_{R\delta_a}}x(12)\right]$$
$$+ K_{R\delta_r}\left[x(13) + \left(2\zeta_{R\delta_r}\omega_{R\delta_r} + \frac{1}{T_{R\delta_r}}\right)x(14) + \left(\frac{2\zeta_{R\delta_r}\omega_{R\delta_r}}{T_{R\delta_r}} + \omega_{R\delta_r}^2\right)x(15) + \frac{\omega_{R\delta_r}^2}{T_{R\delta_r}}x(16)\right]$$

Appendix C

Simulation Code

C.1 Stevens & Lewis F-16

C.1.1 Description

Subs.f is the main program for the Stevens and Lewis non-linear F-16 simulation. Subroutine F is the actual simulation portion of the code since it called at each time step by either the desktop's driving program or *CASTLE*'s control subroutine. F calculates the 12 aircraft state time derivatives $(\dot{x})$ for a given set of controls: thtl, el, ail, and rdr (throttle, elevator, aileron, and rudder deflections respectively). It also generates the body-axis forces (FAX, FAY, FAZ) and moments (TAL, TAN, and TAM) by doing an aerodynamic coefficient build-up where the coefficients are functions of the aircraft states, similar to that found in Etkin. [11] The subroutine RK4 is used to do the actual integration by performing a fourth-order Runge-Kutta integration of the stated derivatives from F. RK4 integrates by calling F a series of times at different time steps (smaller than the time step given to it) to generate a polynomial fit for the desired time step (DT) used by the driving program or *CASTLE*. The LINEARIZE subroutine sets up the state variables, control variables, and output variables and calls the subroutine JACOB to linearization the system as was shown in Section 2.2. TRIMMER performs the control surface trimming for

the F-16 at a given reference flight condition by using the constraints in the CONSTR and the cost function in SF16. Much of the code has been left out for conciseness, but descriptions of all the deleted subroutines are found in Table A.1:

Table C.1 – Stevens and Lewis Subroutine Names, Arguments and Descriptions [2]

Subroutine Name and Arguments	**Description**
`ADC (VT, ALT, AMACH, QBAR)`	Given velocity and altitude, calculates Mach number and dynamic pressure
`TGEAR (THTL)`	Calculates the commanded power for a given throttle setting
`THRUST (CPOW, ALT, AMACH)`	Calculates the thrust at given power setting, altitude, and Mach number
`CX (ALPHA, EL)`	Calculates the Body-axis x-directional force coefficient based on angle of attack and elevator deflection
`CY (BETA, AIL, RDR)`	Calculates the Body-axis y-directional force coefficient based on sideslip angle, aileron, and rudder deflections
`CZ (ALPHA, BETA, EL)`	Calculates the Body-axis z-directional force coefficient based on angle of attack and elevator deflection
`CL (ALPHA, BETA)`	Calculates the Body-axis x-directional moment coefficient based on angle of attack and sideslip angle
`DLDA (ALPHA, BETA)`	Calculates the change in rolling moment due to aileron as a function of angle of attack and sideslip angle
`DLDR (ALPHA, BETA)`	Calculates the change in rolling moment due to rudder as a function of angle of attack and sideslip angle
`CM (ALPHA, EL)`	Calculates the Body-axis y-directional moment coefficient based on angle of attack and elevator deflection
`CN (ALPHA, BETA)`	Calculates the Body-axis z-directional moment coefficient based on angle of attack and sideslip angle
`DNDA (ALPHA, BETA)`	Calculates the change in yawing moment due to aileron as a function of angle of attack and sideslip angle
`DNDR (ALPHA, BETA)`	Calculates the change in yawing moment due to rudder as a function of angle of attack and sideslip angle
`DAMP (ALPHA, D)`	Calculates the non-dimensionalized damping coefficients $D = \{C_{XQ}, C_{YR}, C_{YP}, C_{ZQ}, C_{LR}, C_{LP}, C_{MQ}, C_{NR}, C_{NP}\}$
`OUTMAT`	Outputs the linearized matrices to a given file

C.1.2 Code: Subs.f [6]

```
      SUBROUTINE F (TIME,X,XD)
      REAL X(*), XD(*), D(9)
      COMMON/PARAM/XCG
      COMMON/CONTROLS/THTL,EL,AIL,RDR
      COMMON/OUTPUT/AN,ALAT,AX,QBAR,AMACH,Q,ALPHA,THETA,VT,BETA,P,R
      COMMON/FORCES/FAX,FAY,FAZ,FEX,TAL,TAM,TAN
      DATA S,B,CBAR,RM,XCGR,HE/300,30,11.32,1.57E-3,0.35,160.0/
      DATA C1,C2,C3,C4,C5,C6,C7,C8,C9/ -.770, .02755, 1.055E-4,
     & 1.642E-6, .9604, 1.759E-2, 1.792E-5, -.7336, 1.587E-5/
      DATA RTOD,G / 57.29578, 32.17/
C
C Assign state & control variables
C
      VT    = X(1)
      ALPHA = X(2)*RTOD
      BETA  = X(3)*RTOD
      PHI   = X(4)
      THETA = X(5)
      PSI   = X(6)
      P     = X(7)
      Q     = X(8)
      R     = X(9)
      ALT   = X(12)
C
C Air Data Computer and engine model
C
      CALL    ADC(VT,ALT,AMACH,QBAR)
      CPOW  = TGEAR(THTL)
      T     = THRUST(CPOW,ALT,AMACH)
C
C Look-up tables and component buildup
C
      CXT = CX (ALPHA, EL)
      CYT = CY (BETA,AIL,RDR)
      CZT = CZ (ALPHA,BETA,EL)
      DAIL= AIL/20.0
      DRDR= RDR/30.0
      CLT = CL(ALPHA,BETA) + DLDA(ALPHA,BETA)*DAIL
     &                     + DLDR(ALPHA,BETA)*DRDR
      CMT = CM(ALPHA,EL)
      CNT = CN(ALPHA,BETA) + DNDA(ALPHA,BETA)*DAIL
     &                     + DNDR(ALPHA,BETA)*DRDR
C
C Add damping derivatives :
      TVT= 0.5/VT
      B2V= B*TVT
      CQ = CBAR*Q*TVT
      CALL DAMP(ALPHA,D)
      CXT=  CXT + CQ  *   D(1)
      CYT=  CYT + B2V * ( D(2)*R + D(3)*P )
      CZT=  CZT + CQ  *   D(4)
      CLT=  CLT + B2V * ( D(5)*R + D(6)*P )
      CMT=  CMT + CQ  *   D(7)             + CZT * (XCGR-XCG)
      CNT=  CNT + B2V * ( D(8)*R + D(9)*P )- CYT * (XCGR-XCG) * CBAR/B
```

```
C
C Total Forces
C
      FAX = CXT * QBAR * S
      FAY = CYT * QBAR * S
      FAZ = CZT * QBAR * S
      FEX = T
      TAL = CLT * QBAR * S * B
      TAM = CMT * QBAR * S * CBAR
      TAN = CNT * QBAR * S * B
C
C Get ready for state equations
C
      CBTA  = COS(X(3))
      U= VT * COS(X(2)) * CBTA
      V= VT * SIN(X(3))
      W= VT * SIN(X(2)) * CBTA
      STH   = SIN(THETA)
      CTH   = COS(THETA)
      SPH   = SIN(PHI)
      CPH   = COS(PHI)
      SPSI  = SIN(PSI)
      CPSI  = COS(PSI)
      QS    = QBAR * S
      QSB   = QS * B
      RMQS  = RM * QS
      GCTH  = G * CTH
      QSPH  = Q * SPH
      AY    = RMQS * CYT
      AZ    = RMQS * CZT
C
C Force Equations
C
      UDOT  = R*V – Q*W – G*STH + RM * (QS * CXT + T)
      VDOT  = P*W – R*U + GCTH * SPH + AY
      WDOT  = Q*U – P*V + GCTH * CPH + AZ
      DUM   = (U*U + W*W)
C
      XD(1) = (U*UDOT + V*VDOT + W*WDOT)/VT
      XD(2) = (U*WDOT – W*UDOT) / DUM
      XD(3) = (VT*VDOT- V*XD(1)) * CBTA / DUM
C
C Kinematics
C
      XD(4) =  P + (STH/CTH)*(QSPH + R*CPH)
      XD(5) =  Q*CPH – R*SPH
      XD(6) = (QSPH + R*CPH)/CTH
C
C Moments
C
      XD(7) =  (C2*P + C1*R + C4*HE)*Q + QSB*(C3*CLT + C4*CNT)
      XD(8) =  (C5*P – C7*HE)*R + C6*(R*R-P*P) +QS*CBAR*C7*CMT
      XD(9) =  (C8*P – C2*R + C9*HE)*Q + QSB*(C4*CLT + C9*CNT)
```

```
C
C Navigation
C
      T1= SPH * CPSI
      T2= CPH * STH
      T3= SPH * SPSI
      S1= CTH * CPSI
      S2= CTH * SPSI
      S3= T1  * STH – CPH * SPSI
      S4= T3  * STH + CPH * CPSI
      S5= SPH * CTH
      S6= T2*CPSI + T3
      S7= T2  * SPSI – T1
      S8= CPH * CTH
C
      XD(10)   =  U * S1  + V * S3  + W * S6   ! North Speed
      XD(11)   =  U * S2  + V * S4  + W * S7   ! East Speed
      XD(12)   =  U * STH – V * S5  - W * S8   ! Vertical Speed
C
C OUTPUTS
C
      AN = -AZ/G
      ALAT = AY/G
      RETURN
      END




C*********************************************************************
C   Subroutine: LINEARIZE
C   Purpose: This function linearizes the aircraft for user
C              instructed states and controls
C
*********************************************************************
      SUBROUTINE LINEARIZE
      PARAMETER (NN=21, MM=21, NOP=21)
      DOUBLE PRECISION FDX,FDU,YDX,YDU
      EXTERNAL F, SF16, FDX, FDU, YDX, YDU
      INTEGER LINCHOICE
      REAL X(NN), XD(NN), U(MM), Y(NOP)
      COMMON/ STATE/ X, XD
      COMMON/ CONTROLS/ U
      COMMON/OUTPUT/ Y
C
C LINEARIZATION VARIABLES
      INTEGER IOX(21), JOX(21), NRX, NCU
      INTEGER IOY(21), JOU(21), NRY
      REAL AMAT(400),
     &     BMAT(200),
     &     CMAT(400),
     &     DMAT(200)
C
      WRITE(*,*) 'Enter the type of Linearization'
      WRITE(*,*) '      (Long = 1    Lat-Dir = 2)'
      READ(*,*) LINCHOICE
```

```
      IF(LINCHOICE.EQ.1) THEN
          OPEN(UNIT=4,FILE='LONG.M',STATUS='NEW')
      ELSE
          OPEN(UNIT=4,FILE='LAT.M',STATUS='NEW')
      ENDIF

      WRITE(4,889)

      NCU = 2
      NRY = 3
      NRX = 4

      IF(LINCHOICE.EQ.1) THEN
C       STATES
          IOX(1) = 1                ! VT
          IOX(2) = 2                ! ALPHA
          IOX(3) = 5                ! THETA
          IOX(4) = 8                ! Q
          JOX(1) = 1
          JOX(2) = 2
          JOX(3) = 5
          JOX(4) = 8
C       CONTROLS
          JOU(1) = 1                ! DTH
          JOU(2) = 2                ! DE
C       OUTPUTS
          IOY(1) = 9                ! VT
          IOY(2) = 7                ! ALPHA
          IOY(3) = 6                ! Q

      ELSE
C       STATES
          IOX(1) = 3                ! BETA
          IOX(2) = 4                ! PHI
          IOX(3) = 7                ! P
          IOX(4) = 9                ! R
          JOX(1) = 3
          JOX(2) = 4
          JOX(3) = 7
          JOX(4) = 9
C       CONTROLS
          JOU(1) = 3                ! DA
          JOU(2) = 4                ! DR
C       OUTPUTS
          IOY(1) = 10                ! BETA
          IOY(2) = 11                ! P
          IOY(3) = 12                ! R
      ENDIF
C
      WRITE(6,*) 'Beginning A Matrix Calculations ... '
      CALL JACOB(FDX, F, X, XD, X, IOX, JOX, AMAT, NRX, NRX)
      DO 703 I=1, NRX
 C. .          WRITE(4,799) (AMAT(NRX*(J-1)+I) , J = 1, NRX )
C
      WRITE(4,5000)
      WRITE(4,890)
C
```

```
      WRITE(6,*) 'Beginning B Matrix Calculations ... '
       CALL JACOB(FDU, F, X, XD, U, IOX, JOU, BMAT, NRX, NCU)
       DO 704 I=1, NRX
 C. .         WRITE(4,799) (BMAT(NRX*(J-1)+I), J = 1, NCU)
C
      WRITE(4,5000)
      WRITE(4,4889)
C
      WRITE(6,*) 'Beginning C Matrix Calculations ... '
      CALL JACOB(YDX, F, X, XD, X, IOY, JOX, CMAT, NRY, NRX)
      DO 4703 I=1, NRY
 C. .          WRITE(4,799) (CMAT(NRY*(J-1)+I) , J = 1, NRX )
C
      WRITE(4,5000)
      WRITE(4,4890)
C
      WRITE(6,*) 'Beginning D Matrix Calculations ... '
      CALL JACOB(YDU, F, X, XD, U, IOY, JOU, DMAT, NRY, NCU)
      DO 4704 I=1, NRY
 4704          WRITE(4,799) (DMAT(NRY*(J-1)+I), J = 1, NCU)
      WRITE(4,5000)
 799  FORMAT( 15E14.6 )
 889  FORMAT('A = [',$)
 890  FORMAT('B = [',$)
 4889 FORMAT('C = [',$)
 4890 FORMAT('D = [',$)
 5000 FORMAT('];')
C
      RETURN
      END




C
C *****************************************************************
C
      FUNCTION SF16(S)
       PARAMETER (NN=20)
       REAL S(*)
       COMMON/STATE/ X(NN), XDOT(NN)
       COMMON/CONTROLS/THTL, EL, AIL, RDR
       THTL=  S(1)
       EL  =  S(2)
       X(2)=  S(3)
       AIL =  S(4)
       RDR =  S(5)
       X(3)=  S(6)
       X(13)= TGEAR(THTL)
       CALL   CONSTR(X)
       CALL   F(TIME,X,XDOT)
      SF16 =    XDOT(1)**2 + 100.0*( XDOT(2)**2 + XDOT(3)**2 )
     & + 10.0*( XDOT(7)**2 +         XDOT(8)**2 + XDOT(9)**2 )
       RETURN
       END
```

```
C
C ***************************************************************
C
      SUBROUTINE TRIMMER (NV, COST)
      PARAMETER (NN=20, MM=10)
      EXTERNAL COST
      CHARACTER*1 ANS
      REAL S(6), DS(6)
      COMMON/ STATE/ X(NN)
      COMMON/ CONTROLS/ U(MM)
      COMMON/ OUTPUT/ AN,AY,AX,QBAR,AMACH   ! common to aircraft
      DATA RTOD /57.29577951/
C
      S(1)= U(1)
      S(2)= U(2)
      S(3)= X(2)
      IF(NV .LE. 3) GO TO 10
      S(4)= U(3)
      S(5)= U(4)
      S(6)= X(3)
 C. .  DS(1) = 0.2
      DS(2) = 1.0
      DS(3) = 0.02
      IF (NV .LE. 3) GO TO 20
      DS(4) = 1.0
      DS(5) = 1.0
      DS(6) = 0.02
 C. .  NC= 5000
      SIGMA = -1.0
C
      CALL SMPLX(COST,NV,S,DS,SIGMA,NC,F0,FFIN)
      FFIN = COST(S)
        IF (NV .GT. 3) THEN
      WRITE(*,'(/11X,A)') 'Throttle    Elevator     Ailerons   Rudder'
      WRITE(*,'(9X,4(1PE14.6,3X),/)') U(1), U(2), U(3), U(4)
      WRITE(*,99) 'Angle of attack', RTOD*X(5), 'Sideslip angle',
     &            RTOD*X(3)
      WRITE(*,99) 'Pitch angle', RTOD*X(5), 'Bank Angle', RTOD*X(4)
      WRITE(*,99) 'Normal acceleration', AN,'Lateral acceleration', AY
      WRITE(*,99) 'Dynamic pressure', QBAR, 'Mach number', AMACH
      WRITE(*,99) 'Power Commanded ', TGEAR(U(1))
        ELSE
      WRITE(*,'(/1X,A)')' Throttle     Elevator     Alpha       Pitch'
      WRITE(*,'(1X,4(1PE14.6,3X))')U(1),U(2),X(2)*RTOD,X(3)*RTOD
      WRITE(*,'(/1X,A)')'Normal acceleration   Dynamic Pressure   Mach
'
      WRITE(*,'(5X,3(1PE14.6,7X))') AN,QBAR,AMACH
        END IF
      WRITE(*,99)'Initial cost function',F0,'Final cost function',FFIN
 C. .  FORMAT(2(1X,A22,1PE14.6))

      RETURN
      END
```

```
C
C ***************************************************************************
C
      FUNCTION CLF16(S)
C
      INTEGER NN
      PARAMETER (NN=20, MM=10)
      REAL S(*), X(NN), XDOT(NN), U(MM)
      COMMON/STATE/X, XDOT
      COMMON/CONTROLS/ U
      REAL TIME
C
      U(1) = S(1)
      U(2) = S(2)
      X(2) = S(3)
      U(3) = S(4)
      U(4) = S(5)
      X(3) = S(6)
      X(13)= TGEAR(THTL)
      CALL   CONSTR(X)
      CALL   F(TIME,X,XDOT)
      CLF16 = XDOT(1)**2 + 100.0*( XDOT(2)**2 + XDOT(3)**2 )
     &       + 10.0*( XDOT(7)**2 + XDOT(8)**2 + XDOT(9)**2 )
      RETURN
      END




C
C ***************************************************************************
C
      SUBROUTINE CONSTR (X)   ! used by COST, to apply constraints
      DIMENSION X(*)
      REAL RADGAM,SINGAM,RR,PR,TR,PHI,CPHI,SPHI
      LOGICAL COORD, STAB
      COMMON/CNSTRNT/RADGAM,SINGAM,RR,PR,TR,PHI,CPHI,SPHI,COORD,STAB
C  common to main program
      CALPH = COS(X(2))
      SALPH = SIN(X(2))
      CBETA = COS(X(3))
      SBETA = SIN(X(3))
      PHI = 0.0
      CPHI=COS(PHI)
      SPHI=SIN(PHI)
      IF (COORD) THEN
        WRITE(6,*) '! coordinated turn logic here '
      ELSE IF (TR .NE. 0.0) THEN
        WRITE(6,*) '! skidding turn logic here '
      ELSE   ! non-turning flight
        X(4)= PHI
        D  = X(2)
        IF(PHI .NE. 0.0) D = -X(2)     ! inverted
        IF( SINGAM .NE. 0.0 ) THEN     ! climbing
          SGOCB = SINGAM / CBETA
C     roc constraint
          X(5)  = D + ATAN( SG0CB/SQRT(1.0-SGOCB*SGOCB))
```

```
      ELSE
          X(5)  = D                     ! level
        END IF
        X(7)=RR
        X(8)=PR
        IF (STAB) THEN                    ! stab.-axis roll
          X(9)= RR*SALPH/CALPH
        ELSE
          X(9) = 0.0                      ! body-axis roll
        END IF
      END IF
      RETURN
      END


C
C *****************************************************************
C
C This simplex algorithms minimizes FX(X), where X is (Nx1).  DX
C contains the initial perturbations in X.  SD should be set according
C to the tolerance required; when SD<0 the algorithm calls FX  M times
C
      SUBROUTINE SMPLX(FX,N,X,DX,SD,M,Y0,YL)
      REAL X(*), DX(*)
      DIMENSION XX(32), XC(32), Y(33), V(32,32)
      NV=N+1
      DO 2 I=1,N
      DO 1 J=1,NV
 1    V(I,J)=X(I)
 2    V(I,I+1)=X(I)+DX(I)
      Y0=FX(X)
      Y(1)=Y0
      DO 3 J=2, NV
 3    Y(J)=FX(V(1,J))
      K=NV
 C. .  YH=Y(1)
      YL=Y(1)
      NH=1
      NL=1
      DO 5 J=2,NV
      IF(Y(J).GT.YH) THEN
        YH=Y(J)
        NH=J
      ELSEIF(Y(J).LT.YL) THEN
        YL=Y(J)
        NL=J
      ENDIF
 C. .  CONTINUE
      YB=Y(1)
      DO 6 J=2,NV
 C. .  YB=YB+Y(J)
      YB=YB/NV
      D=0.0
      DO 7 J=1,NV
 C. .  D=D+(Y(J)-YB)**2
      SDA=SQRT(D/NV)
      IF((K.GE.M).OR.(SDA.LE.SD)) THEN
          SD=SDA
```

```
        M=K
        YL=Y(NL)
        DO 8 I=1,N
C. .  X(I)=V(I,NL)
        RETURN
      END IF
      DO 10 I=1,N
      XC(I)=0.0
      DO 9 J=1,NV
9     IF(J.NE.NH) XC(I)=XC(I)+V(I,J)
10    XC(I)=XC(I)/N
      DO 11 I=1,N
C. .  X(I)=XC(I)+XC(I)-V(I,NH)
      K=K+1
      YR=FX(X)
      IF(YR.LT.YL) THEN
        DO 12 I=1,N
C. .    XX(I)=X(I)+X(I)-XC(I)
        K=K+1
        YE=FX(XX)
        IF(YE.LT.YR) THEN
          Y(NH)=YE
          DO 13 I=1,N
C. .      V(I,NH)=XX(I)
        ELSE
          Y(NH)=YR
          DO 14 I=1,N
C. .      V(I,NH)=X(I)
        END IF
        GOTO 4
      ENDIF
      Y2=Y(NL)
      DO 15 J=1,NV
C. .  IF((J.NE.NL).AND.(J.NE.NH).AND.(Y(J).GT.Y2)) Y2=Y(J)
      IF (YR.LT.YH) THEN
        Y(NH)=YR
        DO 16 I=1,N
C. .  V(I,NH)=X(I)
        IF (YR.LT.Y2) GO TO 4
      ENDIF
      DO 17 I=1,N
C. .  XX(I)=0.5*(V(I,NH)+XC(I))
      K=K+1
      YC=FX(XX)
      IF (YC.LT.YH) THEN
        Y(NH)=YC
        DO 18 I=1,N
C. .    V(I,NH)=XX(I)
      ELSE
        DO 20  J=1, NV
        DO 19  I=1,N
19      V(I,J)=0.5*(V(I,J)+V(I,NL))
20      IF (J.NE.NL) Y(J)=FX(V(1,J))
        K=K+N
      ENDIF
      GO TO 4
      END
```

```
C
C ***************************************************************
C
      SUBROUTINE JACOB (FN,F,X,XD,V,IO,JO,ABC,NR,NC)
      DIMENSION X(*),XD(*),V(*),IO(*),JO(*),ABC(*)
      EXTERNAL FN, F
      LOGICAL FLAG, DIAGS
      CHARACTER*1 ANS
      REAL*8 FN,TDV
      DATA DEL,DMIN,TOLMIN,OKTOL /.01, .5, 3.3E-5, 8.1E-4/
C
      DIAGS= .TRUE.
      WRITE (*, '(1X,A,$)'), 'Diagnostics ?  (Y/N, "/"= N) '
      READ(*,'(A)') ANS
      IF (ANS.EQ.'/' .OR. ANS.EQ.'N' .OR. ANS.EQ.'n')  DIAGS=.FALSE.
      IJ=1
      DO 40  J=1,NC
      DV=AMAX1( ABS( DEL*V(JO(J)) ), DMIN )
      DO 40 I=1,NR
      FLAG= .FALSE.
   C. .  TOL= 0.1
      OLTOL= TOL
      TDV= DBLE( DV )
      A2= 0.0
      A1= 0.0
      A0= 0.0
      B1= 0.0
      B0= 0.0
      D1= 0.0
      D0= 0.0
      IF (DIAGS .OR. FLAG) WRITE(*,'(/1X,A8,I2,A1,I2,11X,A12,8X,A5)')
     & 'Element ',I,',',j, 'Perturbation','Slope'
      DO 20 K= 1,18                ! iterations on TDV
      A2= A1
      A1= A0
      B1= B0
      D1= D0
      A0= FN(F,XD,X,IO(I),JO(J),TDV)
      B0= AMIN1( ABS(A0), ABS(A1) )
      D0= ABS(A0-A1)
       IF(DIAGS.OR.FLAG) WRITE(*,'(20X,1P2E17.6)') TDV, A0
       IF(K.LE.2) GO TO 20
       IF (A0.EQ.A1 .AND. A1.EQ.A2) THEN
            ANSR= A1
          GO TO 30
      END IF
       IF (A0.EQ.0.0) GO TO 25
 C. .  IF (D0.LE.TOL*B0 .AND. D1.LE.TOL*B1) THEN
            ANSR= A1
          OLTOL= TOL
          IF(DIAGS.OR.FLAG) WRITE(*,'(1X,A16,F8.7)')'Met tolerance=',
     &         TOL
          IF (TOL.LE.TOLMIN) THEN
               GO TO 30
          ELSE
               TOL= 0.2*TOL
               GO TO 10
          END IF
```

```
      END IF
 20   TDV= 0.6D0*TDV
 25   IF (OLTOL.LE.OKTOL) THEN
             GO TO 30
      ELSE IF (.NOT.FLAG) THEN
            WRITE(*,'(/1X,A)') 'Failed to Converge *****'
          FLAG= .TRUE.
          GO TO 1
       ELSE
 C. .       WRITE(*,'(1X,A,$)') 'Your best guess : '
            READ(*,*,ERR=21) ANSR
          FLAG= .FALSE.
          GO TO 30
       END IF
 30   ABC(IJ)= ANSR
      IF (DIAGS) WRITE (*,'(27X,A9,1PE13.6)'),'Answer = ', ANSR
 C. .  IJ= IJ+1
      RETURN
      END
```

```
C
C ***************************************************************
C
      DOUBLE PRECISION FUNCTION FDX(F,XD,X,I,J,DDX)
       REAL*4 XD(*), X(*)
       DOUBLE PRECISION T, DDX, XD1, XD2
       EXTERNAL F
       TIME= 0.0
       T   = DBLE( X(J) )
       X(J)= SNGL( T-DDX )
       CALL F(TIME,X,XD)
       XD1 = DBLE(XD(I))
       X(J)= SNGL( T+DDX )
       CALL F(TIME,X,XD)
       XD2 = DBLE( XD(I) )
       FDX = (XD2-XD1)/(DDX+DDX)
       X(J)= SNGL( T )
       RETURN
      END
```

```
C
C ***************************************************************************
C
      DOUBLE PRECISION FUNCTION FDU(F,XD,X,I,J,DDU)
       PARAMETER (NIN=10)
       REAL*4 XD(*), X(*)
       COMMON/CONTROLS/U(NIN)
       DOUBLE PRECISION T, DDU, XD1, XD2
       EXTERNAL F
       TIME= 0.0
       T   = DBLE( U(J) )
       U(J)= SNGL( T-DDU )
       CALL F(TIME,X,XD)
       XD1 = DBLE(XD(I))
       U(J)= SNGL( T+DDU )
       CALL F(TIME,X,XD)
       XD2 = DBLE( XD(I) )
       FDU = (XD2-XD1)/(DDU+DDU)
       U(J)= SNGL( T )
       RETURN
      END
C
C ***************************************************************************
C
      DOUBLE PRECISION FUNCTION YDX (F,XD,X,I,J,DDX)
       PARAMETER (NOP=20)
       REAL*4 XD(*), X(*)
       COMMON/OUTPUT/ Y(NOP)
       DOUBLE PRECISION T, DDX, YD1, YD2
       EXTERNAL F
       TIME= 0.0
       T   = DBLE( X(J) )
       X(J)= SNGL( T-DDX )
       CALL F(TIME,X,XD)
       YD1 = DBLE(Y(I))
       X(J)= SNGL( T+DDX )
       CALL F(TIME,X,XD)
       YD2 = DBLE( Y(I) )
       YDX = (YD2-YD1)/(DDX+DDX)
       X(J)= SNGL( T )
       RETURN
      END
```

```
C
C ***************************************************************************
C
      DOUBLE PRECISION FUNCTION YDU (F,XD,X,I,J,DDU)
       PARAMETER (NIN=10, NOP=20)
       REAL*4 XD(*), X(*)
       COMMON/CONTROLS/U(NIN)
       COMMON/OUTPUT/ Y(NOP)
       DOUBLE PRECISION T, DDU, YD1, YD2
       EXTERNAL F
       TIME= 0.0
       T   = DBLE( U(J) )
       U(J)= SNGL( T-DDU )
       CALL F(TIME,X,XD)
       YD1 = DBLE(Y(I))
       U(J)= SNGL( T+DDU )
       CALL F(TIME,X,XD)
       YD2 = DBLE( Y(I) )
       YDU = (YD2-YD1)/(DDU+DDU)
       U(J)= SNGL( T )
       RETURN
      END


C
C ***************************************************************************
C
      SUBROUTINE RK4 (F,TT,DT,XX,XD,NX)
C
      INTEGER NN
      PARAMETER (NN=30)
      REAL XX(*), XD(*), X(NN), XA(NN)
C
      INTEGER M, NX
      REAL TT,DT,Q,T
      EXTERNAL F
C
      CALL F(TT,XX,XD)
      DO 1 M=1,NX
      XA(M)=XD(M)*DT
 C. .  X(M)=XX(M)+0.5*XA(M)
      T=TT+0.5*DT
      CALL F(T,X,XD)
      DO 2 M=1,NX
      Q=XD(M)*DT
      X(M)=XX(M)+0.5*Q
 C. .  XA(M)=XA(M)+Q+Q
      CALL F(T,X,XD)
      DO 3 M=1,NX
      Q=XD(M)*DT
      X(M)=XX(M)+Q
 C. .  XA(M)=XA(M)+Q+Q
      TT=TT+DT
      CALL F(TT,X,XD)
      DO 4 M=1, NX
 C. .  XX(M)=XX(M)+(XA(M)+xd(M)*DT)/6.0
      RETURN
      END
```

C.2 Transfer Function Simulation Model

C.2.1 Description

TransferFunctions.f is the main program for the TFSM. It sets the values for the 52 stability parameters and the reference conditions. At each time step it calculates the aircraft states by using state-space representations and integration depending on the variable. TransferFunctions.f then calculations the output variables: body-axis forces and moments. Integrating the internal states for the state-space representations is the last step performed so it is ready for the next time step.

C.2.2 Code: TransferFunctions.f

```
C*****************************************************************
C
C     TITLE:     TransferFunctions
C
C-----------------------------------------------------------------
C
C     FUNCTION:  Subroutine to determine the aircraft dynamics
C                from the derived transfer functions.
C
C-----------------------------------------------------------------
C
C     MODULE TYPE:     GENERIC
C
C-----------------------------------------------------------------
C
C     DESIGNED BY:     Henrik Pettersson
C
C     CODED BY:        Henrik Pettersson
C
C     MAINTAINED BY:   John Leonard
C
C-----------------------------------------------------------------
C
C     INPUTS:
C
C  D2R      0.01745329 Deg To Rad Conversion Factor RAD/DEG    -------
C  DA       aileron deflection                      RADIANS     +RWD
C  DE       elevator deflection                      RAD        +TED
C  DEMOCASE AOE 4144 Long Demo Cases                 -------    -------
C  DR       rudder deflection                       RADIANS     +TEL
C  DT       Simulation time step (frame time)        SEC       -------
```

```
C  DTH      throttle (0 to +1)                          (0-1)     -------
C  G        Acceleration Due To Gravity                 FT/S2     -------
C  ICASE    Case: A=1, B=2, C=3, D=4                    None       None
C  IMODE    Sim. Mode: -2=init,-1=reset,0=hold,1=ru     -------   -------
C  KALPHADE gain for dalpha to de TF                    Hunh?      None
C  KALPHADT gain for dalpha to dth TF                   Hunh?      None
C  KBETADA  gain for dbeta to da TF                     Hunh?      None
C  KBETADR  gain for dbeta to dr TF                     Hunh?      None
C  KPBDA    gain for dpb to da TF                       Hunh?      None
C  KPBDR    gain for dpb to dr TF                       Hunh?      None
C  KQBDE    gain for dqb to de TF                       Hunh?      None
C  KQBDT    gain for dqb to dth TF                      Hunh?      None
C  KRBDA    gain for drb to da TF                       Hunh?      None
C  KRBDR    gain for drb to dr TF                       -------   -------
C  KUDE     gain for dU to de TF                        Hunh?      None
C  KUDT     gain for dU to dth TF                       Hunh?      None
C  LAMDAFX  FAX model following gain                    None      -------
C  LAMDAFY  FAY model following gain                    None      -------
C  LAMDAFZ  FAZ model following gain                    None      -------
C  LAMDATL  TAL model following gain                    None      -------
C  LAMDATM  TAM model following gain                    None      -------
C  LAMDATN  TAN model following gain                    None      -------
C  OMEGAALPHADE
C!          natural frequency from dalpha to de TF     SEC-1      None
C  OMEGABETADA
C!          Natural frequency from dbeta to DA TF      SEC-1      None
C  OMEGAD   dutch roll natural frequency                SEC-1      None
C  OMEGAP    hugoid natural frequency                   SEC-1      None
C  OMEGAPBDA
C           natural frequency from dpb to da TF        SEC-1      None
C  OMEGARBDA
C           natural frequency from drb to da TF        SEC-1      None
C  OMEGARBDR
C           natural frequency from drb to dr TF        SEC-1      None
C  OMEGASP  short period natural frequency             SEC-1      None
C  OMEGAUDE natural frequency from dU to de TF         SEC-1      None
C  OMEGAUDT natural frequency from dU to dth TF        SEC-1      None
C  PIOVR2   Constant PI / 2.0                          -------   -------
C  R2D      57.2957795 Rad To Deg Conversion Factor    DEG/RAD   -------
C  REF_ALFA Reference Angle of Attack                   RAD       -------
C  REF_ALT  Reference Altitude                          FT         +UP
C  REF_BETA Reference Sideslip Angle                    RAD       -------
C  REF_PB   Reference Roll Rate                         RAD       -------
C  REF_PHI  Reference Roll Angle                        RAD       -------
C  REF_PSI  Reference Heading Angle                     RAD       -------
C  REF_QB   Reference Pitch Rate                        RAD       -------
C  REF_RB   Reference Yaw Rate                          RAD       -------
C  REF_THET Reference Pitch Angle                       RAD       -------
C  REF_VRW  Reference Total Velocity                    FT/S      -------
C  REF_XCG  Reference XCG Location                      FT         +NORTH
C  REF_YCG  Reference YCG Location                      FT         +EAST
C  TALPHADE time constant from dalpha to de TF          SEC-1      None
C  TALPHADT1
C           1st time constant from dalpha to DTH TF    SEC-1      None
C  TALPHADT2
C           2nd time constant from dalpha to DTH TF    SEC-1      None
C  TBETADA  Time constant from dbeta to da TF           SEC-1      None
C  TBETADR1 1st time constant from dbeta to dr TF       SEC-1      None
```

```
C  TBETADR2 2nd time constant from dbeta to dr TF    SEC-1      None
C  TBETADR3 3rd time constant from dbeta to dr TF    SEC-1      None
C  TPBDA    Time constant from dpb to da TF          SEC-1      None
C  TPBDR1   1st time constant from dpb to dr TF      SEC-1      None
C  TPBDR2   2nd time constant from dpb to dr TF      SEC-1      None
C  TPBDR3   3rd time constant from dpb to dr TF      SEC-1      None
C  TQBDE1   1st time constant from dqb to de TF      SEC-1      None
C  TQBDE2   2nd time constant from dqb to de TF      SEC-1      None
C  TQBDT    Time constant from dqb to DTH TF         SEC-1      None
C  TR       roll mode time constant                  SEC-1      None
C  TRBDA    time constant from drb to da TF          SEC-1      None
C  TRBDR    time constant from drb to dr TF          SEC-1      None
C  TS       spiral mode time constant                SEC-1      None
C  TUDE     time constant from dU to de TF           SEC-1      None
C  TUDT     Time constant from dU to DTH TF          SEC-1     -------
C  XIXX     Vehicle X – Moment Of Inertia about Cg   SLUG-FT2   N/A
C  XIXZ     Vehicle Xz Prod Of Inertia about Cg      SLUG-FT2   N/A
C  XIYY     Vehicle Y – Moment Of Inertia about Cg   SLUG-FT2   N/A
C  XIZZ     Vehicle Z – Moment Of Inertia about Cg   SLUG-FT2   N/A
C  XMASS    Mass Of Vehicle (incl Fuel)              SLUG       N/A
C  ZETAALPHADE
C           damping ratio from dalpha to de TF       None       None
C  ZETABETADA
C           Damping ratio from dbeta to DA TF        None       None
C  ZETAD    dutch roll damping ratio                 None       None
C  ZETAP     hugoid damping ratio                    None       None
C  ZETAPBDA damping ratio from dpb to da TF          None       None
C  ZETARBDA damping ratio from drbi to da TF         None       None
C  ZETARBDR damping ratio from drb to dr TF          None       None
C  ZETASP   short period damping ratio               None       None
C  ZETAUDE  damping ratio from dU to de TF           None       None
C  ZETAUDT  damping ratio from dU to dth TF          None       None
C
C---------------------------------------------------------------------
C
C    OUTPUTS:
C
C  ALFA     Angle Of Attack                          DEG        ANU
C  ALFAR    Angle Of Attack                          RAD        ANU
C  ALFA_TRIM
C           Target angle of attack for trim          DEG        ANU
C  ALFD     Angle Of Attack Rate                     RAD/SEC    ANU
C  ALT      Altitude Of Aircraft Above Sea Lvl       FT         UP
C  ALTD     Altitude Rate Of Change                  FT/SEC     DOWN
C  BETA     Sideslip Angle                           DEG        ANL
C  BETAR    Sideslip Angle                           RAD        ANL
C  BETA_TRIM
C           Target angle of sideslip for trim        DEG        ANL
C  BETD     Sideslip Angle Rate                      RAD/SEC    ANR
C  DAREF    aileron reference deflection             RADIANS    +RWD
C  DDA      Change in aileron                        DEG        +RWD
C  DDE      Change in elevator                       DEG        +TED
C  DDR      Change in rudder deflection              DEG        +TER
C  DDTH     Change in throttle position              -------   -------
C  DEREF    elevator reference deflection            RADIANS    +RWD
C  DRREF    rudder reference deflection              RADIANS    +RWD
C  DTHREF   Throttle reference deflection            RADIANS    +RWD
C  FAX      Total Aero Force, X-body                 LB         FWD
```

```
C  FAY      Total Aero Force, Y-body                LB         RT
C  FAZ      Total Aero Force, Z-body                LB         DOWN
C  NX       Acceleration at center of gravity       Gs         FWD
C  NY       Acceleration at center of gravity       Gs         RT
C  NZ       Acceleration at center of gravity       Gs         DOWN
C  PB       Aircraft Roll Velocity, Body Frame      RAD/SEC    RWD
C  PBD      Aircraft Roll Acceleration, B-frame     RAD/SEC2   RWD
C  PBDEG    Roll rate, body axis                    DEG/SEC    RWD
C  PHI      Roll Euler Angle, L (local) Frame       DEG        RWD
C  PHID     Euler roll rate, L_frame                RAD/SEC    RWD
C  PHIR     Roll Angle, L-frame                     RAD        RWD
C  PSI      Yaw Euler Angle, L (local) Frame        DEG        ANR
C  PSID     Euler yaw rate, L-frame                 RAD/SEC    ANR
C  PSIR     Yaw Angle, L-frame                      RAD        ANR
C  QB       Aircraft Pitch Velocity, B-frame        RAD/SEC    ANU
C  QBD      Aircraft Pitch Accel, B-frame           RAD/SEC2   ANU
C  QBDEG    Pitch rate, body axis                   DEG/SEC    ANU
C  RB       Aircraft Yaw Velocity, B-frame          RAD/SEC    ANR
C  RBD      Aircraft Yaw Acceleration, B-frame      RAD/SEC2   ANR
C  RBDEG    Yaw rate, body axis                     DEG/SEC    ANR
C  S0LD     Lat-Dir s^0 coefficient                 RADIANS    +RWD
C  S0LONG   Longitudinal s^0 coefficient            RADIANS    +RWD
C  S1LD     Lat-Dir s^1 coefficient                 RADIANS    +RWD
C  S1LONG   Longitudinal s^1 coefficient            RADIANS    +RWD
C  S2LD     Lat-Dir s^2 coefficient                 RADIANS    +RWD
C  S2LONG   Longitudinal s^2 coefficient            RADIANS    +RWD
C  S3LD     Lat-Dir s^3 coefficient                 RADIANS    +RWD
C  S3LONG   Longitudinal s^3 coefficient            RADIANS    +RWD
C  TAL      Aero Rolling moment, X-body             FT-LB      RWD
C  TAM      Aero pitch moment, Y-body               FT-LB      ANU
C  TAN      Aero Yawing moment, Z-body              FT-LB      ANR
C  TF_ALFA  TFModel Angle of Attack                 None       None
C  TF_ALFD  Angle of Attack Deriv.                  None       None
C  TF_ALT   TFModel Altitude                        None       None
C  TF_ALTD  Altitude Deriv.                         None       None
C  TF_BETA  TFModel Sideslip Angle                  None       None
C  TF_BETD  Sideslip Angle Deriv.                   None       None
C  TF_PB    TFModel Roll Rate                       None       None
C  TF_PBD   Roll Rate Deriv.                        None       None
C  TF_PHI   TFModel Roll Angle                      None       None
C  TF_PHID  Roll Angle Deriv.                       None       None
C  TF_PSI   TFModel Heading Angle                   None       None
C  TF_PSID  Heading Angle Deriv.                    None       None
C  TF_QB    TFModel Pitch Rate                      None       None
C  TF_QBD   Pitch Rate Deriv.                       None       None
C  TF_RB    TFModel Yaw Rate                        None       None
C  TF_RBD   Roll Rate Deriv.                        None       None
C  TF_THED  Pitch Angle Deriv.                      None       None
C  TF_THET  TFModel Pitch Angle                     None       None
C  TF_VRW   TFModel Total Velocity                  None       None
C  TF_VRWD  Total Velocity Deriv.                   None       None
C  TF_XCG   TFModel XCG Location                    None       None
C  TF_XCGD  XCG Location Deriv.                     None       None
C  TF_YCG   TFModel YCG Location                    None       None
C  TF_YCGD  YCG Location Deriv.                     None       None
C  THED     Euler pitch rate, L-frame               RAD/SEC    ANU
C  THET     Pitch Euler Angle, L (local) Frame      DEG        ANU
C  THETR    Pitch Angle, L-frame                    RAD        ANU
```

```
C  TIME     Time Since Start Of Operate Mode         SEC        N/A
C  U        X-Directional Velocity, Body             FT/S       FWD
C  V        Y-Directional Velocity, Body             FT/S       RIGHT
C  VE       Eastward Velocity Over Fixed Earth       FT/SEC     EAST
C  VN       Northward Velocity Over Earth            FT/SEC     NORTH
C  VRW      Velocity w.r.t. wind (true airspeed)     FT/SEC     N/A
C  W        Z-Directional Velocity, Body             FT/S       UP
C  XCGLOC   X-location in inertial space of CG       FT        -------
C  XDA1     Internal State 1 to Aileron              None       None
C  XDA2     Internal State 2 to Aileron              None       None
C  XDA3     Internal State 3 to Aileron              None       None
C  XDA4     Internal State 4 to Aileron              None       None
C  XDDA1    Internal State Derivative 1 to Aileron   None       None
C  XDDA2    Internal State Derivative 2 to Aileron   None       None
C  XDDA3    Internal State Derivative 3 to Aileron   None       None
C  XDDA4    Internal State Derivative 4 to Aileron   None       None
C  XDDE1    Internal State Derivative 1 to Elevator  None       None
C  XDDE2    Internal State Derivative 2 to Elevator  None       None
C  XDDE3    Internal State Derivative 3 to Elevator  None       None
C  XDDE4    Internal State Derivative 4 to Elevator  None       None
C  XDDR1    Internal State Derivative 1 to Rudder    None       None
C  XDDR2    Internal State Derivative 2 to Rudder    None       None
C  XDDR3    Internal State Derivative 3 to Rudder    None       None
C  XDDR4    Internal State Derivative 4 to Rudder    None       None
C  XDDT1    Internal State Derivative 1 to Throttle  None       None
C  XDDT2    Internal State Derivative 2 to Throttle  None       None
C  XDDT3    Internal State Derivative 3 to Throttle  None       None
C  XDDT4    Internal State Derivative 4 to Throttle  None       None
C  XDE1     Internal State 1 to Elevator             None       None
C  XDE2     Internal State 2 to Elevator             None       None
C  XDE3     Internal State 3 to Elevator             None       None
C  XDE4     Internal State 4 to Elevator             None       None
C  XDR1     Internal State 1 to Rudder               None       None
C  XDR2     Internal State 2 to Rudder               None       None
C  XDR3     Internal State 3 to Rudder               None       None
C  XDR4     Internal State 4 to Rudder               None       None
C  XDT1     Internal State 1 to Throttle             None       None
C  XDT2     internal State 2 to Throttle             None       None
C  XDT3     Internal State 3 to Throttle             None       None
C  XDT4     Internal State 4 to Throttle             None       None
C  YCGLOC   Y-location in inertial space of CG       FT        -------
C  ZCGLOC   Z-location in inertial space of CG       FT        -------
C
C-----------------------------------------------------------------------
```

```
      SUBROUTINE TRANSFERFUNCTIONS

C-----------------------------------------------------------------------
C
C     DECLARATION SECTION
C
C-----------------------------------------------------------------------

      IMPLICIT NONE

C           ** INPUTS:

      INTEGER*4 DEMOCASE, ICASE, IMODE
      REAL*4 D2R, DR, DT, DTH, G, DA, DE, KALPHADE, KALPHADT, KBETADA
      REAL*4 KBETADR, KPBDA, KPBDR, KQBDE, KQBDT, KRBDA, KRBDR, KUDE
      REAL*4 KUDT, LAMDAFX, LAMDAFY, LAMDAFZ, LAMDATL, LAMDATM
      REAL*4 LAMDATN, OMEGAALPHADE, OMEGABETADA, OMEGAD, OMEGAP
      REAL*4 OMEGAPBDA, OMEGARBDA, OMEGARBDR, OMEGASP, OMEGAUDE
      REAL*4 OMEGAUDT, PIOVR2, R2D, REF_ALFA, REF_ALT, REF_BETA
      REAL*4 REF_PB, REF_PHI, REF_PSI, REF_QB, REF_RB, REF_THET
      REAL*4 REF_VRW, REF_XCG, REF_YCG, TALPHADE, TALPHADT1
      REAL*4 TALPHADT2, TBETADA, TBETADR1, TBETADR2, TBETADR3, TPBDA
      REAL*4 TPBDR1, TPBDR2, TPBDR3, TQBDE1, TQBDE2, TQBDT, TR, TRBDA
      REAL*4 TRBDR, TS, TUDE, TUDT, XIXX, XIXZ, XIYY, XIZZ, XMASS
      REAL*4 ZETAALPHADE, ZETABETADA, ZETAD, ZETAP, ZETAPBDA
      REAL*4 ZETARBDA, ZETARBDR, ZETASP, ZETAUDE, ZETAUDT

C           ** OUTPUTS:

      REAL*4 ALFA, ALFAR, ALFA_TRIM, ALFD, ALT, ALTD, BETA, BETAR
      REAL*4 BETA_TRIM, BETD, DAREF, DDA, DDE, DDR, DDTH, DEREF
      REAL*4 DRREF, DTHREF, FAX, FAY, FAZ, NX, NY, NZ, PB, PBD, PBDEG
      REAL*4 PHI, PHID, PHIR, PSI, PSID, PSIR, QB, QBD, QBDEG, RB
      REAL*4 RBD, RBDEG, S0LD, S0LONG, S1LD, S1LONG, S2LD, S2LONG
      REAL*4 S3LD, S3LONG, TAL, TAM, TAN, TF_ALFA, TF_ALFD, TF_ALT
      REAL*4 TF_ALTD, TF_BETA, TF_BETD, TF_PB, TF_PBD, TF_PHI
      REAL*4 TF_PHID, TF_PSI, TF_PSID, TF_QB, TF_QBD, TF_RB, TF_RBD
      REAL*4 TF_THED, TF_THET, TF_VRW, TF_VRWD, TF_XCG, TF_XCGD
      REAL*4 TF_YCG, TF_YCGD, THED, THET, THETR, TIME, U, V, VE, VN
      REAL*4 VRW, W, XCGLOC, XDA1, XDA2, XDA3, XDA4, XDDA1, XDDA2
      REAL*4 XDDA3, XDDA4, XDDE1, XDDE2, XDDE3, XDDE4, XDDR1, XDDR2
      REAL*4 XDDR3, XDDR4, XDDT1, XDDT2, XDDT3, XDDT4, XDE1, XDE2
      REAL*4 XDE3, XDE4, XDR1, XDR2, XDR3, XDR4, XDT1, XDT2, XDT3
      REAL*4 XDT4, YCGLOC, ZCGLOC

C           ** OTHER LOCALS:

      BYTE ATMPAR, CONPAR, KONSTANTS, LCLBUF, SHIPPAR, SIMPAR
      BYTE TFMODPAR
      LOGICAL*4 INITIALIZED
      REAL*4 KTHETADE, KTHETADT, KPSIDR, OMEGAALPHA, OMEGAPHIDA
      REAL*4 OMEGAPSIDA, OMEGAPSIDR, KPHIDA, KPHIDR, TALPHA, KPSIDA
      REAL*4 TPHIDR1, TPHIDR2, TPSIDA, TPSIDR, TTHETA1, TTHETA2
      REAL*4 TTHETADE1, TTHETADE2, TTHETADT, ZETAALPHA, ZETAPHIDA
      REAL*4 ZETAPSIDA, ZETAPSIDR
```

```
C-----------------------------------------------------------------------
C
C     COMMON SECTION
C
C-----------------------------------------------------------------------



      COMMON/ XTFMOD / TFMODPAR(704)
      COMMON/ SHELL1 / CONPAR(424)
      COMMON/ SHELL2 / SIMPAR(1024)
      COMMON/ KONSTANT / KONSTANTS(40)

C-----------------------------------------------------------------------
C
C     EQUIVALENCE SECTION
C
C-----------------------------------------------------------------------

C          ** INPUTS:

      EQUIVALENCE( KONSTANTS(5), D2R)
      EQUIVALENCE( TFMODPAR(5), DA)
      EQUIVALENCE( TFMODPAR(9), DE)
      EQUIVALENCE( TFMODPAR(289), DEMOCASE)
      EQUIVALENCE( TFMODPAR(13), DR)
      EQUIVALENCE( CONPAR(145), DT)
      EQUIVALENCE( TFMODPAR(1), DTH)
      EQUIVALENCE( KONSTANTS(1), G)
      EQUIVALENCE( TFMODPAR(285), ICASE)
      EQUIVALENCE( CONPAR(1), IMODE)
      EQUIVALENCE( TFMODPAR(21), KALPHADE)
      EQUIVALENCE( TFMODPAR(53), KALPHADT)
      EQUIVALENCE( TFMODPAR(25), KBETADA)
      EQUIVALENCE( TFMODPAR(29), KBETADR)
      EQUIVALENCE( TFMODPAR(33), KPBDA)
      EQUIVALENCE( TFMODPAR(37), KPBDR)
      EQUIVALENCE( TFMODPAR(41), KQBDE)
      EQUIVALENCE( TFMODPAR(57), KQBDT)
      EQUIVALENCE( TFMODPAR(45), KRBDA)
      EQUIVALENCE( TFMODPAR(237), KRBDR)
      EQUIVALENCE( TFMODPAR(17), KUDE)
      EQUIVALENCE( TFMODPAR(49), KUDT)
      EQUIVALENCE( TFMODPAR(341), LAMDAFX)
      EQUIVALENCE( TFMODPAR(345), LAMDAFY)
      EQUIVALENCE( TFMODPAR(349), LAMDAFZ)
      EQUIVALENCE( TFMODPAR(353), LAMDATL)
      EQUIVALENCE( TFMODPAR(357), LAMDATM)
      EQUIVALENCE( TFMODPAR(361), LAMDATN)
      EQUIVALENCE( TFMODPAR(133), OMEGAALPHADE)
      EQUIVALENCE( TFMODPAR(313), OMEGABETADA)
      EQUIVALENCE( TFMODPAR(121), OMEGAD)
      EQUIVALENCE( TFMODPAR(125), OMEGAP)
      EQUIVALENCE( TFMODPAR(141), OMEGAPBDA)
      EQUIVALENCE( TFMODPAR(145), OMEGARBDA)
      EQUIVALENCE( TFMODPAR(149), OMEGARBDR)
      EQUIVALENCE( TFMODPAR(129), OMEGASP)
      EQUIVALENCE( TFMODPAR(137), OMEGAUDE)
```

```
      EQUIVALENCE( TFMODPAR(153), OMEGAUDT)
      EQUIVALENCE( KONSTANTS(29), PIOVR2)
      EQUIVALENCE( KONSTANTS(9), R2D)
      EQUIVALENCE( TFMODPAR(369), REF_ALFA)
      EQUIVALENCE( TFMODPAR(409), REF_ALT)
      EQUIVALENCE( TFMODPAR(373), REF_BETA)
      EQUIVALENCE( TFMODPAR(389), REF_PB)
      EQUIVALENCE( TFMODPAR(377), REF_PHI)
      EQUIVALENCE( TFMODPAR(385), REF_PSI)
      EQUIVALENCE( TFMODPAR(393), REF_QB)
      EQUIVALENCE( TFMODPAR(397), REF_RB)
      EQUIVALENCE( TFMODPAR(381), REF_THET)
      EQUIVALENCE( TFMODPAR(365), REF_VRW)
      EQUIVALENCE( TFMODPAR(401), REF_XCG)
      EQUIVALENCE( TFMODPAR(405), REF_YCG)
      EQUIVALENCE( TFMODPAR(73), TALPHADE)
      EQUIVALENCE( TFMODPAR(305), TALPHADT1)
      EQUIVALENCE( TFMODPAR(309), TALPHADT2)
      EQUIVALENCE( TFMODPAR(77), TBETADA)
      EQUIVALENCE( TFMODPAR(85), TBETADR1)
      EQUIVALENCE( TFMODPAR(89), TBETADR2)
      EQUIVALENCE( TFMODPAR(93), TBETADR3)
      EQUIVALENCE( TFMODPAR(697), TPBDA)
      EQUIVALENCE( TFMODPAR(97), TPBDR1)
      EQUIVALENCE( TFMODPAR(101), TPBDR2)
      EQUIVALENCE( TFMODPAR(701), TPBDR3)
      EQUIVALENCE( TFMODPAR(105), TQBDE1)
      EQUIVALENCE( TFMODPAR(109), TQBDE2)
      EQUIVALENCE( TFMODPAR(81), TQBDT)
      EQUIVALENCE( TFMODPAR(65), TR)
      EQUIVALENCE( TFMODPAR(113), TRBDA)
      EQUIVALENCE( TFMODPAR(117), TRBDR)
      EQUIVALENCE( TFMODPAR(69), TS)
      EQUIVALENCE( TFMODPAR(61), TUDE)
      EQUIVALENCE( TFMODPAR(301), TUDT)
      EQUIVALENCE( SIMPAR(345), XIXX)
      EQUIVALENCE( SIMPAR(357), XIXZ)
      EQUIVALENCE( SIMPAR(349), XIYY)
      EQUIVALENCE( SIMPAR(353), XIZZ)
      EQUIVALENCE( SIMPAR(369), XMASS)
      EQUIVALENCE( TFMODPAR(169), ZETAALPHADE)
      EQUIVALENCE( TFMODPAR(317), ZETABETADA)
      EQUIVALENCE( TFMODPAR(157), ZETAD)
      EQUIVALENCE( TFMODPAR(161), ZETAP)
      EQUIVALENCE( TFMODPAR(173), ZETAPBDA)
      EQUIVALENCE( TFMODPAR(177), ZETARBDA)
      EQUIVALENCE( TFMODPAR(181), ZETARBDR)
      EQUIVALENCE( TFMODPAR(165), ZETASP)
      EQUIVALENCE( TFMODPAR(185), ZETAUDE)
      EQUIVALENCE( TFMODPAR(189), ZETAUDT)

C          ** OUTPUTS:

      EQUIVALENCE( SIMPAR(97), ALFA)
      EQUIVALENCE( SIMPAR(105), ALFAR)
      EQUIVALENCE( SIMPAR(877), ALFA_TRIM)
      EQUIVALENCE( SIMPAR(113), ALFD)
      EQUIVALENCE( SIMPAR(241), ALT)
```

```
      EQUIVALENCE( SIMPAR(229), ALTD)
      EQUIVALENCE( SIMPAR(101), BETA)
      EQUIVALENCE( SIMPAR(109), BETAR)
      EQUIVALENCE( SIMPAR(881), BETA_TRIM)
      EQUIVALENCE( SIMPAR(117), BETD)
      EQUIVALENCE( TFMODPAR(513), DAREF)
      EQUIVALENCE( TFMODPAR(265), DDA)
      EQUIVALENCE( TFMODPAR(269), DDE)
      EQUIVALENCE( TFMODPAR(273), DDR)
      EQUIVALENCE( TFMODPAR(277), DDTH)
      EQUIVALENCE( TFMODPAR(517), DEREF)
      EQUIVALENCE( TFMODPAR(521), DRREF)
      EQUIVALENCE( TFMODPAR(525), DTHREF)
      EQUIVALENCE( SIMPAR(393), FAX)
      EQUIVALENCE( SIMPAR(397), FAY)
      EQUIVALENCE( SIMPAR(401), FAZ)
      EQUIVALENCE( SIMPAR(657), NX)
      EQUIVALENCE( SIMPAR(661), NY)
      EQUIVALENCE( SIMPAR(665), NZ)
      EQUIVALENCE( SIMPAR(145), PB)
      EQUIVALENCE( SIMPAR(157), PBD)
      EQUIVALENCE( SIMPAR(625), PBDEG)
      EQUIVALENCE( SIMPAR(1), PHI)
      EQUIVALENCE( SIMPAR(25), PHID)
      EQUIVALENCE( SIMPAR(13), PHIR)
      EQUIVALENCE( SIMPAR(9), PSI)
      EQUIVALENCE( SIMPAR(33), PSID)
      EQUIVALENCE( SIMPAR(21), PSIR)
      EQUIVALENCE( SIMPAR(149), QB)
      EQUIVALENCE( SIMPAR(161), QBD)
      EQUIVALENCE( SIMPAR(629), QBDEG)
      EQUIVALENCE( SIMPAR(153), RB)
      EQUIVALENCE( SIMPAR(165), RBD)
      EQUIVALENCE( SIMPAR(633), RBDEG)
      EQUIVALENCE( TFMODPAR(545), S0LD)
      EQUIVALENCE( TFMODPAR(529), S0LONG)
      EQUIVALENCE( TFMODPAR(549), S1LD)
      EQUIVALENCE( TFMODPAR(533), S1LONG)
      EQUIVALENCE( TFMODPAR(553), S2LD)
      EQUIVALENCE( TFMODPAR(537), S2LONG)
      EQUIVALENCE( TFMODPAR(557), S3LD)
      EQUIVALENCE( TFMODPAR(541), S3LONG)
      EQUIVALENCE( SIMPAR(469), TAL)
      EQUIVALENCE( SIMPAR(473), TAM)
      EQUIVALENCE( SIMPAR(477), TAN)
      EQUIVALENCE( TFMODPAR(469), TF_ALFA)
      EQUIVALENCE( TFMODPAR(421), TF_ALFD)
      EQUIVALENCE( TFMODPAR(509), TF_ALT)
      EQUIVALENCE( TFMODPAR(461), TF_ALTD)
      EQUIVALENCE( TFMODPAR(473), TF_BETA)
      EQUIVALENCE( TFMODPAR(425), TF_BETD)
      EQUIVALENCE( TFMODPAR(489), TF_PB)
      EQUIVALENCE( TFMODPAR(441), TF_PBD)
      EQUIVALENCE( TFMODPAR(477), TF_PHI)
      EQUIVALENCE( TFMODPAR(429), TF_PHID)
      EQUIVALENCE( TFMODPAR(485), TF_PSI)
      EQUIVALENCE( TFMODPAR(437), TF_PSID)
      EQUIVALENCE( TFMODPAR(493), TF_QB)
```

```
      EQUIVALENCE( TFMODPAR(445), TF_QBD)
      EQUIVALENCE( TFMODPAR(497), TF_RB)
      EQUIVALENCE( TFMODPAR(449), TF_RBD)
      EQUIVALENCE( TFMODPAR(433), TF_THED)
      EQUIVALENCE( TFMODPAR(481), TF_THET)
      EQUIVALENCE( TFMODPAR(465), TF_VRW)
      EQUIVALENCE( TFMODPAR(417), TF_VRWD)
      EQUIVALENCE( TFMODPAR(501), TF_XCG)
      EQUIVALENCE( TFMODPAR(453), TF_XCGD)
      EQUIVALENCE( TFMODPAR(505), TF_YCG)
      EQUIVALENCE( TFMODPAR(457), TF_YCGD)
      EQUIVALENCE( SIMPAR(29), THED)
      EQUIVALENCE( SIMPAR(5), THET)
      EQUIVALENCE( SIMPAR(17), THETR)
      EQUIVALENCE( SIMPAR(577), TIME)
      EQUIVALENCE( TFMODPAR(561), U)
      EQUIVALENCE( TFMODPAR(565), V)
      EQUIVALENCE( SIMPAR(185), VE)
      EQUIVALENCE( SIMPAR(181), VN)
      EQUIVALENCE( SIMPAR(205), VRW)
      EQUIVALENCE( TFMODPAR(569), W)
      EQUIVALENCE( SIMPAR(725), XCGLOC)
      EQUIVALENCE( TFMODPAR(633), XDA1)
      EQUIVALENCE( TFMODPAR(637), XDA2)
      EQUIVALENCE( TFMODPAR(641), XDA3)
      EQUIVALENCE( TFMODPAR(645), XDA4)
      EQUIVALENCE( TFMODPAR(649), XDDA1)
      EQUIVALENCE( TFMODPAR(653), XDDA2)
      EQUIVALENCE( TFMODPAR(657), XDDA3)
      EQUIVALENCE( TFMODPAR(661), XDDA4)
      EQUIVALENCE( TFMODPAR(617), XDDE1)
      EQUIVALENCE( TFMODPAR(621), XDDE2)
      EQUIVALENCE( TFMODPAR(625), XDDE3)
      EQUIVALENCE( TFMODPAR(629), XDDE4)
      EQUIVALENCE( TFMODPAR(681), XDDR1)
      EQUIVALENCE( TFMODPAR(685), XDDR2)
      EQUIVALENCE( TFMODPAR(689), XDDR3)
      EQUIVALENCE( TFMODPAR(693), XDDR4)
      EQUIVALENCE( TFMODPAR(585), XDDT1)
      EQUIVALENCE( TFMODPAR(589), XDDT2)
      EQUIVALENCE( TFMODPAR(593), XDDT3)
      EQUIVALENCE( TFMODPAR(597), XDDT4)
      EQUIVALENCE( TFMODPAR(601), XDE1)
      EQUIVALENCE( TFMODPAR(605), XDE2)
      EQUIVALENCE( TFMODPAR(609), XDE3)
      EQUIVALENCE( TFMODPAR(613), XDE4)
      EQUIVALENCE( TFMODPAR(665), XDR1)
      EQUIVALENCE( TFMODPAR(669), XDR2)
      EQUIVALENCE( TFMODPAR(673), XDR3)
      EQUIVALENCE( TFMODPAR(677), XDR4)
      EQUIVALENCE( TFMODPAR(413), XDT1)
      EQUIVALENCE( TFMODPAR(573), XDT2)
      EQUIVALENCE( TFMODPAR(577), XDT3)
      EQUIVALENCE( TFMODPAR(581), XDT4)
      EQUIVALENCE( SIMPAR(729), YCGLOC)
      EQUIVALENCE( SIMPAR(733), ZCGLOC)
```

```
C-----------------------------------------------------------------------
C
C      INITIALIZATION SECTION
C
C-----------------------------------------------------------------------

      IF( IMODE.LE.-2 .OR. .NOT. Initialized ) THEN

          ALFA_TRIM  = REF_ALFA * R2D
          BETA_TRIM  = REF_BETA * R2D

          OMEGASP      =  2.22
          ZETASP       =  0.454
          OMEGAP       =  0.0845
          ZETAP        =  0.0842

          KUDT         =  19.72
          TUDT         = -270.27027
          OMEGAUDT     =  2.22
          ZETAUDT      =  0.454

          KALPHADT     = -0.002612
          TALPHADT1    =  0.51546
          TALPHADT2    =  0.79365

          KQBDT        =  0.01117
          TQBDT        =  0.51546

          KUDE         =  0.09849
          TUDE         =  1.72117
          OMEGAUDE     =  7.39
          ZETAUDE      =  0.207

          KALPHADE     = -0.00159
          TALPHADE     =  0.012165
          OMEGAALPHADE =  0.0933
          ZETAALPHADE  =  0.0943

          KQBDE        = -0.1386
          TQBDE1       =  1.42045
          TQBDE2       = 68.49315



          OMEGAD       =  3.03
          ZETAD        =  0.12
          TR           =  0.3937
          TS           = 94.33962

          KBETADA      =  0.0002173
          TBETADA      = -0.019011
          OMEGABETADA  =  1.13
          ZETABETADA   =  0.276

          KPBDA        = -0.5415
          TPBDA        = -235.84906
          OMEGAPBDA    =  2.98
          ZETAPBDA     =  0.111
```

```
            KRBDA         = -0.02415
            TRBDA         =  0.98039
            OMEGARBDA     =  3.52
            ZETARBDA      =  0.131

            KBETADR       =  0.0005933
            TBETADR1      =  0.010776
            TBETADR2      =  0.3937
            TBETADR3      = -1329.78723

            KPBDR         =  0.09382
            TPBDR1        =  0.40984
            TPBDR2        = -0.43668
            TPBDR3        = -234.19204

            KRBDR         = -0.04893
            TRBDR         =  0.39063
            OMEGARBDR     =  0.520
            ZETARBDR      =  0.313

            LAMDAFX           =  0.0
            LAMDAFY           =  0.0
            LAMDAFZ           =  0.0
            LAMDATL           =  0.0
            LAMDATM           =  0.0
            LAMDATN           =  0.0

      ENDIF

C-----------------------------------------------------------------------
C
C      RESET SECTION
C
C-----------------------------------------------------------------------

      IF( IMODE.LT.0 .OR. .NOT. Initialized ) THEN

      Initialized = .TRUE.

      TIME = 0.0
C
      IF (ICASE.EQ.1) THEN       ! CASE A
            OMEGASP    =  3.13
            ZETASP     =  0.704
            OMEGAP     =  0.0759
            ZETAP =  0.224
      ELSEIF (ICASE.EQ.2) THEN        ! CASE B
            OMEGASP    =  2.34
            ZETASP     =  0.607
            OMEGAP     =  0.0806
            ZETAP =  0.199
      ELSEIF (ICASE.EQ.3) THEN        ! CASE C
            OMEGASP    =  0.864
            ZETASP     =  0.995
            OMEGAP     =  0.0974
            ZETAP =  0.0924
      END IF
```

```
       S0LONG = -OMEGASP*OMEGASP*OMEGAP*OMEGAP
       S1LONG = -2*(ZETASP*OMEGASP*OMEGAP*OMEGAP+
      &                ZETAP*OMEGAP*OMEGASP*OMEGASP)
       S2LONG = -OMEGAP*OMEGAP
      &                -4*ZETASP*ZETAP*OMEGASP*OMEGAP-OMEGASP*OMEGASP
       S3LONG = -2*(ZETAP*OMEGAP+ZETASP*OMEGASP)
C
       S3LD = -2*ZETAD*OMEGAD-1/TR-1/TS
       S2LD = -OMEGAD*OMEGAD
      &           -2*(ZETAD*OMEGAD/TR+ZETAD*OMEGAD/TS)-1/TS/TR
       S1LD = -OMEGAD*OMEGAD/TR-OMEGAD*OMEGAD/TS
      &           -2*ZETAD*OMEGAD/TS/TR
       S0LD = -OMEGAD*OMEGAD/TS/TR

       IF (ICASE.EQ.4) THEN     ! CASE D
             S0LONG =  0.0024623
             S1LONG =  0.0599280
             S2LONG =  1.7651000
             S3LONG = -0.6340000
       END IF

C       STATES DUE TO CONTROLS

       XDT1 = 0.0
       XDT2 = 0.0
       XDT3 = 0.0
       XDT4 = 0.0
       XDE1 = 0.0
       XDE2 = 0.0
       XDE3 = 0.0
       XDE4 = 0.0
       XDA1 = 0.0
       XDA2 = 0.0
       XDA3 = 0.0
       XDA4 = 0.0
       XDR1 = 0.0
       XDR2 = 0.0
       XDR3 = 0.0
       XDR4 = 0.0

C       XD( ) = TIME DERIVATIVES OF X( )

       XDDT1 = 0.0
       XDDT2 = 0.0
       XDDT3 = 0.0
       XDDT4 = 0.0
       XDDE1 = 0.0
       XDDE2 = 0.0
       XDDE3 = 0.0
       XDDE4 = 0.0
       XDDA1 = 0.0
       XDDA2 = 0.0
       XDDA3 = 0.0
       XDDA4 = 0.0
       XDDR1 = 0.0
       XDDR2 = 0.0
       XDDR3 = 0.0
       XDDR4 = 0.0
```

```
C      YREF( ) = REFERENCE OUTPUTS

       REF_VRW  = 500.0
       REF_ALFA = 3.788625/57.3
       REF_BETA = 0.0
       REF_PHI  = 0.0
       REF_THET = REF_ALFA
       REF_PSI  = 0.0
       REF_PB   = 0.0
       REF_QB   = 0.0
       REF_RB   = 0.0
       REF_XCG  = 0.0
       REF_YCG  = 0.0
       REF_ALT  = 10000.0

       TF_VRW  = REF_VRW
       TF_ALFA = REF_ALFA
       TF_BETA = 0.0
       TF_PHI  = 0.0
       TF_THET = REF_ALFA
       TF_PSI  = 0.0
       TF_PB   = 0.0
       TF_QB   = 0.0
       TF_RB   = 0.0
       TF_XCG  = 0.0
       TF_YCG  = 0.0
       TF_ALT  = REF_ALT

       TF_VRWD  = 0.0
       TF_ALFD  = 0.0
       TF_BETD  = 0.0
       TF_PHID  = 0.0
       TF_THED  = 0.0
       TF_PSID  = 0.0
       TF_PBD   = 0.0
       TF_QBD   = 0.0
       TF_RBD   = 0.0
       TF_XCGD  = REF_VRW
       TF_YCGD  = 0.0
       TF_ALTD  = 0.0

       VRW    = TF_VRW
       ALFAR  = TF_ALFA
       BETAR  = TF_BETA
       PHIR   = TF_PHI
       THETR  = TF_THET
       PSIR   = TF_PSI
       PB     = TF_PB
       QB     = TF_QB
       RB     = TF_RB
       XCGLOC = TF_XCG
       YCGLOC = TF_YCG
       ALT    = TF_ALT
       ZCGLOC = -TF_ALT

       U = REF_VRW
       V = 0.0
       W = 0.0
```

```
C      REFERENCE CONTROL DEFLECTIONS

       DTHREF = 0.1962366
       DEREF  = -3.851317
       DAREF  = 0
       DRREF  = 0

       DTH = DTHREF
       DE  = DEREF
       DA  = DAREF
       DR  = DRREF

       ENDIF

C-------------------------------------------------------------------------
C
C      RUN SECTION
C
C-------------------------------------------------------------------------

C        Change in control inputs

       DDTH = DTH – DTHREF
       DDA  = DA – DAREF
       DDE  = DE – DEREF
       DDR  = DR – DRREF
C
C      TF_VRW = Total Velocity (ft/sec)
C      TF_ALFA = alpha (deg)
C      TF_BETA = beta (deg)
C      TF_PHI = phi (deg)
C      TF_THET = theta (deg)
C      TF_PSI = psi (deg)
C      TF_PB = p (deg/sec)
C      TF_QB = q (deg/sec)
C      TF_RB = r (deg/sec)
C      TF_XCG = North position (ft)
C      TF_YCG = East position (ft)
C      TF_ALT = altitude (ft)
C
       TF_VRW = KUDT*(XDT1+(2*ZETAUDT*OMEGAUDT+1/TUDT)*XDT2
      & +(OMEGAUDT*OMEGAUDT+2*ZETAUDT*OMEGAUDT/TUDT)*XDT3
      & +(OMEGAUDT*OMEGAUDT/TUDT)*XDT4)
      & +KUDE*(XDE1+(2*ZETAUDE*OMEGAUDE+1/TUDE)*XDE2
      & +(OMEGAUDE*OMEGAUDE+2*ZETAUDE*OMEGAUDE/TUDE)*XDE3
      & +(OMEGAUDE*OMEGAUDE/TUDE)*XDE4)
C
       TF_ALFA = KALPHADT*(XDT1+(1/TALPHADT1+1/TALPHADT2)*XDT2
      & +(1/TALPHADT1/TALPHADT2)*XDT3)
      & +KALPHADE*(XDE1+(2*ZETAALPHADE*OMEGAALPHADE+1/TALPHADE)*XDE2
      & +OMEGAALPHADE*(OMEGAALPHADE+2*ZETAALPHADE*1/TALPHADE)
      & *XDE3+(OMEGAALPHADE*OMEGAALPHADE/TALPHADE)*XDE4)
C
```

```
       TF_BETA = KBETADA*(XDA1
      & +(2*ZETABETADA*OMEGABETADA+1/TBETADA)*XDA2
      & +(OMEGABETADA*OMEGABETADA
      & +2*ZETABETADA*OMEGABETADA/TBETADA)*XDA3
      & +OMEGABETADA*OMEGABETADA/TBETADA*XDA4)
      & +KBETADR*(XDR1+(1/TBETADR1+1/TBETADR2+1/TBETADR3)*XDR2
      & +(1/TBETADR1/TBETADR2+1/TBETADR1/TBETADR3
      & +1/TBETADR2/TBETADR3)*XDR3
      & +(1/TBETADR1/TBETADR2/TBETADR3)*XDR4)
C
       TF_PB = KPBDA*(XDA1+(2*ZETAPBDA*OMEGAPBDA+1/TPBDA)*XDA2
      & +(2*ZETAPBDA*OMEGAPBDA/TPBDA+OMEGAPBDA*OMEGAPBDA)*XDA3
      & +OMEGAPBDA*OMEGAPBDA/TPBDA*XDA4)
      & +KPBDR*(XDR1+(1/TPBDR1+1/TPBDR2+1/TPBDR3)*XDR2
      & +(1/TPBDR1/TPBDR2+1/TPBDR1/TPBDR3+1/TPBDR2/TPBDR3)*XDR3
      & +(1/TPBDR1/TPBDR2/TPBDR3)*XDR4)
C
       TF_QB = KQBDT*(XDT2+1/TQBDT*XDT3)
      & +KQBDE*(XDE1+(1/TQBDE1+1/TQBDE2)*XDE2
      & +(1/TQBDE1/TQBDE2)*XDE3)
C
       TF_RB = KRBDA*(XDA1+(2*ZETARBDA*OMEGARBDA+1/TRBDA)*XDA2
      & +(OMEGARBDA*OMEGARBDA+2*ZETARBDA*OMEGARBDA/TRBDA)*XDA3
      & +(OMEGARBDA*OMEGARBDA/TRBDA)*XDA4)
      & +KRBDR*(XDR1+(2*ZETARBDR*OMEGARBDR+1/TRBDR)*XDR2
      & +(OMEGARBDR*OMEGARBDR+2*ZETARBDR*OMEGARBDR/TRBDR)*XDR3
      & +(OMEGARBDR*OMEGARBDR/TRBDR)*XDR4)
C
       TF_VRW  = TF_VRW  + REF_VRW
       TF_ALFA = TF_ALFA + REF_ALFA
       TF_BETA = TF_BETA + REF_BETA
C
       TF_PHI  = TF_PHI  + TF_PHID * DT
       TF_THET = TF_THET + TF_THED * DT
       TF_PSI  = TF_PSI  + TF_PSID * DT
C
       TF_PB   = TF_PB   + REF_PB
       TF_QB   = TF_QB   + REF_QB
       TF_RB   = TF_RB   + REF_RB
C
       TF_XCG  = TF_XCG  + TF_XCGD * DT
       TF_YCG  = TF_YCG  + TF_YCGD * DT
       TF_ALT  = TF_ALT  + TF_ALTD * DT
C
       U = TF_VRW * COS(TF_ALFA) * COS(TF_BETA)
       V = TF_VRW * SIN(TF_BETA)
       W = TF_VRW * COS(TF_BETA) * SIN(TF_ALFA)
C
C         calculate the time derivatives of the output variables
C
       TF_VRWD = KUDT*(XDDT1+(2*ZETAUDT*OMEGAUDT+1/TUDT)*XDDT2
      & +(OMEGAUDT*OMEGAUDT+2*ZETAUDT*OMEGAUDT/TUDT)*XDDT3
      & +(OMEGAUDT*OMEGAUDT/TUDT)*XDDT4)
      & +KUDE*(XDDE1+(2*ZETAUDE*OMEGAUDE+1/TUDE)*XDDE2
      & +(OMEGAUDE*OMEGAUDE+2*ZETAUDE*OMEGAUDE/TUDE)*XDDE3
      & +(OMEGAUDE*OMEGAUDE/TUDE)*XDDE4)
C
```

```
       TF_ALFD = KALPHADT*(XDDT1+(1/TALPHADT1+1/TALPHADT2)*XDDT2
      & +(1/TALPHADT1/TALPHADT2)*XDDT3)
      & +KALPHADE*(XDDE1+(2*ZETAALPHADE*OMEGAALPHADE+1/TALPHADE)*XDDE2
      & +OMEGAALPHADE*(OMEGAALPHADE+2*ZETAALPHADE*1/TALPHADE)
      & *XDDE3+(OMEGAALPHADE*OMEGAALPHADE/TALPHADE)*XDDE4)
C
       TF_BETD = KBETADA*(XDDA1
      & +(2*ZETABETADA*OMEGABETADA+1/TBETADA)*XDDA2
      & +(OMEGABETADA*OMEGABETADA
      & +2*ZETABETADA*OMEGABETADA/TBETADA)*XDDA3
      & +OMEGABETADA*OMEGABETADA/TBETADA*XDDA4)
      & +KBETADR*(XDDR1+(1/TBETADR1+1/TBETADR2+1/TBETADR3)*XDDR2
      & +(1/TBETADR1/TBETADR2+1/TBETADR1/TBETADR3
      & +1/TBETADR2/TBETADR3)*XDDR3
      & +(1/TBETADR1/TBETADR2/TBETADR3)*XDDR4)
C
       TF_PHID = TF_PB + (SIN(TF_THET)/COS(TF_THET))
      & *(TF_QB*SIN(TF_PHI)+TF_RB*COS(TF_PHI))
C
       TF_THED = TF_QB*COS(TF_PHI)-TF_RB*SIN(TF_PHI)
C
       TF_PSID = (TF_QB*SIN(TF_PHI)+TF_RB*COS(TF_PHI))/COS(TF_THET)
C
       TF_PBD = KPBDA*(XDDA1+(2*ZETAPBDA*OMEGAPBDA+1/TPBDA)*XDDA2
      & +(2*ZETAPBDA*OMEGAPBDA/TPBDA+OMEGAPBDA*OMEGAPBDA)*XDDA3
      & +OMEGAPBDA*OMEGAPBDA/TPBDA*XDDA4)
      & +KPBDR*(XDDR1+(1/TPBDR1+1/TPBDR2+1/TPBDR3)*XDDR2
      & +(1/TPBDR1/TPBDR2+1/TPBDR1/TPBDR3+1/TPBDR2/TPBDR3)*XDDR3
      & +(1/TPBDR1/TPBDR2/TPBDR3)*XDDR4)
C
       TF_QBD = KQBDT*(XDDT2+1/TQBDT*XDDT3)
      & +KQBDE*(XDDE1+(1/TQBDE1+1/TQBDE2)*XDDE2
      & +(1/TQBDE1/TQBDE2)*XDDE3)
C
       TF_RBD = KRBDA*(XDDA1+(2*ZETARBDA*OMEGARBDA+1/TRBDA)*XDDA2
      & +(OMEGARBDA*OMEGARBDA+2*ZETARBDA*OMEGARBDA/TRBDA)*XDDA3
      & +(OMEGARBDA*OMEGARBDA/TRBDA)*XDDA4)
      & +KRBDR*(XDDR1+(2*ZETARBDR*OMEGARBDR+1/TRBDR)*XDDR2
      & +(OMEGARBDR*OMEGARBDR+2*ZETARBDR*OMEGARBDR/TRBDR)*XDDR3
      & +(OMEGARBDR*OMEGARBDR/TRBDR)*XDDR4)
C
       TF_XCGD = U*COS(TF_THET)*COS(TF_PSI)
      & +V*(-COS(TF_PHI)*SIN(TF_PSI)
      & +SIN(TF_PHI)*SIN(TF_THET)*COS(TF_PSI))
      & +W*(SIN(TF_PHI)*SIN(TF_PSI)
      & +COS(TF_PHI)*SIN(TF_THET)*COS(TF_PSI))
C
       TF_YCGD = U*COS(TF_THET)*SIN(TF_PSI)
      & +V*(COS(TF_PHI)*COS(TF_PSI)
      & +SIN(TF_PHI)*SIN(TF_THET)*SIN(TF_PSI))
      & +W*(-SIN(TF_PHI)*COS(TF_PSI)
      & +COS(TF_PHI)*SIN(TF_THET)*SIN(TF_PSI))
C
       TF_ALTD = U*SIN(TF_THET)-V*SIN(TF_PHI)*COS(TF_THET)
      & -W*COS(TF_PHI)*COS(TF_THET)
```

```
C
C      CALCULATE THE FORCES, MOMENTS AND CASTLE OUTPUTS
C
       FAX = XMASS*(TF_VRWD*COS(TF_ALFA)*COS(TF_BETA)
     & -TF_VRW*TF_ALFD*COS(TF_BETA)*SIN(TF_ALFA)
     & -TF_VRW*TF_BETD*COS(TF_ALFA)*SIN(TF_BETA)
     & +G*SIN(TF_THET)-TF_RB*TF_VRW*SIN(TF_BETA)
     & +TF_QB*TF_VRW*SIN(TF_ALFA)*COS(TF_BETA))
C
       FAY = XMASS*(TF_VRWD*SIN(TF_BETA)
     & +TF_VRW*TF_BETD*COS(TF_BETA)
     & -G*SIN(TF_PHI)*COS(TF_THET)
     & -TF_PB*TF_VRW*SIN(TF_ALFA)*COS(TF_BETA)
     & +TF_RB*TF_VRW*COS(TF_ALFA)*COS(TF_BETA))
C
       FAZ = XMASS*(TF_VRWD*SIN(TF_ALFA)*COS(TF_BETA)
     & -TF_VRW*TF_BETD*SIN(TF_ALFA)*SIN(TF_BETA)
     & +TF_VRW*TF_ALFD*COS(TF_BETA)*COS(TF_ALFA)
     & -G*COS(TF_PHI)*COS(TF_THET)
     & -TF_QB*TF_VRW*COS(TF_ALFA)*COS(TF_BETA)
     & +TF_PB*TF_VRW*SIN(TF_BETA))
C
       TAL = -XIXZ*TF_PB*TF_QB+XIZZ*TF_QB*TF_RB
     & -XIYY*TF_QB*TF_RB+XIXX*TF_PBD-XIXZ*TF_RBD
C
       TAM = XIYY*TF_QBD+(XIXX-XIZZ)*TF_PB*TF_RB
     & +XIXZ*(TF_PB*TF_PB-TF_RB*TF_RB)
C
       TAN = XIZZ*TF_RBD-XIXZ*TF_PBD+XIXZ*TF_QB*TF_RB
     & +XIYY*TF_PB*TF_QB-XIXX*TF_PB*TF_QB
C
       NX =  FAX/XMASS/G
       NY =  FAY/XMASS/G
       NZ = -FAZ/XMASS/G
C
       VRW    =  TF_VRW
       ALFAR  =  TF_ALFA
       BETAR  =  TF_BETA
       PHIR   =  TF_PHI
       THETR  =  TF_THET
       PSIR   =  TF_PSI
       PB     =  TF_PB
       QB     =  TF_QB
       RB     =  TF_RB
       XCGLOC =  TF_XCG
       YCGLOC =  TF_YCG
       ALT    =  TF_ALT
       ZCGLOC = -TF_ALT
C
       ALFA  =  TF_ALFA * R2D
       BETA  =  TF_BETA * R2D
       PHI   =  TF_PHI  * R2D
       THET  =  TF_THET * R2D
       PSI   =  TF_PSI  * R2D
       PBDEG =  TF_PB   * R2D
       QBDEG =  TF_QB   * R2D
       RBDEG =  TF_RB   * R2D
C
```

```
      ALFD = TF_ALFD
      BETD = TF_BETD
      PHID = TF_PHID
      THED = TF_THED
      PSID = TF_PSID
      PBD  = TF_PBD
      QBD  = TF_QBD
      RBD  = TF_RBD
      VN   = TF_XCGD
      VE   = TF_YCGD
      ALTD = TF_ALTD
C
C      calculate the time derivatives of the states
C
      XDDT1 = S3LONG*XDT1+S2LONG*XDT2+S1LONG*XDT3+S0LONG*XDT4+DDTH
      XDDT2 = XDT1
      XDDT3 = XDT2
      XDDT4 = XDT3
C
      XDDE1 = S3LONG*XDE1+S2LONG*XDE2+S1LONG*XDE3+S0LONG*XDE4+DDE
      XDDE2 = XDE1
      XDDE3 = XDE2
      XDDE4 = XDE3
C
      XDDA1 = S3LD*XDA1+S2LD*XDA2+S1LD*XDA3+S0LD*XDA4+DDA
      XDDA2 = XDA1
      XDDA3 = XDA2
      XDDA4 = XDA3
C
      XDDR1 = S3LD*XDR1+S2LD*XDR2+S1LD*XDR3+S0LD*XDR4+DDR
      XDDR2 = XDR1
      XDDR3 = XDR2
      XDDR4 = XDR3
C
C      INTEGRATE STATES
C
      XDT1 = XDT1 + XDDT1 * DT
      XDT2 = XDT2 + XDDT2 * DT
      XDT3 = XDT3 + XDDT3 * DT
      XDT4 = XDT4 + XDDT4 * DT

      XDE1 = XDE1 + XDDE1 * DT
      XDE2 = XDE2 + XDDE2 * DT
      XDE3 = XDE3 + XDDE3 * DT
      XDE4 = XDE4 + XDDE4 * DT

      XDA1 = XDA1 + XDDA1 * DT
      XDA2 = XDA2 + XDDA2 * DT
      XDA3 = XDA3 + XDDA3 * DT
      XDA4 = XDA4 + XDDA4 * DT

      XDR1 = XDR1 + XDDR1 * DT
      XDR2 = XDR2 + XDDR2 * DT
      XDR3 = XDR3 + XDDR3 * DT
      XDR4 = XDR4 + XDDR4 * DT

      RETURN
      END
```

Vita

John Leonard was born on December 11, 1980, in Washington, D.C. to Barbara and Raymond Leonard. For the first ten years of his life, he traveled around the world with his parents and brother, Kevin, as his father was stationed at various U.S. Naval bases before coming back to the U.S. for good in 1990. After graduating from Chantilly High School in 1998, he attended Virginia Polytechnic Institute and State University for his undergraduate studies. In May of 2002, he graduated with a B.S. degree in Aerospace Engineering and a minor in Computer Science. His education continued at Virginia Tech as he worked closely with his advisor, Dr. Wayne Durham, and the Department of Aerospace and Ocean Engineering's full-motion manned flight simulator. In December of 2003, he graduated with a M.S. degree in Aerospace Engineering concentrating in Flight Dynamics and Control.