8-1-2009

Development of a novel method for autonomous navigation and landing of unmanned aerial vehicles

David Grymin

Development of a Novel Method for Autonomous Navigation and Landing of Unmanned Aerial Vehicles

by

David J. Grymin

Thesis submitted to the Faculty of the Department of Mechanical Engineering in the Kate Gleason College of Engineering at Rochester Institute of Technology in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

Rochester, New York

August, 2009

Keywords: Aerospace, Dubins Path, Sliding Mode Control, Unmanned Aerial Vehicle, Navigation, Control Systems

Permission to Reproduce Thesis

Development of a Novel Method for Autonomous Navigation and Landing of Unmanned Aerial Vehicles

Acknowledgements

First and foremost, I would like to thank all of the faculty and staff members of the Mechanical Engineering department at Rochester Institute of Technology for providing me with an exceptional educational experience. In particular, I would like to thank Dr. Agamemnon Crassidis. Without his guidance and patience, this work would not have been possible. I would also like to thank Dr. Mark Kempski and Dr. Wayne Walter for their assistance and advice over the past several years. Finally, I would like to thank my family members and friends for their support and encouragement.

Tommy: *"You know, a lot of people go to college for seven years."*
Richard: *"I know. They're called doctors."*

Tommy Boy. Dir. Peter Segal. Paramount Pictures, 1995.

Abstract

In this work, control techniques for the autonomous navigation and landing of an Unmanned Aerial Vehicle (UAV) are developed and compared. Controllers were developed and implemented on two different aircraft models: the Lockheed-Martin F-16 and AAI Corporation/Israel Aircraft Industries RQ-2 Pioneer. Due to the expense of modifying the pre-existing F-16 flight control system, the controller is implemented outside of the closed loop. Proportional-integral-derivative and proportional-integral controllers are developed for holding the aircraft at a desired velocity and altitude. The aircraft are approximated as Dubins vehicles constrained to travel on a two-dimensional surface for decreased simulation time. Using the simplified model two control techniques are developed and then compared. The first uses a proportional feedback controller based on the Rhumb-line that the aircraft is traveling along. The second control technique uses a trajectory determined from an algorithm using the Dubins path determination for the shortest travel distance between two points. A sliding mode controller is developed to guide the simplified model along the Dubins path trajectory. The advantage of the Dubins path trajectory is that it allows for a closed-form time estimate to reach the desired way-point. Comparison between the two navigation techniques using the simplified system shows a significant decrease in time to way-point for the Dubins curve trajectory controller. The Rhumb-line controller and a hybrid Rhumb-line/Dubins path controller are implemented on nonlinear models of both aircraft. Simulation of both controllers on the nonlinear model shows acceptable performance in guiding the aircraft between way-points. Also, the time to way-point for the nonlinear aircraft model guided by the hybrid controller is within 5% of the closed-form Dubins trajectory estimate. Autonomous landing is accomplished utilizing the path guidance and altitude controllers. The nonlinear simulated aircraft successfully followed the glideslope from way-point to runway.

Contents

List of Figures

List of Tables

Nomeclature

δa_{cmd}	aileron position command (radians)
δa_{max}	aileron hard stop limit
δlat	current position to waypoint latitude difference (radians)
$\delta long$	current position to waypoint longitude difference (radians)
ϕ_{0_d}	initial Dubins aircraft heading angle (Dubins reference frame)
ϕ_{0_f}	final Dubins aircraft heading angle (Dubins reference frame)
ϕ_{deg}	aircraft bank angle (degrees)
$\phi_{max}\ \psi$	commanded bank angle limit
ψ	heading angle, ground track reference frame
ψ_{cr}	corrected bearing angle from Dubins to ground track
ρ	turning radius of aircraft (ft)
θ_p	Rhumb-line for first target point, Dubins/Rhumb-line controller
$azerr$	azimuth error (degrees)
d	non-Dimensional Dubins reference frame distance (D/ρ)
D	Dubins reference frame distance (ft)
$dist$	Rhumb-line distance to waypoint (radians)
$ft2deg$	conversion factor, feet to degrees latitude ($2.742980561538962 \cdot 10^{-6}\ ft/deg$)
$\mathbf{H}$	angular momentum vector
lat_{ac}	current latitude of aircraft (radians)
lat_{out}	Dubins path output, latitude (degrees)
lat_p	latitude for first point, hybrid Dubins/Rhumb-line controller
lat_{wi}	latitude weighting factor
lat_{wp}	latitude of waypoint (radians)
$long_{ac}$	current longitude of aircraft (radians)
$long\_deg\_convert$	conversion factor, degrees latitude to longitude ($-1/0.8213$)

lon_{out}	Dubins path output, longitude (degrees)
$long_p$	longitude for first point, hybrid Dubins/Rhumb-line controller
lon_{wi}	longitude weighting factor
$long_{wp}$	longitude of waypoint (radians)
L_v	Left turn segment, Dubins set
p_n	Dubins path second segment length
q	horizontal scale component
q_n	Dubins path third segment length
$refcourse_{0-360}$	desired bearing angle at waypoint (degrees)
R_v	Right turn segment, Dubins set
S_v	Straight path segment, Dubins set
t_n	Dubins path first segment length
tol	tolerance for determining if path is directly East-West
$track_{0-360}$	aircraft ground track angle (degrees)
u	turning rate command input (sliding mode controller); Dubins turning rate
v	velocity command input (sliding mode controller)
V_T	aircraft true velocity (stability axis)
x	horizontal position, Dubins frame
x_f	Dubins path output in x-position coordinates
y	vertical position, Dubins frame
y_f	Dubins path output in y-position coordinates

Chapter 1

Introduction

1.1 Unmanned Aerial Vehicles

Unmanned aerial systems (UAS), which includes unmanned aerial vehicles (UAVs), are an area of great interest to the United States (US) government. A document released recently by the US Air Force (USAF) Intelligence, Surveillance and Reconnaissance division lays out the goals for UAS research and development for the year 2009 through 2047. UAS have seen significant growth in operations in the past six years. Figure 1.1 shows a 660% increase in use of two USAF UAVs, the MQ-1 Predator and MQ-9 Reaper, in the past six years. Also included in the USAF document are future plans for UAS research and development, which include a push towards decreasing the amount of pilot interaction. This movement affects the entire range of air vehicles: from micro-air vehicles to large transport aircraft [1].

The importance of decision making in UAVs was outlined by Clough in 2001 [2]. It was his belief that UAVs must be given a certain level of intelligence in relation to decision making in order to achieve further autonomy. A specific example would be rather than just following instructions given to the aircraft, the systems on board the aircraft should be able to decide whether or not those instructions will place it in a dangerous condition. Clough emphasized this point, quoting figures that stated 42% of UAV failures could be attributed to human error. Also important is the ability of UAV systems to relay information in a form that the human operator can understand. These improvements will lead to aircraft where the human role becomes closer to that of a supervisor instead

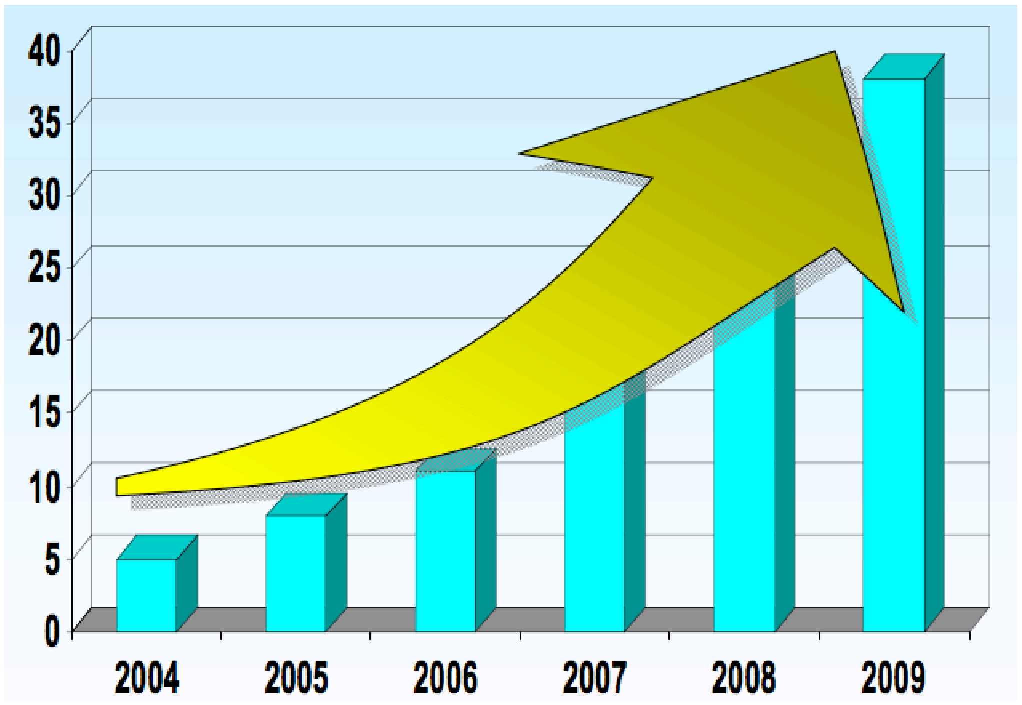

Figure 1.1: MQ-1 Predator and MQ-9 Reaper Combat Air Patrols [1].

of an operator.

The goal of this work is to develop and implement fully autonomous navigation and landing control on two aircraft: the Lockheed-Martin F-16 and the AAI/Israel Aircraft Industries RQ-2 Pioneer. Both will require different approaches as the F-16 is a high-performance fighter jet, originally designed to have a human pilot on-board, while the Pioneer was designed as a drone surveillance UAV. In the case of the F-16, the internal flight controls will not be modified, as validation required for changes to the internal flight code can be cost prohibitive. Redling provided support for this notion in 2001. In the case of a fighter jet that is inherently unstable and requires a control system in order for a pilot to operate the aircraft, Redling states that the control system should not be modified and autonomous control be completed through supplemental electronics [3].

Because the on-board control system and sensors will be used in controlling the F-16 aircraft, the proposed work intends to utilize real-time Global Positioning System (GPS) measurements in tracking location. Wang et. al. discussed the use of GPS in the navigation of pilotless aircraft. GPS is able to provide a higher level of accuracy over long distances and also faster response time compared to traditional positioning methods. Radio signals and telemetering are not as accurate as GPS and they also have a much smaller effective range. The authors strongly recommend against a strictly time-dependant trajectory, as external disturbances such as wind will push the aircraft off course. A recommended method is to determine a GPS-based course for the aircraft and determine the error signal between actual and desired position in real-time. The error signal should then be sent to a flight computer that decides how to return the aircraft to the course [4].

Issues with GPS resolution in relation to altitude were also found in the literature. Baird and Snyder stated that although GPS provides a high-resolution for absolute position, it is not able to account for changes in local relative altitude. Safe operation at very low altitudes can be compromised because of this. A mathematical model was developed to apply the Sandia Inertial Terrain-Aided Navigation algorithm to correlate inertial navigation and radar altimeter data. The range of accuracy that the authors were striving for was in the range of 1-2 feet [5]. This methodology was validated by Snyder et. al., who tested the accuracy of a GPS and inertial navigation system with supplemental radar altimeter data. Using a laser tracking system as a basis, the altitude error was reduced to a maximum of 2 feet from a previous maximum of 14 feet [6].

1.2 Dubin's Path Generation

The work of L.E. Dubins provided a basis for current theory regarding two-dimensionsal trajectory determination. His work provided a method for determining a series of segments, including curved and straight, that represented the shortest path from an initial point with a prescribed heading angle to a final way-point, also with a desired heading angle [7].

The method proposed by Dubins works well, however it is computationally intensive to determine and compare the resulting paths. Shkel and Lumelsky, in 2001, proposed a method of significantly decreasing the computation time for selecting the appropriate path set by using equivalence groups. Based on the initial and final heading angles and the orientation of the initial and final points, the desired trajectory can be classified and the path motions chosen from a predetermined set [8]. This method has become increasingly popular in two-dimensional robotics applications. Jeyaraman et. al applied this method to determine appropriate paths for a group of UAVs [9].

The use of the Dubins aircraft model is also quite common in the literature for simulation of an aircraft traveling along a two-dimensional surface. This model is also known as a Dubins car or unicycle model depending on the application that it is used in. For the purposes of this work, the model will be referred to as a Dubins simplified model or Dubins aircraft model. Equation 1.1 gives the general form of the Dubins simplified model, where v is velocity, ψ heading angle and u the vehicle's turning rate.

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} v\cos\psi \\ v\sin\psi \\ u \end{pmatrix} \tag{1.1}$$

The Dubins vehicle model frequently appears in solutions for time-optimal and optimal control scenarios. A set of formulas was derived by Tang et. al. for calculating feedback control steering. The optimal control law developed provided for guiding a Dubins vehicle along the shortest path problem [10]. Balluchi et. al. examined the stability and robustness of applying optimal control to a Dubins vehicle. The turning radius was constrained and the vehicle could only travel in the forward direction. The resulting controller was robust against noise in state measurements and also structurally stable with respect to modeling errors [11]. Chitsaz and LaValle expanded on the simplified Dubins vehicle model by also adding altitude. Components of paths were examined to allow an aircraft to reach the desired way-point, heading and altitude. These components of paths are intended to be used in a library of paths that a navigation controller can select from to piece together for a time-optimal trajectory [12].

Bhatia et. al. developed a system to implement the Dubins path on UAVs utilizing commercially available autopilot systems. The autopilot system selected provided for way-point navigation as well as altitude, airspeed and turning rate holds. Inputs for roll and airspeed were determined from the desired trajectory and a control law including saturation for physical limitations was developed. Software in-loop simulations were run to verify the control law tracking. The commercial autopilot systems provided acceptable tracking and time to reach way-point and the authors intend to implement the control system on the UAV and autopilot hardware for further performance evaluation [13].

1.3 Autonomous Landing

The subject of autonomous landing has also received attention in the literature. Yakimenko and Kaminer presented mathematical techniques for computing landing trajectory in real-time as well as tracking the desired trajectory with respect to uncertainties and disturbances. Using a simplified aircraft model, they provided for a solution to landing approach trajectory determination. The paper also stresses the need for optimal trajectories to be determined in real-time, as most other techniques compute trajectories offline. The technique developed determines the trajectory and then inverts the simplified aircraft model dynamics in order to determine the required bank angle and normal load factor projection [14]. The difficulty in applying this method to the proposed work would be in determining control surface commands on a far more complicated model in real-time.

Shaoyan and Zongji provided a theoretical model example for a landing control system applied to a combat air vehicle. The requirements imposed on their aircraft are to follow the landing track, precise longitudinal and lateral positioning, aircraft response to wind gusts and the ability to handle failures in the system. Robust control methods were used to address these issues, and a weighting matrix was introduced to compare an ideal model to inputs from the aircraft and make appropriate corrections from there. The speed and altitude commands were decoupled in their model, as a change in one was found to have little influence on another. The simulated aircraft was able to return to a constant state with a wind gust disturbance introduced [15].

The topic of autonomous landing was also explored by Pashilkar et. al. A neural-aided controller was chosen due to the significant advantages that such controllers have over classical gain controllers when failures are introduced into the system. Control laws for a high-performance fighter aircraft were designed using classical gain methods, a Baseline Trajectory Following Controller (BTFC) and a neural-aided BTFC. The BTFC controller was found to meet the landing requirements only for small stuck aileron deflections. The neural controller, however, was able to meet landing requirements for much larger stuck aileron and elevator deflections [16].

1.4 Overview and Motivation for Present Work

The original scope of this work involved developing and simulating an autonomous navigation and landing controller on a Lockheed-Martin F-16 aircraft. Due to the cost associated with modifying the internal flight control code already on the aircraft, a controller that is "outside-the-loop" was proposed. During the completion of this task, a project related to developing a way-point to way-point navigation system that could also estimate the time to reach the final destination on the RQ-2 Pioneer became available. It was decided that the RQ-2 Pioneer navigation controller should follow a similar approach as that developed for the F-16.

The majority of the work is completed in MATLAB and Simulink utilizing nonlinear aircraft models. Due to the computation time involved in simulating these models, a simplified model was proposed to decrease the time for navigation control system development and subsequent analysis.

The resulting control architecture will be implemented in MATLAB and Simulink utilizing non-linear aircraft models that are known to be accurate. A description of the requirements is below:

1. Develop an autonomous navigation and landing controller for a Lockheed-Martin F-16 combat aircraft.

2. Develop a method to simplify the RQ-2 Pioneer aircraft model for decreased computation time.

3. Develop a controller to guide aircraft from initial way-point to final desired way-point with required heading angle. An accurate estimation of time to way-point is also required.

4. Implement improved control technique on nonlinear RQ-2 Pioneer aircraft model.

Chapter 2

Aircraft Model

2.1 Nonlinear Model

2.1.1 Rigid Body Equations

A nonlinear model was developed for three-dimensional simulations of the aircraft. The model will be used in control system development and also to verify the implementation of the control system. The model was built utilizing MATLAB and Simulink. Table 2.1.1 displays a list and definition of variables used in this section. Newton's second law is used to derive the rigid body equations of motion stating that the time rate of change of the momentum of the body is equal to the summation of all the external forces and the time rate change of angular momentum is equal to the summation of the external moments acting on the body. Newton's second law is represented by the vector equations below [17].

$$\begin{aligned} \textstyle\sum \vec{\mathbf{F}} &= \tfrac{d}{dt}(m\vec{\mathbf{v}}) \\ \textstyle\sum \vec{\mathbf{M}} &= \tfrac{d}{dt}\vec{\mathbf{H}} \end{aligned} \tag{2.1}$$

In equation 2.1, $\vec{\mathbf{F}}$ is the net vector force on the rigid body, m the mass, $\vec{\mathbf{v}}$ the velocity, $\vec{\mathbf{M}}$ the net moment vector and $\vec{\mathbf{H}}$ the angular momentum, referenced to the defined coordinate frame. The scalar form of the above equations is given below.

Table 2.1: Reference of forces, moments and rates for fixed aircraft body axis system.

	Roll Axis x_b	Pitch Axis y_b	Yaw Axis z_b
Aerodynamic force components	X	Y	Z
Aerodynamic moment components	L	M	N
Angular rates	p	q	r
Velocity components	u	v	w
Moment of inertia (about axis)	I_{xx}	I_{yy}	I_{zz}
Products of inertia	I_{yz}	I_{xz}	I_{xy}

$$\begin{aligned} &F_x = \tfrac{d}{dt}(mu) \quad F_y = \tfrac{d}{dt}(mv) \quad F_z = \tfrac{d}{dt}(mw) \\ &L = \tfrac{d}{dt}H_x \qquad M = \tfrac{d}{dt}H_y \qquad N = \tfrac{d}{dt}H_z \end{aligned} \tag{2.2}$$

The force components along the x, y and z axes are represented by F_x, F_y and F_z while the velocity components are represented by u, v and w, respectively. The moment components along the x, y and z axes are represented by L, M and N while the moment of momentum components are represented by H_x, H_y and H_z respectively.

Considering the individual differential mass elements of the aircraft, Newton's second law can be written as below where $\vec{\mathbf{v}}_c$ is the velocity of the aircraft's center of mass, $d\vec{\mathbf{r}}/dt$ is the relative velocity of an element to the center of mass, and δm is an element of mass.

$$\sum \delta\vec{\mathbf{F}} = \vec{\mathbf{F}} = \frac{d}{dt}\sum\left(\vec{\mathbf{v}}_c + \frac{d\vec{\mathbf{r}}}{dt}\right)\delta m \tag{2.3}$$

Assuming that mass is constant and since $\vec{\mathbf{r}}$ is measured from the center of mass, the sum of all the $\vec{\mathbf{r}}\delta m$ elements is equal to zero and Equation 2.3 can be reduced to the form in Equation 2.4.

$$\vec{\mathbf{F}} = m\frac{d\vec{\mathbf{v}}_c}{dt} \tag{2.4}$$

The moment equation referring to moving center of mass for a differential element can be derived in the same fashion and written as in Equation 2.5.

$$\delta\vec{\mathbf{M}} = \frac{d}{dt}\delta\vec{\mathbf{H}} = \frac{d}{dt}\left(\vec{\mathbf{r}} \times \vec{\mathbf{v}}\right)\delta m \tag{2.5}$$

Where $\vec{\mathbf{v}}$ in Equation 2.5 is equal to the expression in Equation 2.6.

$$\vec{\mathbf{v}} = \vec{\mathbf{v}}_c + \vec{\omega} \times \vec{\mathbf{r}} \tag{2.6}$$

Substituting Equation 2.6 into Equation 2.5 and assuming $\vec{\mathbf{v}}_c$ to be constant yields the moment equation given in Equation 2.7.

$$\vec{\mathbf{H}} = \sum \vec{\mathbf{r}}\delta m \times \vec{\mathbf{v}}_c + \sum \left[\vec{\mathbf{r}} \times \left(\vec{\omega} \times \vec{\mathbf{r}}\right)\right]\delta m \tag{2.7}$$

Equation 2.7 can also be written as below, where p, q and r are the angular rates about the x, y and z axes, respectively.

$$\vec{\mathbf{H}} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{2.8}$$

The matrix in Equation 2.8 is the inertia tensor for the aircraft, with the components of the matrix as defined in Equation 2.9.

$$\begin{aligned} I_{xx} &= \textstyle\int\int\int \left(y^2 + z^2\right)\delta m & I_{xy} &= \textstyle\int\int\int xy\;\delta m \\ I_{yy} &= \textstyle\int\int\int \left(x^2 + z^2\right)\delta m & I_{xz} &= \textstyle\int\int\int xz\;\delta m \\ I_{zz} &= \textstyle\int\int\int \left(x^2 + y^2\right)\delta m & I_{yz} &= \textstyle\int\int\int yz\;\delta m \end{aligned} \tag{2.9}$$

These terms represent the mass moments of inertia about each body axis and the products of inertia. In a fixed reference frame, if the aircraft rotates these terms will vary. The axis system is therefore fixed to the aircraft to assume the mass moments of inertia remain constant for a given mass. The resulting equations of motion are given in Equations 2.10 and 2.11.

$$\vec{\mathbf{F}} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} m\,(\dot{u} + qw - rv) \\ m\,(\dot{v} + ru - pw) \\ m\,(\dot{w} + pv - qu) \end{bmatrix} = \vec{\mathbf{W}}_{ref} + \vec{\mathbf{F}}_{aerodynamic} + \vec{\mathbf{F}}_{thrust} \tag{2.10}$$

$$\vec{\mathbf{M}} = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \tag{2.11}$$
$$\begin{pmatrix} qr(I_{yy} - I_{zz}) + (q^2 - r^2)I_{xy} - prI_{xy} + pqI_{xz} \\ pr(I_{zz} - I_{xx}) + (r^2 - p^2)I_{xz} - pqI_{yz} + qrI_{xy} \\ pq(I_{xx} - I_{yy}) + (p^2 - q^2)I_{xy} - qrI_{xz} + prI_{yz} \end{pmatrix} + \vec{\mathbf{M}}_{external}$$

These equations of motion are applicable to both the F16 and RQ-2 Pioneer aircraft. The aircraft mass and inertia tensor will vary between the two models. Aerodynamic, gravitational and propulsive forces make up the components of force and moment that act upon the aircraft. The factors consisting of the components of the the summation of forces and moments are covered in Section 2.1.3.

2.1.2 Euler Kinematic Equations

In order to track the orientation of the aircraft to the Earth fixed-axis system, a standard method for angular transformations must be used. The Euler angle rotation convention is a standard sequence of applying rotations for aerospace applications. The rotations are applied about the primary, secondary and tertiary axes (x_b, y_b, z_b) in that order.

Equation 2.12 defines the transformation matrix for each of the rotations. Equation 2.13 shows how the rotation matrices are applied in order to transform rotations from the Earth to vehicle axis system and vice-versa, respectively.

$$R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}$$

$$R_2 = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.12}$$

$$R_3 = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} \mathbf{T}_{E\to V} &= R_1 \cdot R_2 \cdot R_3 \\ \mathbf{T}_{V\to E} &= R_3 \cdot R_2 \cdot R_1 \end{aligned} \tag{2.13}$$

The transformation matrices for converting Earth fixed axis rotation rates to body axis rotation rates (and vice-versa) are given by Equations 2.14 and 2.15. Note: in Equation 2.15, singularities in the matrix exist as pitch angle θ approaches 90 degrees. To avoid singularities, a quaternion transformation is typically used, however it is not necessary for this application as pitch angle should not approach 90 degrees while navigating from way-point to way-point while holding altitude near constant.

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} \tag{2.14}$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{2.15}$$

2.1.3 Stability Axis Coordinate System

The nonlinear equations of motion for the aircraft are based in the stability axis system. The stability axis is based around the aircraft's true velocity, V_T, which is the magnitude of the body axis velocity components. The angle of attack, α, is defined as the pitch angle of the aircraft relative to the oncoming wind. The sideslip angle, β, is also measured relative to the oncoming wind. Figure 2.1 provides an illustration of α and β in relation to the oncoming wind.

Equations 2.16 and 2.17 provide transformations from the body to stability axis system and vice-versa, respectively.

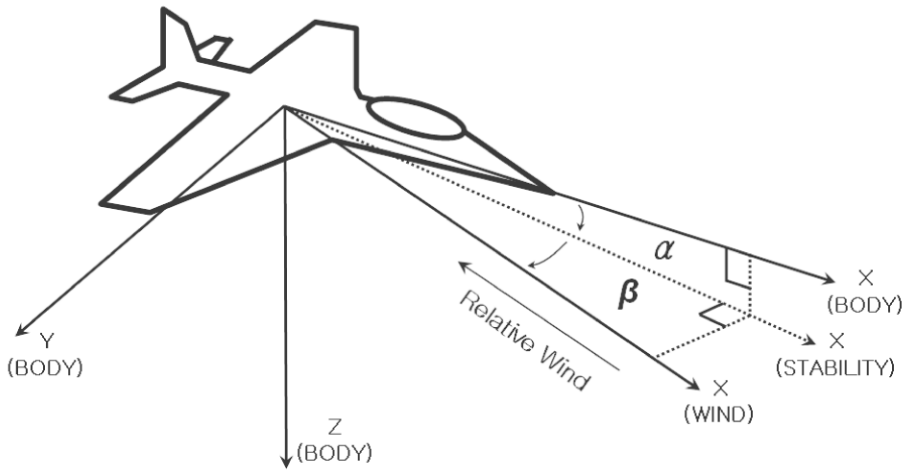

Figure 2.1: Illustration of stability axis and aircraft body axis [18].

$$\begin{aligned} \alpha &= \arctan \tfrac{w}{u} \\ \beta &= \arcsin \tfrac{v}{V_T} \\ V_T &= \sqrt{u^2 + v^2 + w^2} \end{aligned} \tag{2.16}$$

$$\begin{aligned} u &= V_T \cos\alpha \cos\beta \\ v &= V_T \sin\beta \\ w &= V_T \sin\alpha \cos\beta \end{aligned} \tag{2.17}$$

Equation 2.18 gives the stability force equations, where DOM, YOM and LOM are defined by Equation 2.19. The variables D, T, Y, and L in Equation 2.19 represent the drag, thrust, side and lift forces acting on the aircraft. The drag, side and lift forces are assumed to act along the stability axes while thrust acts along the aircraft body axis, x_b.

$$\begin{aligned}
&\dot{\alpha} = q - (p\cos\alpha + r\sin\alpha)\tan\beta - \tfrac{LOM}{V_T\cos\beta} + \\
&\quad \tfrac{g}{V_T\cos\beta}\left(\cos\theta\cos\phi\cos\alpha + \sin\theta\sin\alpha\right) \\
&\dot{\beta} = p\sin\alpha - r\cos\alpha + \tfrac{1}{V_T}\left(YOM\cos\beta + DOM\sin\beta\right) + \\
&\quad \tfrac{g}{V_T}\left(\cos\theta\sin\phi\cos\beta + \sin\theta\sin\beta\cos\alpha - \cos\theta\cos\phi\sin\beta\sin\alpha\right) \\
&\dot{V_T} = YOM\sin\beta - DOM\cos\beta + \\
&\quad g\left[(\cos\theta\cos\phi\sin\alpha - \sin\theta\cos\alpha)\cos\beta + \cos\theta\sin\phi\sin\beta\right]
\end{aligned} \tag{2.18}$$

$$DOM = \tfrac{D - T\cos\alpha}{m};\quad YOM = \tfrac{Y}{m};\quad LOM = \tfrac{L + T\sin\alpha}{m} \tag{2.19}$$

2.1.4 Stability Derivative Equations

As stated in the previous section, the external forces and moments acting on the aircraft are composed of aerodynamic, gravitational and thrust contributions. Equation 2.20 gives the components of the aircraft weight force, where θ is pitch angle and ϕ is bank angle.

$$\mathbf{W} = mg\begin{bmatrix} \sin\theta \\ \sin\phi\cos\theta \\ \cos\phi\cos\theta \end{bmatrix} \tag{2.20}$$

Equations 2.21 and 2.22 give the force and moment vectors, where $\bar{q}$ is dynamic pressure defined in equation 2.23, S is the wing reference area, $\bar{c}$ the wing mean aerodynamic chord, and b the wing span. The force and moment coefficients, denoted by a C with a subscript, are determined by the summation of contributions to each term.

$$\mathbf{F}_{aerodynamic} = \begin{bmatrix} Drag \\ Side \\ Lift \end{bmatrix} = \begin{bmatrix} \bar{q}SC_D \\ \bar{q}SC_Y \\ \bar{q}SC_L \end{bmatrix} \tag{2.21}$$

$$\mathbf{M}_{external} = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} \bar{q}SbC_l \\ \bar{q}S\bar{c}C_m \\ \bar{q}SbC_n \end{bmatrix} \tag{2.22}$$

$$\bar{q} = \frac{1}{2}\rho(V_T)^2 \tag{2.23}$$

Equations 2.24 gives the components of the force and moment coefficients. The coefficients are defined by nonlinear lookup tables. Equations 2.25 and 2.26 represent linear approximations of the nonlinear table lookup models. The variables δa, δe, δf, and δr are the control surface deflections corresponding to the aileron, elevator, flap and rudder, respectively.

$$\begin{aligned}
&\text{Longitudinal Axis:}\\
&C_D = f\left(\alpha, \delta e\right)\\
&C_L = f\left(\alpha, \dot{\alpha}, q_b, \delta e, \delta f, V_T\right)\\
&C_m = f\left(\alpha, \dot{\alpha}, q_b, \delta e, V_T\right)\\
&\\
&\text{Lateral-Directional Axis:}\\
&C_Y = f\left(\beta, \delta r\right)\\
&C_l = f\left(\alpha, \beta, p_b, r_b, \delta a, \delta r, V_T\right)\\
&C_n = f\left(\alpha, \beta, p_b, r_b, \delta a, \delta r, V_T\right)
\end{aligned} \tag{2.24}$$

$$\begin{aligned}
C_D &= C_{D_0} + C_{D_\alpha}\alpha + \tfrac{c}{2V_T}\left(C_{D_q}q + C_{D_{\dot{\alpha}}}\dot{\alpha}\right) + C_{D_{\delta e}}\delta e + C_{D_{\delta f}}\delta f\\
C_Y &= C_{Y_0} + C_{Y_\beta}\beta + \tfrac{b}{2V_T}\left(C_{Y_p}p + C_{Y_r}r\right) + C_{Y_{\delta a}}\delta a + C_{Y_{\delta r}}\delta r\\
C_L &= C_{L_0} + C_{L_\alpha}\alpha + \tfrac{c}{2V_T}\left(C_{L_q}q + C_{L_{\dot{\alpha}}}\dot{\alpha}\right) + C_{L_{\delta e}}\delta e + C_{L_{\delta f}}\delta f
\end{aligned} \tag{2.25}$$

$$\begin{aligned}
C_l &= C_{l_0} + C_{l_\beta}\beta + \tfrac{b}{2V_T}\left(C_{l_p}p + C_{l_r}r\right) + C_{l_{\delta a}}\delta a + C_{l_{\delta r}}\delta r\\
C_m &= C_{m_0} + C_{m_\alpha}\alpha + \tfrac{c}{2V_T}\left(C_{m_q}q + C_{m_{\dot{\alpha}}}\dot{\alpha}\right) + C_{m_{\delta e}}\delta e + C_{m_{\delta f}}\delta f\\
C_n &= C_{n_0} + C_{n_\beta}\beta + \tfrac{b}{2V_T}\left(C_{n_p}p + C_{n_r}r\right) + C_{n_{\delta a}}\delta a + C_{n_{\delta r}}\delta r
\end{aligned} \tag{2.26}$$

Figure B.7 in (Appendix B.2) shows the Simulink block diagram of the general nonlinear aircraft model.

2.2 Aircraft Trim Conditions

Trim is defined as the lack of rotation about the aircraft's center of gravity when not performing flight maneuvers [17]. The trim conditions for each aircraft are given in this section. In each case, a MATLAB® script was generated in order to trim the aircraft. This was done by running the nonlinear Simulink® model of each aircraft iteratively and varying flight parameters until the trim index (shown in Equation 2.27) was within a specified tolerance.

$$\text{trim\_index} = \sqrt{\dot{V_T}^2 + \dot{\alpha}^2 + \dot{\beta}^2 + \dot{p}^2 + \dot{q}^2 + \dot{r}^2} \tag{2.27}$$

2.2.1 F-16 Trim Specifications

The F-16 was trimmed at an altitude of $2500ft$ with a desired airspeed of 160 knots. The resulting values for the trim condition are included in Table 2.2.

Table 2.2: F-16 trim conditions

Altitude (ft)	2.4500000E+03
Mach	2.5273570E-01
V_{cas} (ft/s)	1.6006514E+02
V_T (ft/s)	2.7977589E+02
α (deg)	1.0474635E+01
β (deg)	−9.5515703E-31
ϕ (deg)	0
θ (deg)	1.0474635E+01
ψ (deg)	−1.4000000E+02

2.2.2 RQ-2 Pioneer Trim Conditions

In the case of the Pioneer aircraft, two flight conditions for navigation to the desired way-point were defined. The first was quickest time to way-point and the second minimum fuel consumption. For the minimum fuel condition, the lift-to-drag ratio defined as L/D

was found in order to determine most efficient airspeed to operate at. The nonlinear model of the RQ-2 Pioneer in Simulink® was used to determine the lift and drag forces for the trimmed aircraft at a variety of airspeeds. The data points were then fit with a polynomial function using the *polyfit* m-function in MATLAB®. Equation 2.28 gives the polynomial fit equation shown in Figure 2.2. The derivative of the polynomial fit was then taken (as shown in Equation 2.29).

$$\frac{L}{D} = -1.21404\text{E-}05V_T^4 + 6.16361\text{E-}03V_T^3 - 1.117114V_T^2 + 98.6808V_T - 3100.36 \quad (2.28)$$

The roots of Equation 2.29 give the V_T values where local minima and maxima of L/D occur. The roots are shown in Equation 2.30. Ignoring the roots with imaginary components, V_T is found to be 115.294 ft/s in order to operate at maximum L/D.

$$\frac{d}{dV_T}\left(\frac{L}{D}\right) = -4.85614\text{E-}05V_T^3 + 1.84908\text{E-}02V_T^2 - 2.34228V_T + 98.6808 \quad (2.29)$$

$$V_T = \begin{pmatrix} 132.739\text{E+}02 + 2.35758i \\ 132.739\text{E+}02 - 2.35758i \\ 115.294 \end{pmatrix} ft/s \quad (2.30)$$

The RQ-2 Pioneer model was then trimmed to a velocity of 115.294 ft/s. The corresponding aircraft state values and control surface deflections are given in Table 2.3. The trim conditions for minimum time to way-point, corresponding to a trimmed V_T of 167.856 ft/s are also included in Table 2.3. Equation 2.27 was used for determining trim conditions for both RQ-2 Pioneer cases.

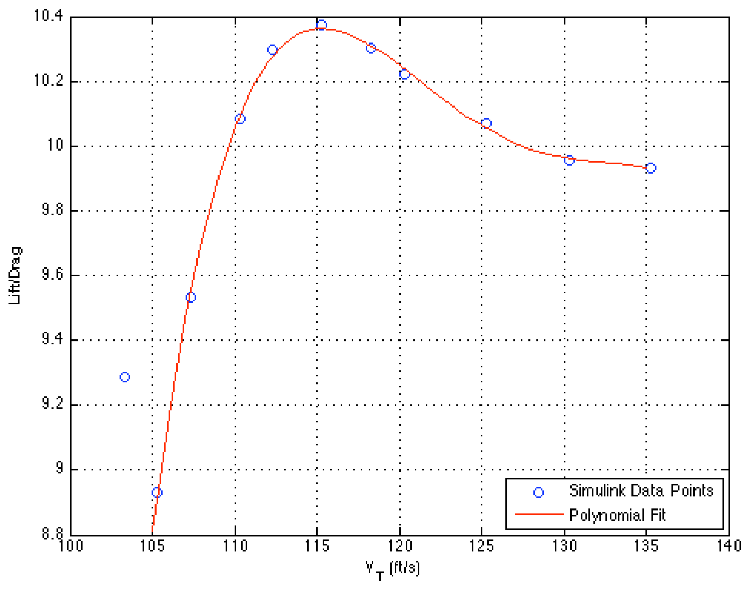

Figure 2.2: Lift/Drag versus V_T curve for RQ-2 Pioneer.

Table 2.3: RQ-2 Pioneer trim conditions for fuel conservation and minimum time to way-point.

Parameter	Min. Fuel	Min. Time
Altitude (ft)	5000	5000
V_T (ft/s)	115.294	167.856
α (deg)	7.0000	0.16111
β (deg)	0	0
ϕ (deg)	0	0
θ (deg)	7.0000	0.16111
ψ (deg)	70	70
δe (deg)	-2.0433	11.803
δa (deg)	0	0
δr (deg)	0	0

2.2.3 F-16 Control Law

Outer Loop Control Law

As stated previously, the F-16 is a fly-by-wire aircraft with a pre-existing flight control system. The control system provides stability augmentation, as the aircraft is inherently unstable. Equation 2.31 shows the longitudinal axis **A** matrix for the state-space model of the F-16 aircraft linearized at the trim conditions specified in Section 2.2.1. Equation 2.33 gives the **A** matrix for the lateral-directional state-space model linearized at the same conditions. The state vectors corresponding to these matrices are given in Equations 2.32 and 2.34.

$$A_{lon} = \begin{bmatrix} -4.7977\text{E-}02 & -5.3805\text{E-}02 & -4.0941\text{E-}03 & -5.6157\text{E-}01 \\ -4.5370\text{E-}02 & -4.7251\text{E-}01 & 9.9521\text{E-}01 & 0 \\ 2.2923\text{E-}03 & 2.3891\text{E-}01 & -5.1226\text{E-}01 & 0 \\ 0 & 0 & 1.0000\text{E+}00 & 0 \end{bmatrix} \tag{2.31}$$

$$\mathbf{x}_{lon} = \begin{bmatrix} V_T & \alpha & q & \theta \end{bmatrix}^T \tag{2.32}$$

$$A_{lat} = \begin{bmatrix} -1.7634\text{E-}01 & 1.8247\text{E-}01 & -9.7912\text{E-}01 & 1.1309\text{E-}01 \\ -1.7866\text{E+}01 & -1.8552\text{E+}00 & 9.1787\text{E-}02 & 0 \\ 1.4047\text{E+}00 & -4.1356\text{E-}02 & -3.1466\text{E-}01 & 0 \\ 0 & 1.0000\text{E+}00 & 1.8488\text{E-}01 & 0 \end{bmatrix} \tag{2.33}$$

$$\mathbf{x}_{lat} = \begin{bmatrix} \beta & p & r & \phi \end{bmatrix}^T \tag{2.34}$$

The eigenvalues of these matrices provide information about the stability of the linearized aircraft at the trim condition. Tables 2.4 and 2.5 give the eigenvalues, damping ratios and natural frequencies for Equations 2.31 and 2.33, respectively.

Note that the negative eigenvalues indicate poles in the Left Half Plane (LHP), while positive eigenvalues indicate a pole is in the Right Half Plane (RHP). A system is considered stable if its poles are location in the LHP. The linearized F-16 aircraft at trim

Table 2.4: Eigenvalues, damping and natural frequency for longitudinal axis F-16 state-space model.

Eigenvalue	Damping	Freq. (rad/s)
1.53E-01	−1.00	1.53E-01
−9.85E-02+1.63E-01i	5.18E-01	1.90E-01
−9.85E-02−1.63E-01i	5.18E-01	1.90E-01
−9.89E-01	1.00E+00	9.89E-01

Table 2.5: Eigenvalues, damping and natural frequency for lateral-directional axis F-16 state-space model.

Eigenvalue	Damping	Freq. (rad/s)
−9.40E-02	1.00E+00	9.40E-02
−4.28E-01+2.00E+00i	2.09E-01	2.05E+00
−4.28E-01−2.00E+00i	2.09E-01	2.05E+00
−1.40E+00	1.00E+00	1.40E+00

condition has a pole in the longitudinal axis system in the RHP. This pole indicates a pitching moment instability. Looking at the A_{lon} matrix, it can be seen that a positive angle-of-attack disturbance will produce a positive rotation q about the y-axis, whereas a stable aircraft would return to the trim condition after such a disturbance. The pre-existing F-16 flight control system provides stability augmentation in this case and for other cases of instability through the aircraft's flight envelope. In order to develop the outer-loop navigation controller, the control system need to be modeled. A block diagram of illustrating the outer-loop controller interface with the F-16 control system is shown in Appendix B.1.

Input Conditioning

In the case of the F16 aircraft modeled, inputs to the control stick must be scaled in order to be passed into the control system. A dead zone breakout and nonlinear gain is built into the internal flight control software and must be accounted for, with the goal of the controller to be completely outside the pre-existing control loop.

The pitch and roll inputs both have a dead-zone and nonlinear gain, while the yaw input has a dead-zone and constant gain. The solution to this issue is to invert the lookup table and then add a constant to the input signal cancelling out the dead-zone effect. The three input scalars were created in Simulink® so that they could easily be placed between the output of the control system and the input to the aircraft model in later work. Block diagrams of these three input scalers are shown in Appendix B.1.2. Figure 2.3 shows the signal that the control system sees with and without the scaled input.

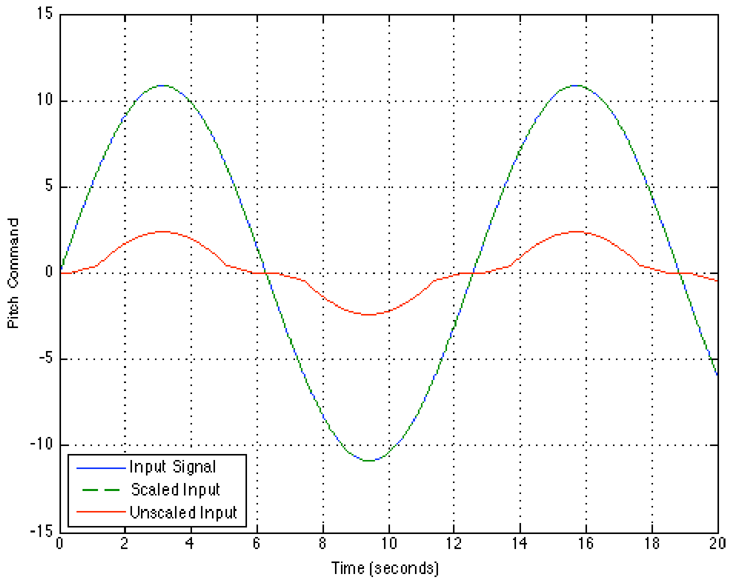

Figure 2.3: Comparison of scaled and unscaled sinusoidal pitch input command.

2.2.4 RQ-2 Pioneer Control Law

The stability of the RQ-2 Pioneer was investigated in a similar method to the F-16 aircraft. The aircraft was linearized about trim conditions and state-space models for the longitudinal and lateral-directional axes were determined. The **A** matrix for each of the axes are shown in Equations 2.35 and 2.36.

$$A_{lon} = \begin{bmatrix} -5.3739\text{E-}02 & 2.9495\text{E-}01 & 0 & -5.6157\text{E-}01 \\ -2.7409\text{E-}01 & -1.2839 & 9.8270\text{E-}01 & 0 \\ 1.9165\text{E-}16 & -5.0967\text{E+}01 & -3.1758\text{E+}00 & 0 \\ 0 & 0 & 1.0000\text{E+}00 & 0 \end{bmatrix} \tag{2.35}$$

$$A_{lat} = \begin{bmatrix} -1.8601\text{E-}01 & 1.2187\text{E-}01 & -9.9255\text{E-}01 & 2.7699\text{E-}01 \\ -7.7377\text{E+}00 & -6.5874\text{E+}00 & 4.1138\text{E+}00 & 0 \\ 2.2134\text{E+}01 & -2.9371\text{E-}01 & -1.4934\text{E+}00 & 0 \\ 0 & 1.0000\text{E+}00 & 1.2278\text{E-}01 & 0 \end{bmatrix} \tag{2.36}$$

Examining the eigenvalues of these matrices provides information regarding the stability of the linearized RQ-2 Pioneer at the trim conditions. The eigenvalues for the longitudinal and lateral-directional axes are given in Tables 2.6 and 2.7.

Table 2.6: Eigenvalues, damping and natural frequency for longitudinal axis RQ-2 Pioneer state-space model.

Eigenvalue	Damping	Freq. (rad/s)
$-2.33\text{E-}02+3.80\text{E-}01i$	6.11E-02	3.81E-01
$-2.33\text{E-}02-3.80\text{E-}01i$	6.11E-02	3.81E-01
$-2.23 + 7.01i$	3.04E-01	7.36
$-2.23 - 7.01i$	3.04E-01	7.36

As indicated by the eigenvalues, the RQ-2 Pioneer is stable at trim as indicated by the eigenvalues lying in the LHP with the exception of the spiral roll motion, dependent primarily on sideslip, β, and yaw rate, r. A controller was implemented to improve the aircraft's handling qualities, primarily to increase the long-period longitudinal damping and Dutch roll lateral-directional damping. Pole placement was used to determine gains that provided improved handling qualities. Figures 2.4 and 2.5 show the response of both the open-loop aircraft and closed-loop nonlinear models to an aileron doublet maneuver. This involves commanding an aileron input and then returning to the original position.

Table 2.7: Eigenvalues, damping and natural frequency for lateral-directional axis RQ-2 Pioneer state-space model.

Eigenvalue	Damping	Freq. (rad/s)
1.83E-01	-1.00	1.83E-01
$-1.08 + 4.72i$	2.23E-01	4.84
$-1.08 - 4.72i$	2.23E-01	4.84
-6.29	1.00	6.29

As can be seen in the figures, the controller provides improved handling for the aircraft. Note the plot of p, shown in the fourth subplot of Figure 2.4. The roll rate for the closed-loop controller tracks to a steady value while the open-loop model exhibits poor damping. After the maneuver is completed the open-loop model still continues to roll, while the closed-loop aircraft does not. Also note the bank angle response, ϕ, shown in the first subplot of Figure 2.5. The open-loop model turns at a quick rate than the closed loop, however after the maneuver has been completed bank angle continues to increase. Additional figures showing the closed-loop aircraft responses for a step-input to the elevator and rudder doublet are provided in Section C.

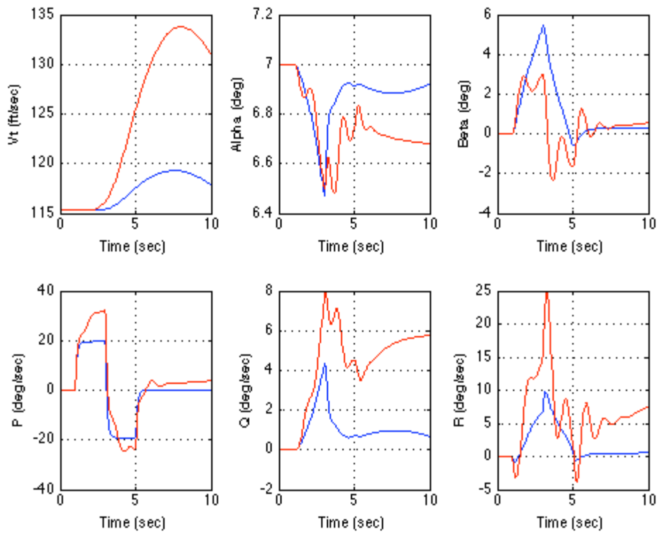

Figure 2.4: Orientation and rotation rates for δa aileron doublet. Closed-loop is shown in blue, open-loop in red.

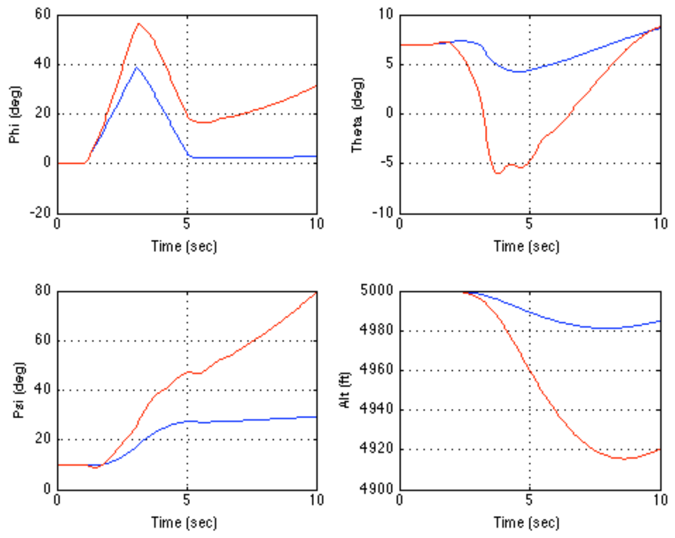

Figure 2.5: Orientation and altitude for δa aileron doublet. Closed-loop is shown in blue, open-loop in red.

2.3 NASA Dryden Wind Gust Model

A time-based wind gust similar to the NASA Dryden wind gust model was implemented to simulate head and tailwinds for the nonlinear aircraft model. The magnitude of the wind gust is defined in Equation 2.37. Three parameters are used to calculate the wind gust. U_{wind} is the end magnitude of the gust, in ft/s. U_wind_d is the time to reach peak from when the wind gust begins. U_wind_Td is a time delay for the wind gust to be applied. Figure 2.6 shows a sample output of a wind gust. U_{wind} was defined as 10 ft/s, U_wind_d was 2 s, and U_wind_Td was 0.5 s. The wind gust was modeled in Simulink® and is applied as a head or tail wind along the x_b body axis on the nonlinear aircraft model.

$$\begin{array}{ll} \frac{|U_{wind}|}{2} \cdot \left[1 - \frac{\cos \pi t}{U\_wind\_d}\right] & \text{for} \quad U \leq U\_wind\_d \\ & \\ |U_{wind}| & for \quad U > U\_wind\_d \end{array} \tag{2.37}$$

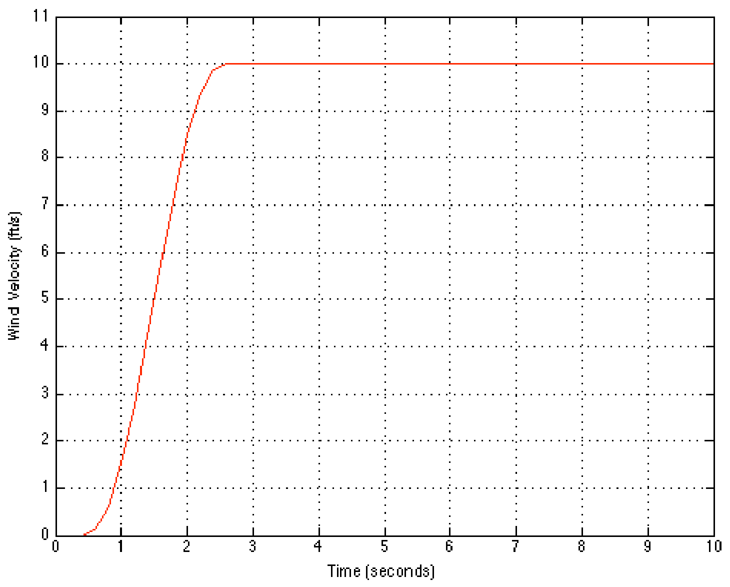

Figure 2.6: Plot of wind-gust along x_b axis over time.

2.4 Dubins Vehicle Simplified Model

The non-holonomic car, or here-after referred to as the "Dubins vehicle", is a simplified model to represent a vehicle constrained to traveling on a two-dimensional surface. The inputs to the system are the vehicle's velocity and its turning rate. The general form of a Dubins vehicle is given in Equation 2.38. The inputs to the model equation are u, the turning rate of the vehicle, and v, the velocity. Because the aircraft is restricted to a two-dimensional surface through the altitude hold, it can be approximated as a Dubins vehicle. Note that the ϕ specified in Equation 2.38 is not the same as the aircraft's bearing angle. It corresponds to the $X-Y$ plane heading angle and to avoid confusion it is not denoted as ψ.

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ \phi_d \end{bmatrix} = \begin{bmatrix} v\cos\phi_d \\ v\sin\phi_d \\ u \end{bmatrix} \tag{2.38}$$

The Dubins vehicle model used varies slightly in that a gain corresponding to the turning rate caused by a deflection of the aileron was used. Data was generated by running multiple simulations of the aircraft encompassing a range of turning rates and then used to calculate the gain. A cost function, given in Equation 2.39, was developed in order to minimize the error in latitude and longitude over the entire trajectory. Latitude and longitude errors can be independently weighted. The cost function was implemented in MATLAB utilizing the FMINSEARCH function in order to minimize k. The Dubins aircraft model is implemented in Simulink®. The block diagram for the model is given in Appendix B.5.

$$k = \int_0^{t_f} \left(\sum_{i=1}^{n} [\Delta long_i \cdot w_{long}] + \sum_{i=1}^{n} [\Delta lat_i \cdot w_{lat}] \right) dt \tag{2.39}$$

Chapter 3

Navigation Control

In this section, the development and implementation of the autonomous landing controller as well as two different navigation control schemes are described. The Rhumb-line navigation controller (first navigation scheme) was first implemented on the nonlinear F16 aircraft model. This controller was then applied to the RQ-2 Pioneer. The Dubins path navigation controller (second navigation scheme) is an improvement on the Rhumb-line controller in order to decrease time to way-point and also provide a closed-form prediction of the time to way-point.

3.1 Rhumb-Line Navigation

3.1.1 Rhumb-Lines

Rhumb-lines are lines of constant bearing, also known as loxodromes. When projected onto a sphere, they spiral from one pole to another. On a Mercator projection map, a Rhumb-line will appear as a series of parallel lines crossing the globe. Figure 3.1 shows Rhumb-lines in both spherical and Mercator projections.

Rhumb-lines were chosen to guide the aircraft because of the requirement of autonomous landing. The aircraft can be guided to the vortac point (where the landing procedure begins) by choosing a Rhumb-line located along the path of the runway.

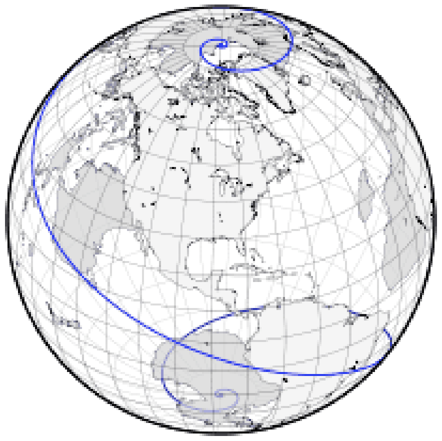
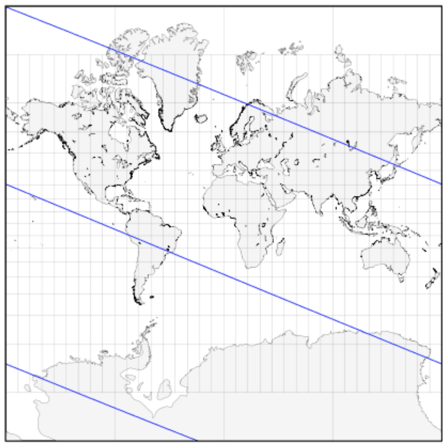

Figure 3.1: Loxodrome projected onto sphere and Mercator map projection of Rhumb-line [19].

3.1.2 Rhumb-Line Controller

The Rhumb-line controller is a proportional feedback controller that commands an aileron deflection based on the course error of the aircraft. The following shows the derivation of course error, computed from the current position and bearing angle and the desired way-point position and bearing angle.

Equation 3.1 defines q^2, the square of the horizontal scale component. Note that when the latitude of the way-point is within a given tolerance from the latitude of the current position, the horizontal scale component is defined differently. The quantity $\delta\phi$ is defined in Equation 3.2.

$$\begin{aligned} & if\ |lat_{wp} - lat_{ac}| < tol \\ & then \\ & q^2 = [\cos(lat_{ac})]^2 \\ & else \\ & q^2 = \left[\tfrac{lat_{wp} - lat_{ac}}{\delta\phi}\right]^2 \end{aligned} \tag{3.1}$$

$$\delta\phi = \ln\left[\frac{\tan(lat_2/2 + \pi/4)}{\tan(lat_1/2 + \pi/4)}\right] \tag{3.2}$$

The components of the squares in differences of latitude and longitude are calculated in

Equations 3.3 through 3.5. Note that the difference in longitude is calculated in both the East and West directions.

$$\delta lat^2 = [lat_{wp} - lat_{ac}]^2 \tag{3.3}$$

$$\delta long_{west}^2 = [\mathrm{mod}(long_{wp} - long_{ac}, 2\pi)]^2 \tag{3.4}$$

$$\delta long_{east}^2 = [\mathrm{mod}(long_{ac} - long_{wp}, 2\pi)]^2 \tag{3.5}$$

Equation 3.6 calculates the distance from the current position to the final distance in units of radians. This quantity is converted to feet in Equation 3.7.

$$\begin{array}{l} if \;\; \mathrm{mod}\,(long_{wp} - long_{ac}, 2\pi) < \mathrm{mod}(long_{ac} - long_{wp}, 2\pi) \\ d_{rad} = \sqrt{\delta lat^2 + q^2 \cdot \delta long_{west}^2} \\ else \\ d_{rad} = \sqrt{\delta lat^2 + q^2 \cdot \delta long_{east}^2} \end{array} \tag{3.6}$$

$$d_{ft} = d_{rad} \cdot \frac{180}{\pi} \cdot 60 \cdot 6076.1154856 \tag{3.7}$$

The bearing angle, the angle between the aircraft's current course and True North, is computed by Equation 3.8. Note that the output is in terms of degrees, not radians as previous quantities were. Also note that the calculation uses the atan2 function in MATLAB, which provides the four-quadrant inverse tangent.

$$\begin{array}{l} if \;\; \mathrm{mod}\,(long_{wp} - long_{ac}, 2\pi) < \mathrm{mod}(long_{ac} - long_{wp}, 2\pi) \\ then \\ bearing_{0-360} = \mathrm{mod}\left(\mathrm{atan2}\left[-\sqrt{\delta long_{west}^2}, \delta\phi\right], 2\pi\right) \cdot \frac{180}{\pi} \\ else \\ bearing_{0-360} = \mathrm{mod}\left(\mathrm{atan2}\left[\sqrt{\delta long_{east}^2}, \delta\phi\right], 2\pi\right) \cdot \frac{180}{\pi} \end{array} \tag{3.8}$$

The azimuth error is defined as in Equation 3.9, where $refcourse_{0-360}$ is the desired final reference course, or the desired bearing angle at the way-point, and $track_{0-360}$ is the aircraft ground track. Note that the azimuth error is resolved between ±180 degrees.

$$azimuth\ error_{0-360} = refcourse_{0-360} - track_{0-360} \tag{3.9}$$

Deviation is defined in Equation 3.10 and is the difference between the bearing angle from the current position to the way-point and the reference course (desired bearing angle).

$$deviation_{\pm 180} = bearing_{0-360} - refcourse_{0-360} \tag{3.10}$$

Equation 3.11 defines the course error, computed from the deviation calculated in Equation 3.10, the distance from the way-point (in feet) and the azimuth error determined in 3.9. Note the saturation function with a value of ±60 degrees. This is the maximum bank angle allowed for navigation maneuvers.

$$course\ error_{\pm 180} = sat_{\pm 60}\left[\sin deviation_{\pm 180} \cdot d_{ft} \cdot 0.007 + azimuth\ error_{\pm 180}\right] \tag{3.11}$$

Finally, the aileron input is computed from the course error. A saturation is placed on this function, as the aileron has a physical limit for its deflection which is given in Equation 3.12 as $\pm\delta a_{max}$.

$$\delta a_{cmd} = sat_{\pm \delta a_{max}}\left[-1.25 \cdot (sat_{\pm \phi_{max}}(8 \cdot course\ error_{\pm 180} - \phi_{deg}))\right] \tag{3.12}$$

3.1.3 Implementation of Rhumb-Line Controller

The Rhumb-line controller was implemented in Simulink®. The model continuously computes the course error throughout the simulation based on the current location. Simulink®block diagrams of the Rhumb-line controller are included in Figures B.8 and B.9 in Appendix B.3.

3.2 Autonomous Landing Controller

Once the aircraft has been guided to the point located on the Rhumb-line along the runway, it begins descent along the glideslope. The aircraft can be guided along this path by specifying the required altitude as a function of longitude. Although autonomous landing is not a specified goal of the work related to the RQ-2 Pioneer, the altitude and velocity controllers developed for the F-16 aircraft are beneficial in implementing the Rhumb-line navigation controller on the Pioneer.

3.2.1 Velocity Controller

The velocity controller is a proportional-integral-derivative (PID) controller. Equation 3.13 gives the form of the equation to control velocity.

$$\begin{aligned} &u = K_p e_{V_T} + K_i \int e_{V_T} + K_d \tfrac{d}{dt} e_{V_T} \\ &\text{where: } e_{V_T} = V_{T_{desired}} - V_T \end{aligned} \tag{3.13}$$

The output of the PID is the throttle command for the aircraft. The PID gains were tuned manually. Figure 3.2 shows the response of the velocity-hold controller to a step input of $10ft/s$ on the Pioneer aircraft. The overshoot that can be seen in the figure is 4.3%. A similar method was employed for determining the PID controller gains in the F-16 landing controller.

3.2.2 Altitude Controller

The altitude controller is a proportional-integral feed back (PI) controller. Equation 3.14 gives the form of the altitude controller.

$$\begin{aligned} &u = K_p e_{alt} + K_i \int e_{alt} \\ &\text{where: } e_{alt} = altitude_{desired} - altitude - roll\ compensation \end{aligned} \tag{3.14}$$

The output of the PI controller is then used to calculate appropriate aileron and elevator deflections based on the orientation of the aircraft. The gains for the PI controller were manually tuned in a similar fashion to the PID velocity controller. Figure 3.3 shows the

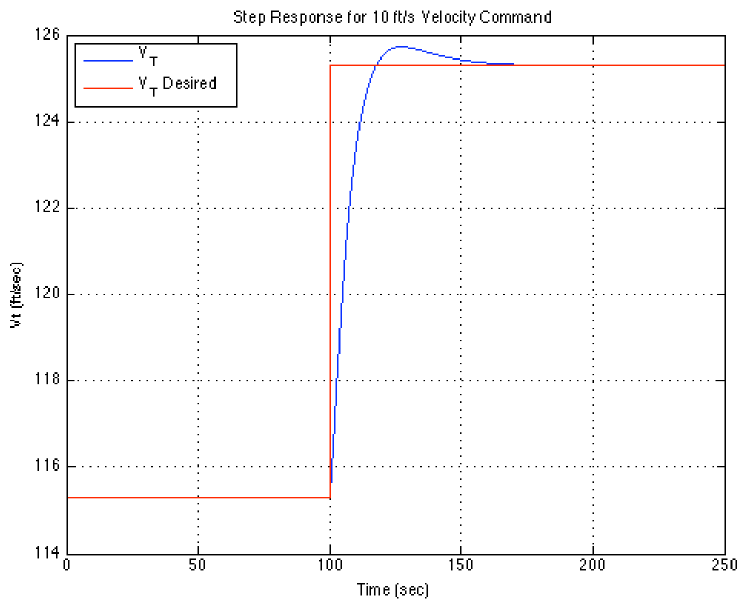

Figure 3.2: Step response for commanded change in V_T.

response of the altitude controller to a $-5ft/s$ ramp input on the Pioneer aircraft. As was the case with the velocity control, the gains on the altitude PI controller on the F-16 nonlinear model were also determined manually.

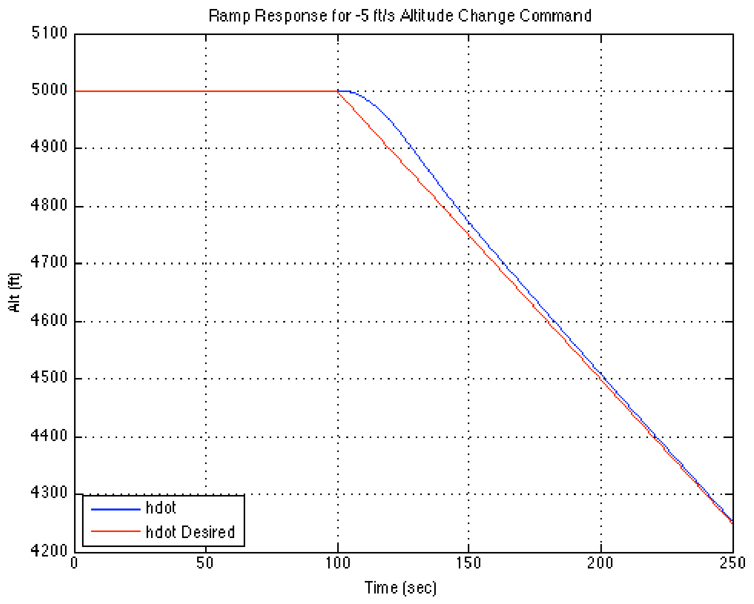

Figure 3.3: Ramp response for altitude change command.

3.2.3 Glide Slope Trajectory

The glideslope is defined as a the path that the aircraft follows from the vortac point to the runway threshold. In the case of the F-16 aircraft, the glideslope desired was a constant rate of altitude change. A relation between pitch angle, θ, and angle-of-attack, α, defines the glide slope. Equation 3.15 below defines this relation. The glideslope is usually $2.5 - 3^{\circ}$ [17].

$$\gamma = \theta - \alpha \tag{3.15}$$

The runway threshold is defined as the point at which the aircraft touches down. Just before reaching the runway point, the aircraft should perform a flare maneuver in which pitch angle is increased and airspeed quickly decreased. Conditions desired for the flare maneuver determine control surface inputs as the plane reaches the runway threshold. The purpose of the flare maneuver is to limit the vertical descent to a rate that the landing gear can absorb the landing impact of [17].

3.3 Dubins Trajectory Determination

As stated in section 1.2, the Dubins curve path determination has been used for path determination for two-dimensional vehicles. This section explains the work done by Shkel and Lumelsky in quickly determining the shortest path and then its implementation for the Dubins path navigation controller.

3.3.1 Dubins Curves

The original work of L.E. Dubins presented a method of finding the shortest path between two points that a particle can take based on the velocity vector at each point and the radius of curvature [7]. There are three options for travel that the particle has: straight line, left curve, and right curve. Using a combination of these three options, the shortest path between two points can be determined with respect to the initial and final heading angles. The resulting path will either be of the form CCC or CSC, where C corresponds to an arc with a radius ρ and S represents a straight line segment (Curved-Curved-Curved or Curved-Straight-Curved). The arcs represent a right or left turn, designated as R or L, respectively. From this conclusion, it can be seen that there are six possible general paths that exist: LSL, LSR, RSR, RSL, LRL, RLR [8].

Dubins work provides a set of curves that satisfy the required position and orientation requirements. In order to determine the shortest trajectory, the lengths of all the curves found must be calculated and compared, however the calculation can be computationally intensive. The work of Shkel and Lumelsky provided for a quick solution to the long-path case, which is of the form CSC. This long path solution is applicable for cases where $d \geq 4\rho$ and also in certain cases where $d < 4\rho$, where d is the non-dimensional distance from initial position to the final way-point [8]. Their work was specifically intended for applications such as aircraft navigation, where decreased computation time is of great importance. The algorithm for quickly finding the shortest path solution forms the foundation of the Dubins curve navigation controller developed in this work.

Three operators corresponding to the left or right circle arc and straight-line motion are given in Equation 3.16, where ϕ is the heading angle, x and y the current location, and v the segment length.

$$\begin{aligned}
L_v(x,y,\phi) &= [x + \rho\sin(\phi+v) - \rho\sin\phi, y - \rho\cos(\phi+v) + \rho\cos\phi, \phi+v] \\
R_v(x,y,\phi) &= [x - \rho\sin(\phi-v) + \rho\sin\phi, y + \rho\cos(\phi-v) - \rho\cos\phi, \phi-v] \\
S_v(x,y,\phi) &= [x + v\cos\phi, y + v\sin\phi, \phi]
\end{aligned} \tag{3.16}$$

Using these three transformations, the length of each segment along the path and thus the total path length, L, can be calculated. Specific segment lengths are noted as t, p and q, where the total path length is given in Equation 3.17.

$$L = t + p + q \tag{3.17}$$

The algorithm assumes travel from an initial point with a given heading angle, noted as (P_i, α), to a final point given as (P_f, β). As shown in Figure 3.4, α and β are the orientation angles of the vehicle from the X axis. P_i is given as the origin, $(0,0)$, and P_f is located at a point along the X-axis, $(d,0)$. Consider a vehicle traveling from $(0,0,\alpha)$ to $(d,0,\beta)$. For a vehicle traveling along a LSL (left turn-straight-left turn) path, the

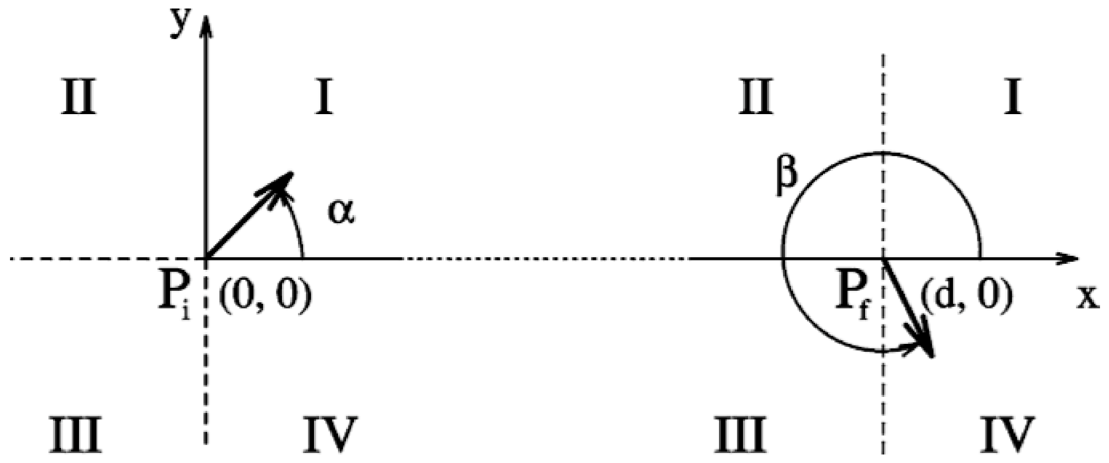

Figure 3.4: Orientation of α and β in relation to X-axis [8].

application of Equation 3.16 is given in Equation 3.18. Note: the path length computed assumes a turning radius ρ of 1.

$$L_q(S_p(L_t(0,0,\alpha))) = (d,0,\beta) \tag{3.18}$$

The position of the vehicle after completing the L_t path is given in Equation 3.19.

$$\begin{aligned} x_t &= 0 + \sin(\alpha + t) - \sin\alpha \\ y_t &= 0 - \cos(\alpha + t) + \cos\alpha \\ \phi_t &= \alpha + t \end{aligned} \tag{3.19}$$

Using (x_t, y_t, ϕ_t) as the as the input point, the S_p transformation is applied to find (x_p, y_p, ϕ_p) as displayed in Equation 3.20.

$$\begin{aligned} x_p &= \sin(\alpha + t) - \sin\alpha + p\cos(\alpha + t) \\ y_p &= -\cos(\alpha + t) + \cos\alpha + p\sin(\alpha + t) \\ \phi_p &= \alpha + t \end{aligned} \tag{3.20}$$

A left turn transformation is then applied to point (x_p, y_p, ϕ_p) in order to calculate (x_q, y_q, ϕ_q). This transformation is displayed in Equation 3.21.

$$\begin{aligned} x_q &= \sin(\alpha+t) - \sin\alpha + p\cos(\alpha+t) + \sin(\alpha+t+q) - \sin(\alpha+t) \\ y_q &= -\cos(\alpha+t) + \cos\alpha + p\sin(\alpha+t) - \cos(\alpha+t+q) + \cos(\alpha+t) \\ \phi_q &= \alpha+t+q \end{aligned} \tag{3.21}$$

Knowing that the final point (x_q, y_q, ϕ_q) is located at $(d, 0, \beta)$, equation 3.21 can be reduced to the form given in Equation 3.22.

$$\begin{aligned} d &= -\sin\alpha + p\cos(\alpha+t) + \sin(\alpha+t+q) \\ 0 &= \cos\alpha + p\sin(\alpha+t) - \cos(\alpha+t+q) \\ \beta &= \alpha+t+q \end{aligned} \tag{3.22}$$

Equation 3.22 yields a system of three equations with three unknowns. These equations can then be solved to find t, p and q for the LSL path. Equation 3.23 gives the resulting segment lengths.

$$\begin{aligned} t_{LSL} &= -\alpha + \arctan\tfrac{\cos\beta-\cos\alpha}{d+\sin\alpha-\sin\beta}[\text{mod}2\pi] \\ p_{LSL} &= \sqrt{2+d^2-2\cos(\alpha-\beta)+2d(\sin\alpha-\sin\beta)} \\ q_{LSL} &= \beta - \arctan\tfrac{\cos\beta-\cos\alpha}{d+\sin\alpha-\sin\beta}[\text{mod}2\pi] \end{aligned} \tag{3.23}$$

Substituting the segment lengths from Equation 3.23 into Equation 3.17 yields an equation for path length L_{LSL} given in Equation 3.24.

$$L_{LSL} = -\alpha + \beta + p_{LSL} \tag{3.24}$$

This process was performed for all the possible path combinations. The simplified path length calculations allowed for application of the algorithm derived by Shkel and Lumelsky. Based on the quadrant where the initial and final heading angles, α and β, are located, the shortest path can be chosen based on Table 3.1. The various possibilities of initial and final heading angle are grouped into classes, noted as a_{ij}, where i is the quadrant the initial heading angle is located in, and j the quadrant of the final heading angle. The variables S_{12}, S_{13}, etc. in table 3.1 below are switching functions calculated in cases where there exists more than one solution for a path. The switching functions are defined in Appendix A. The output of this algorithm is the set of curves (in the form CSC) and their corresponding path lengths.

Table 3.1: Lookup table for Dubins curve long path solution from Shkel and Lumelsky [8].

	$\beta = 1$	$\beta = 2$	$\beta = 3$	$\beta = 4$

$\alpha = 1$	RSL	if $S_{12} < 0$ then RSR
		if $S_12 > 0$ then RSL
			if $S_{13} < 0$ then RSR
			if $S_{13} > 0$ then LSR
				if $S^1_{14} > 0$ then LSR
				if $S^2_{14} > 0$ then RSL
				else RSR

$\alpha = 2$	if $S_{21} < 0$ then LSL
	if $S_{21} > 0$ then RSL
		if $S^1_{22} < 0$ then LSL
		if $S^1_{22} > 0$ then RSL
		if $S^2_{22} < 0$ then RSR
		if $S_{22} > 0$ then RSL
			RSR	if $S_{24} < 0$ then RSR
				if $S_{24} > 0$ then RSL

$\alpha = 3$	if $S_{31} < 0$ then LSL
	if $S_{31} > 0$ then LSR
		LSL	if $S^1_{33} < 0$ then RSR
			if $S^1_{33} > 0$ the LSR
			if $S^2_{33} < 0$ then LSL
			if $S^2_{33} > 0$ then LSR
				if $S_{34} < 0$ then RSR
				if $S_{34} > 0$ then LSR

$\alpha = 4$	if $S^1_{41} > 0$ then RSL
	if $S^2_{41} > 0$ then LSR
	else LSL
		if $S_{42} < 0$ then LSL
		if $S_{42} > 0$ then RSL
			if $S_{43} < 0$ then LSL
			if $S_{43} > 0$ then LSR
				LSR

3.3.2 Implementation of Algorithm

The algorithm for quickly determining the Dubins curve long path case, as described in the previous section, was implemented in MATLAB as a part of the navigation controller. The inputs to the trajectory determination script are the latitude and longitude of both the initial and final (desired) locations, the initial and final bearing angles, and also the turning radius of the aircraft. The output of this script is a set of vectors containing the time, latitude, longitude and bearing angle that the aircraft should follow in order to reach the desired way-point in the shortest amount of time.

In order to calculate the desired path, the initial location and desired way-point must be converted from degrees latitude and longitude to (X, Y) coordinates with units of feet. Feet are chosen as the aircraft velocity is specified in ft/s. The turning radius is also specified in units of ft. The initial point is taken as the origin, $(0,0)$ and the desired way-point is located at (X_d, Y_d). The calculation of the desired way-point in terms of feet is given in Equation 3.25.

$$\begin{aligned} X_d &= \tfrac{lon_2 - lon_1}{ft2deg * long.deg.convert} \\ Y_d &= \tfrac{lat_2 - lat_1}{ft2deg} \end{aligned} \tag{3.25}$$

The bearing angle of the aircraft is not the same as the heading angle defined in Section 3.3.1. Bearing angle, in aeronautical terms, is the angle from North to the aircraft body x-axis as depicted in Figure 3.5. In order to determine α and β, the initial and final bearing angles, ϕ_{0_d} and ϕ_{f_d} (the subscript d in this case denotes the angle is in the Dubins reference frame), must be resolved to the ground coordinate system. These conversions are shown in Equation 3.26. Because the long path Dubins curve is solved from $(0,0)$ to $(0,d)$, the distance and bearing angle from the initial point to the final way-point must also be taken into account. To accomplish this, the distance and angle between the two points are first found, shown in Equation 3.27. Note that the atan2 function is used in MATLAB rather than the standard arc tangent function, as this is a four-quadrant inverse tangent. Note, when calculating θ as well as ϕ_{0_d} and ϕ_{f_d} that the modulo operator is used with respect to 2π.

$$\begin{aligned} \phi_{0_d} &= \mathrm{mod}[-\psi_{0_g} + \pi/2, 2\pi] \\ \phi_{f_d} &= \mathrm{mod}[-\psi_{f_g} + \pi/2, 2\pi] \end{aligned} \tag{3.26}$$

$$\begin{aligned} D &= \sqrt{X_d^2 + Y_d^2} \\ \theta &= \mathrm{mod}[\arctan(Y_d/X_d), 2\pi] \end{aligned} \tag{3.27}$$

After θ has been calculated, the orientation of the aircraft can be resolved to fit the form required in Section 3.3.1, done by rotating the X, Y Dubins frame by θ radians so that the final way-point is at a location of $(0, d)$. Again, the modulo operator is used to resolve the angles between 0 and 2π as shown in Equation 3.28. The distance is also scaled as

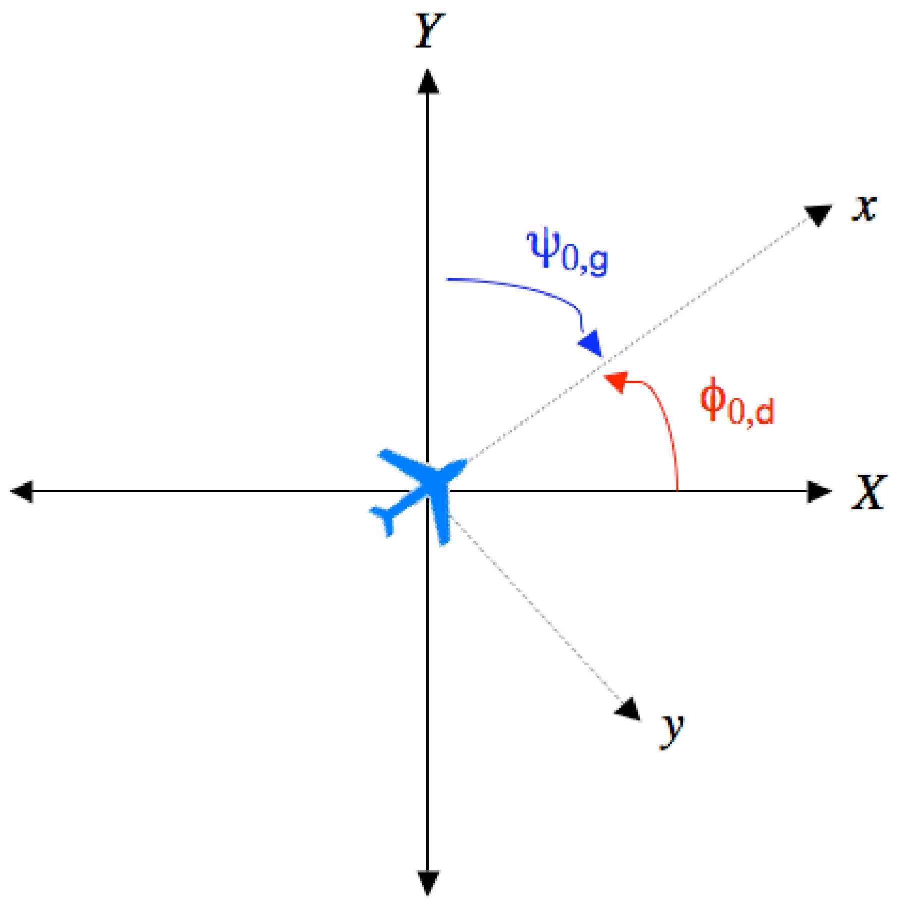

Figure 3.5: Comparison of aircraft body-axis bearing angle, ψ_{0_g}, to initial Dubins path heading angle, ϕ_{0_d}.

shown in Equation 3.29. The reason for scaling the distance by ρ is the solutions of Shkel and Lumelsky were found with the radius of curvature assumed to be equal to 1. This allows for the system of three simultaneous equations in Equation 3.16 to be solved easily for a combination of paths.

$$\begin{aligned} \alpha &= \mathrm{mod}[\phi_{0_d} - \theta, 2\pi] \\ \beta &= \mathrm{mod}[\phi_{f_d} - \theta, 2\pi] \end{aligned} \tag{3.28}$$

$$d = D/\rho \tag{3.29}$$

With the inputs transformed, Table 3.1 is implemented using conditional $if - then$ operators. In the case where there is more than one path, the switching functions mentioned in Section 3.3.1 must also be found. Once the switching function terms are calculated, conditional operators then choose the appropriate set of curves and the curve lengths are calculated. Individual functions for the LSL, LSR, RSL, & RSR paths were created in order to improve calculation time. Also output in each case from Table 3.1 is a flagging vector to signify whether the current segment is a left-turn, right-turn or straight-line. A left-turn has a value of 3, a right-turn 2, and a straight-line segment 1.

Knowing the scaled path lengths, initial and final position and bearing angles, the coordinates and bearing angle at each point along the desired trajectory can then be determined. The scaled paths are corrected using the modulo operator with respect to 2π, as shown in Equation 3.30. They are multiplied by ρ in order to scale them back to the appropriate size.

$$\begin{aligned} t_n &= \mathrm{mod}[t, 2\pi]\rho \\ p_n &= p\rho \\ q_n &= \mathrm{mod}[q, 2\pi]\rho \end{aligned} \tag{3.30}$$

The path lengths are then divided by the desired average true velocity in order to find the time to complete each path, as the average velocity remains constant. Equation 3.31 gives the equations used to calculate the path times for each segment.

$$\begin{aligned} T_t &= t_n/Vt0 \\ T_p &= p_n/Vt0 \\ T_q &= q_n/Vt0 \end{aligned} \tag{3.31}$$

The algorithm then computes the first curved segment. Equation 3.32 shows the calculation of point (a_1, b_1), located ρ feet away from the initial point. This point will be used to sweep an arc t_n radians from the starting position about (a_1, b_1). An example of this is shown in Figure 3.6. The swept path output is a set of $X - Y$ points (Dubins frame coordinates) and also the heading angle at each location. The calculation for one time step is shown in Equation 3.33. The variable t_a is from a vector created from 0 to T_t

at an interval of dt, the simulation time step. The final point of (x_{tl}, y_{tl}) is the starting point for the p, straight-line, path segment.

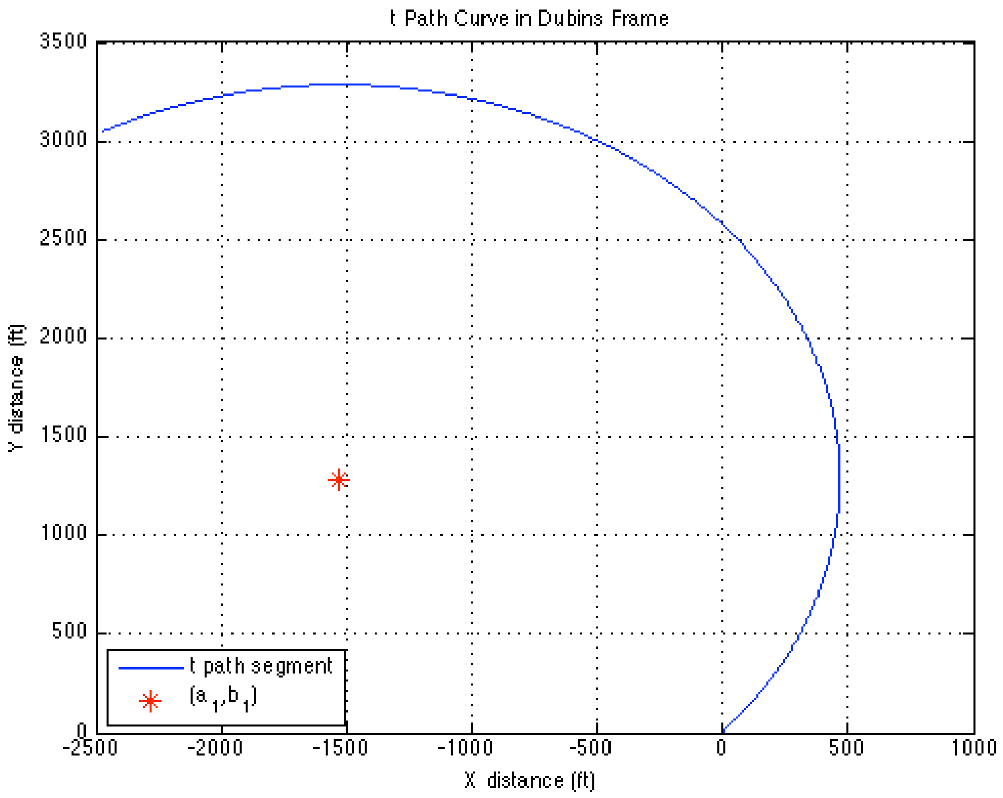

Figure 3.6: Plot of t path and point (a_1, b_1).

$$\begin{aligned}
&if\ turn1 = 3\\
&then\ \omega_t = \operatorname{mod}[\phi_{0_d} + \pi/2, 2\pi]\\
&elseif\ turn1 = 2\\
&then\ \omega_t = \operatorname{mod}[\phi_{0_d} - \pi/2, 2\pi]\\
&end\\
&a_1 = \rho \cos \omega_t\\
&b_1 = \rho \sin \omega_t
\end{aligned} \tag{3.32}$$

$$\begin{aligned} x_t(i,1) &= a_1 + \rho \cdot \cos\left[\omega_t + t_a(i,1) \cdot t_n/T_t - \pi\right] \\ y_t(i,1) &= b_1 + \rho \cdot \sin\left[\omega_t + t_a(i,1) \cdot t_n/T_t - \pi\right] \\ \phi_t(i,1) &= \phi_{0_d} + t_a(i,1) \cdot t_n/T_t \end{aligned} \tag{3.33}$$

The heading angle for segment p is calculated as shown in Equation 3.34. The points along path p (straight-line path) and heading angle are calculated in the same manner as for segment t utilizing the simulation time step. Equation 3.35 shows the calculation at each time step. The last point of the straight line segment is the initial point and bearing angle for segment q, the final turn. The points along path q and their accompanying heading angles are determined by sweeping an arc starting at the end of segment p. The equations for this step are not included as the methodology is the same as for segment t, also utilizing the turn flag to determine the appropriate point to sweep an arc about.

$$\begin{aligned} &if\ turn1 = 3 \\ &then\ \omega_p = \mathrm{mod}[\phi_{0_d} + t_n, 2\pi] \\ &elseif\ turn1 = 2 \\ &then\ \omega_p = \mathrm{mod}[\phi_{0_d} - t_n, 2\pi] \\ &end \end{aligned} \tag{3.34}$$

$$\begin{aligned} x_p(j,1) &= x_{tl} + \rho \cdot pa(j,1) \cdot p_n/T_p \cdot \cos\omega_p \\ y_p(j,1) &= y_{tl} + \rho \cdot pa(j,1) \cdot p_n/T_p \cdot \sin\omega_p \\ \phi_p(j,1) &= \omega_p \end{aligned} \tag{3.35}$$

Once the points for all three segments have been calculated, they are combined into one vector providing the desired Dubins frame position for the aircraft. Because the vector of angles is still defined in terms of heading angle, it must be resolved to aircraft bearing angle. This is shown in Equation 3.36, which is the inverse of what was done in Equation 3.26. Because the axis system that the Dubins path is generated in is not the same as the ground track system for the aircraft, the output in feet is then converted to latitude and longitude coordinates.

$$\psi_{cr} = -1 \cdot [\phi_d - \pi/2] \tag{3.36}$$

The variables x_f and y_f in Equation 3.37 above are the combined sets from the three individual t, p and q position vectors. Because there exists some roundoff error, the final bearing angle may be slightly off from the desired final bearing angle. For this reason, the desired way-point is not the vortac point for the runway. A set of points and bearing angles are added to the end of the calculated values corresponding to a path of constant bearing equal to the desired angle. This path goes from the calculated final position to the vortac point. This allows the aircraft time to correct its course before beginning the landing procedure.

$$\begin{aligned} lat_{out} &= lat_1 + y_f \cdot ft2deg \\ lon_{out} &= lon_1 + x_f \cdot ft2deg \cdot -long.deg.convert \end{aligned} \tag{3.37}$$

Included in Fgures 3.7 and 3.8 are plots of the desired Dubins curve trajectory and also of the bearing angle versus time. The simulation was run using an initial latitude of 34.9557^o and longitude of -117.716^o. The desired way-point was located at a latitude of 34.9163^o and a longitude of -117.862^o. The initial bearing angle was 40^o and the final bearing angle 70^o. Average velocity was chosen to be 115.2944 ft/s. The turning radius, ρ, was 807.061 ft. The distance in feet between these two points is 4.6110E+04. Since D is much larger than 4ρ, the long path case provides the shortest solution. The time from the initial position to reach the final way-point was 494.71 seconds.

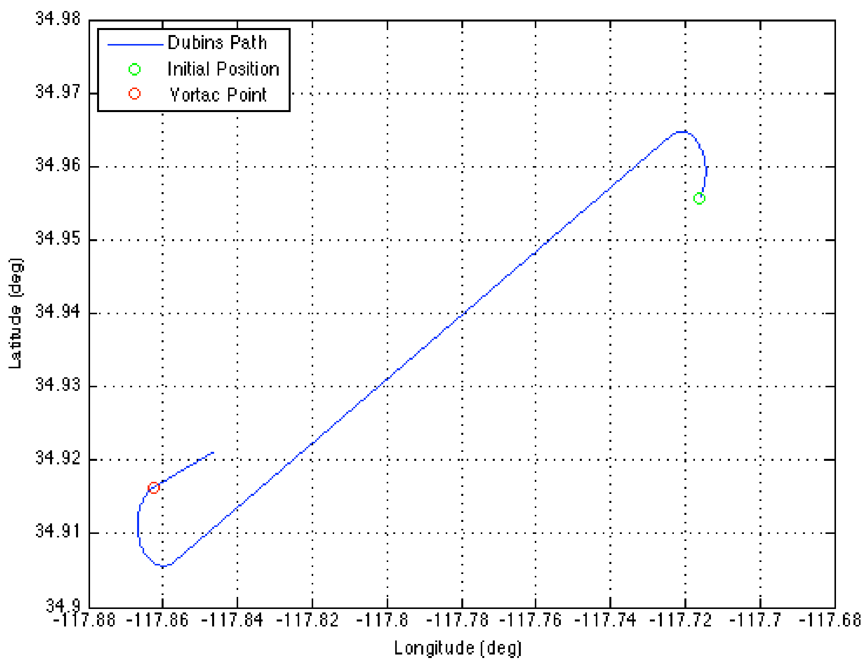

Figure 3.7: Longitude and Latitude plot of Dubins calculated path.

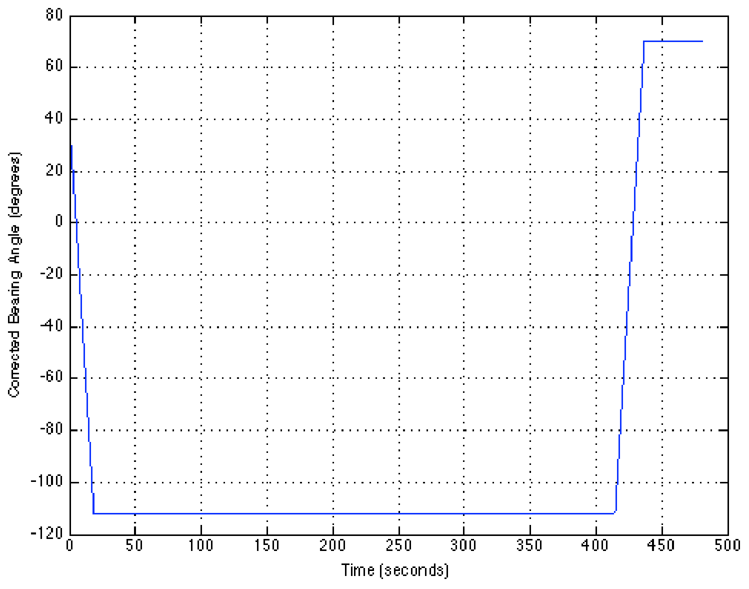

Figure 3.8: Plot of corrected bearing angle versus time for Dubins curve trajectory example

3.4 Sliding Mode Controller

The explanation of sliding mode control and the example derivation of the control law are given in the following section as presented by Slotine and Li [20]. The same approach is then taken in order to determine the control law for the Dubins simplified vehicle.

3.4.1 Sliding Mode Control

Sliding mode control is a form of robust control based on the idea that controlling a 1^{st}-order system is easier than controlling a n^{th}-order system. This notion is applicable to both nonlinear and uncertain systems. A notational simplification allows n^{th}-order problems to be represented by 1^{st}-order problems where it can be shown that good performance can be achieved, though usually at the expense of high controller activity. An explanation of this transformation and control input derivation is shown below considering a general system given by Equation 3.38. The variable $x^{(n)}$ is the scalar output, $\mathbf{x}$ is the state-variable (given by Equation 3.39) and u is the control input.

$$x^{(n)} = f(\mathbf{x}) + b(\mathbf{x})\,u \tag{3.38}$$

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & \dots & x^{(n-1)} \end{bmatrix}^T \tag{3.39}$$

The goal of the controller is for $\mathbf{x}$ to track $\mathbf{x}_d$, a time-varying state vector with specific states. This is accomplished despite inaccuracies in $f(\mathbf{x})$ and $b(\mathbf{x})$. The tracking error is represented by $\tilde{\mathbf{x}}$, which is given by Equation 3.40.

$$\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_d = \begin{bmatrix} \tilde{x} & \dot{\tilde{x}} & \dots & \tilde{x}^{(n-1)} \end{bmatrix}^T \tag{3.40}$$

The time-varying surface, $s(t)$, also known as the sliding surface, is defined in Equation 3.41. The variable λ is a strictly positive constant.

$$s(\mathbf{x};t) = \left(\frac{d}{dt} + \lambda\right)^{n-1} \tilde{x} \tag{3.41}$$

Equation 3.42 shows the sliding surface for $n = 2$.

$$s = \dot{\tilde{x}} + \lambda\tilde{x} \tag{3.42}$$

The derivative of the sliding surface, $\dot{x}$, is given by Equation 3.43.

$$\dot{s} = \ddot{x} - \ddot{x}_d + \lambda\dot{\tilde{x}} \tag{3.43}$$

Setting the sliding surface equal to 0 ensures that once the sliding surface is reached, the system remains there and the error tends towards 0. Applying this condition, the control input can be determined. For the case of a second-order system given in equation 3.44, the control input can be determined using the steps in Equations 3.41 through 3.43.

$$\ddot{x} = f(x) + u \tag{3.44}$$

The derivative of the sliding surface is the same as Equation 3.43, as $n = 2$ in this example. Substituting Equation 3.44 into Equation 3.43 and solving for 0 gives the control input u. These steps are shown in Equation 3.45.

$$\begin{gathered} \dot{s} = f(x) + u - \ddot{x}_d + \lambda\dot{\tilde{x}} = 0 \\ u = -f(x) + \ddot{x}_d - \lambda\dot{\tilde{x}} \end{gathered} \tag{3.45}$$

3.4.2 Model Transformation

In order to simplify the control system development, the Dubins aircraft model was transformed using a derivation created by Sira-Ramirez and Villeda [21]. They derived a transformation that expresses the same model with the states being $\ddot{x}$ and $\ddot{y}$ and the control inputs being u and v, the turning rate and velocity, respectively. Equation 3.46 shows the basic form of a Dubins, or non-holonomic car, subject to the constraints given in equation 3.47. Also, let w be defined as in Equation 3.48.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ u \end{bmatrix} \tag{3.46}$$

$$\begin{aligned} \psi &= \arctan\tfrac{\dot{y}}{\dot{x}} \\ v &= \sqrt{\dot{x}^2+\dot{y}^2} \end{aligned} \tag{3.47}$$

$$w = \frac{\dot{y}}{\dot{x}} \tag{3.48}$$

The derivative property in Equation 3.49 will be used for inverse trigonometric functions. Taking the time derivative of Equation 3.48 results Equation 3.50.

$$\frac{d}{dt}\arctan w = \frac{1}{1+w^2}\frac{dw}{dt} \tag{3.49}$$

$$\frac{dw}{dt} = \frac{d}{dt}\dot{y}\dot{x}^{-1} = \ddot{y}\dot{x}^{-1} - \dot{y}\dot{x}^{-2}\ddot{x} \tag{3.50}$$

Substituting Equations 3.48 and 3.50 into Equations 3.46 and 3.49 solving for u yields the solution in Equation 3.51.

$$u = \frac{\ddot{y}\dot{x} - \dot{y}\ddot{x}}{\dot{x}^2+\dot{y}^2} \tag{3.51}$$

Equation 3.52 can be found by taking the derivative of the second portion of Equation 3.47.

$$\frac{dv}{dt} = \frac{1}{2}\left(\dot{x}^2+\dot{y}^2\right)^{-1/2}(2\dot{x}\ddot{x}+2\dot{y}\ddot{y}) \tag{3.52}$$

Equation 3.52 can be rewritten in the form shown in Equation 3.53.

$$\dot{v} = \frac{\dot{x}\ddot{x} + \dot{y}\ddot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2}} \tag{3.53}$$

Equations 3.51 and 3.53 can be combined and rewritten in the matrix form shown in Equation 3.54.

$$\begin{pmatrix} u \\ \dot{v} \end{pmatrix} = \begin{bmatrix} -\frac{\dot{y}}{\dot{x}^2+\dot{y}^2} & \frac{\dot{x}}{\dot{x}^2+\dot{y}^2} \\ \frac{\dot{x}}{\sqrt{\dot{x}^2+\dot{y}^2}} & \frac{\dot{y}}{\sqrt{\dot{x}^2+\dot{y}^2}} \end{bmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} \tag{3.54}$$

The matrix equation 3.54 above can be represented in the shortened notation shown in Equation 3.55 where $U = \begin{pmatrix} u & \dot{v} \end{pmatrix}^T$ and $z = \begin{pmatrix} x & y \end{pmatrix}^T$.

$$U = \mathbf{M}\ddot{z} \tag{3.55}$$

With the system in this form, the sliding surface is defined in Equation 3.56.

$$s = (\dot{z} - \dot{z}_d) + \gamma_1(z - z_d) + \gamma_2 \int_0^\infty (z - z_d)\, dr \tag{3.56}$$

Setting the derivative of the sliding surface to zero ensures that there is no movement once the state trajectories reach the sliding surface. The derivative of Equation 3.56 is shown in Equation 3.57 where $\tilde{z} = z - z_d$.

$$\dot{s} = \ddot{z} - \ddot{z}_d + \gamma_1\dot{\tilde{z}} + \gamma_2\tilde{z} \tag{3.57}$$

Solving for $\ddot{z}$ in Equation 3.57 yields the result shown in Equation 3.58.

$$\ddot{z} = \ddot{z}_d - \gamma_1\dot{\tilde{z}} - \gamma_2\tilde{z} \tag{3.58}$$

Equation 3.58 can then be substituted back into Equation 3.55. The result of this substitution is shown in Equation 3.59, the final form of the sliding mode controller. The variables γ_1 and γ_2 are arbitrary parameters which require tuning.

$$U = \mathbf{M}(\ddot{z}_d - \gamma_1\dot{\tilde{z}} - \gamma_2\tilde{z}) \tag{3.59}$$

3.4.3 Implementation of Controller

The resulting sliding mode control equation from Section 3.4.2 was implemented in Simulink®. The controller was implemented successfully to control the Dubins vehicle simplified model with the turning rate and velocity command inputs. The trajectory generated by the Dubins curve algorithm does not change throughout the simulation. A block diagram of this controller is shown in Figure B.12, included in Appendix B.6.

3.5 Rhumb-Line Controller Utilizing Dubins Curve Trajectory

In an effort to improve the Rhumb-line controller, particularly in decreasing simulation time and also predicting time to way-point, a hybrid Dubins curve and Rhumb-line navigation controller was created. The basic Rhumb-line controller is utilized with changes made to the commanded inputs. The Dubins curve trajectory algorithm runs in the same manner as before, however rather than computing the latitude, longitude and bearing angles for each time step, the latitude and longitude at the end of the t (first curve) and p (straight-line) segments are returned. Using these two points, the Rhumb-line between $(long_t, lat_t)$ and $(long_p, lat_p)$ is computed. Equation 3.60 gives the calculation of the Rhumb-line angle, given as θ_p.

$$\begin{aligned} \delta\phi &= \log\left[\frac{\tan\left(lat_p/2+\pi/4\right)}{\tan\left(lat_t/2+\pi/4\right)}\right] \\ \theta_p &= \operatorname{atan2}\left(\frac{long_p - long_t}{\delta\phi}\right)\cdot\frac{180}{\pi} \end{aligned} \tag{3.60}$$

The Rhumb-line controller was adjusted so that the desired way-point and Rhumb-line could be changed during the simulation. The point $(long_p, lat_p)$ is the first commanded way-point. The bearing angle θ_p is the commanded bearing angle. Once the aircraft is within a given tolerance of this location, the input to the Rhumb-line controller input then becomes the location of the vortac point and its corresponding Rhumb-line. The Rhumb-line controller utilizing the Dubins trajectory was implemented on both the Dubins simplified vehicle model and the nonlinear aircraft simulation.

Chapter 4

Results

4.1 Discussion of Results

4.1.1 Dubins Simplified Model Representation

The object of the Dubins simplified model was to provide an accurate represenation that could be used to decrease simulation time in further control system development. The method in finding the aileron-turning rate gain was originally to run a number of simulations while varying the gain manually until a fit was found. The optimization routine was developed to expedite this process while also allowing the gain to be found for a variety of aircraft. In order to utilize this methodology, the altitude and velocity of the aircraft must be held constant by some outer-loop controller.

For the case where fuel is to be conserved with the Pioneer UAV, the Dubins gain was found to be -0.36. In the case where the minimum time to the way-point was desired, and thus a higher velocity, the gain was found to be -0.29. Simulations were then run using the simplified model in order to compare its tracking with the full non-linear model. Figure 4.1 displays the results of the nonlinear simulation and the Dubins simplified model. As can be seen in the figure, the Dubins vehicle tracks the non-linear model well.

The Dubins vehicle model was used for development of the Dubins path controller as well as for comparison purposes between the two control methods. The simulation time for the Rhumb-line controller with the simplified model was decreased significantly from the nonlinear simulation. The Dubins simplified model is approximately ten times quicker than the full nonlinear model. This allowed for simulations to be run with a

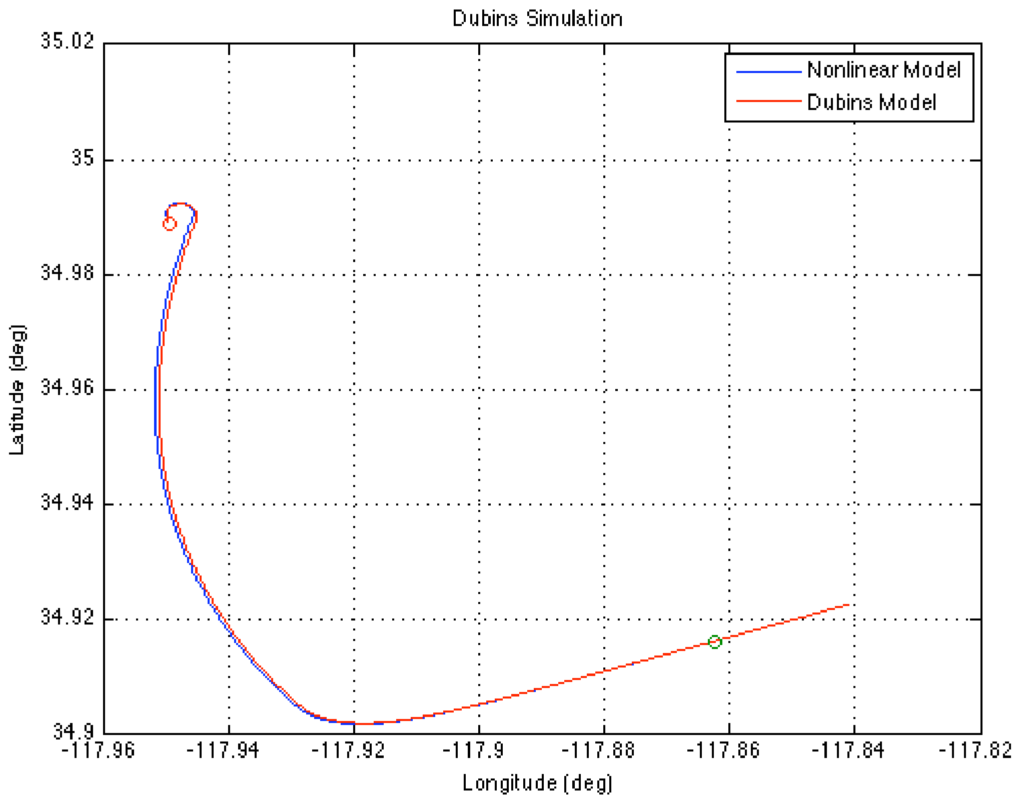

Figure 4.1: Non-linear simulation trajectory versus Dubins simplified model trajectory for a given control input.

variety of initial latitude, longitude and bearing conditions to analyze the controller's performance.

4.1.2 Rhumb-Line Controller

Dubins Vehicle Model

Simulations to check that the Rhumb-line controller works for a variety of input conditions (longitude, latitude and bearing angle) were run with the Dubins controller as the computation time is much less than the full nonlinear model. For the simulation shown in Fgure 4.2, the initial conditions are given in Table 4.1.

The simulation took 1357.30 seconds to travel from the initial location to the way-point. The bearing angle at the way-point was 70.03°.

Table 4.1: Initial Conditions for Rhumb-line Dubins Simplified Vehicle Simulation.

Initial Longitude	$-117.8994°$
Initial Latitude	$35.17233°$
Initial Bearing Angle	$10°$
Way-point Longitude	$-117.8624°$
Way-point Latitude	$34.91629°$
Way-point Bearing Angle	$70°$
V_T	$115.2944\ ft/s$
Dubins Gain	-0.36

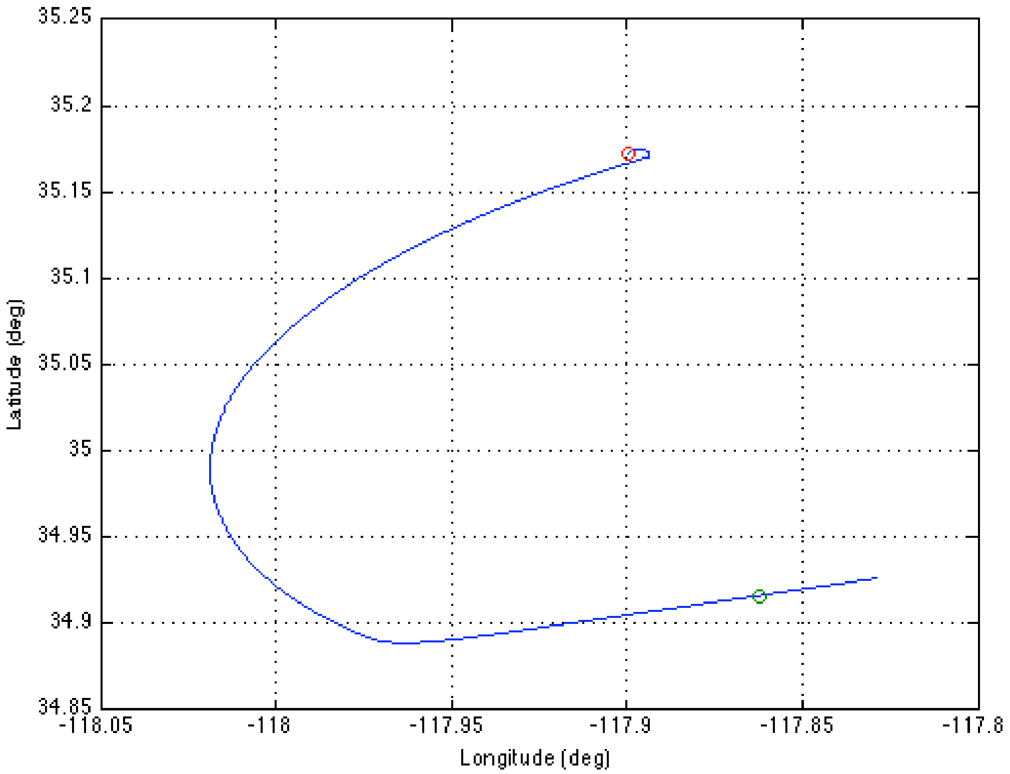

Figure 4.2: Rhumb-line control implemented on Dubins simplified vehicle model.

Nonlinear Aircraft Model

The nonlinear aircraft model was also run with same initial and final conditions given in 4.1. The time to way-point for the nonlinear simulation was 1355.85 seconds, which is within 0.1% of the Dubins simplified model simulation time. Figure 4.3 shows the trajectory of the nonlinear RQ-2 Pioneer aircraft model with the Rhumb-line controller implemented. Figures 4.4 and 4.5 display the nonlinear model aircraft body rates and orientation. The aircraft velocity is held near constant, as seen in the first subplot of Figure 4.4. It also remains near trim, as seen by the plots of α, β, P, Q and R also included in this figure. Figure 4.5 shows the orientation of the aircraft throughout the simulation and its altitude. Note that altitude remains near constant throughout the simulation. Also note that the aircraft reaches the desired bearing angle ψ of 70° at the end of the simulation.

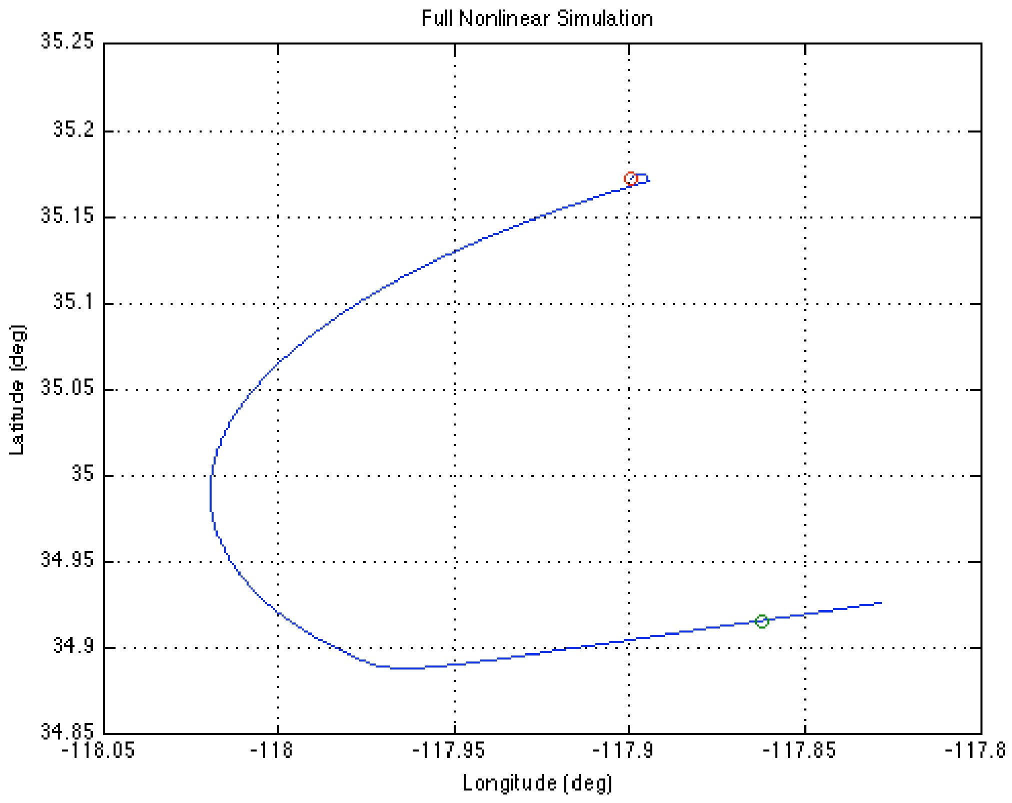

Figure 4.3: Rhumb-line control implemented on RQ-2 Pioneer nonlinear aircraft model.

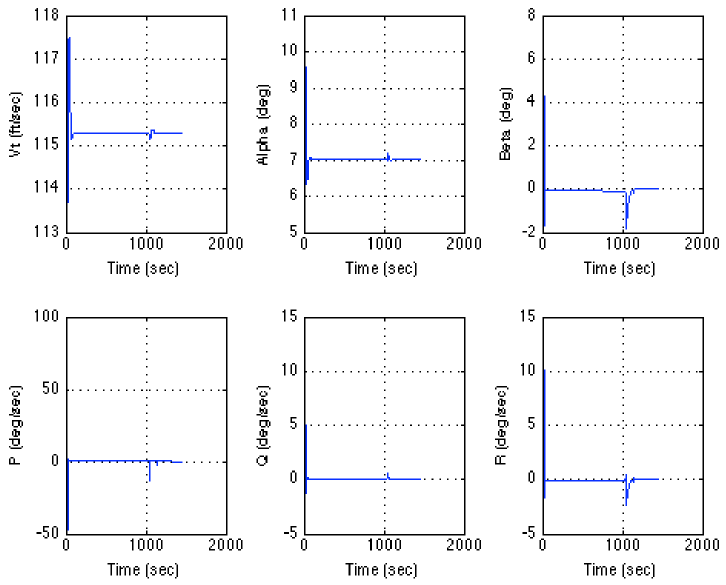

Figure 4.4: Stability axis and vehicle body axis rate plots Rhumb-line control implemented on RQ-2 Pioneer nonlinear aircraft model.

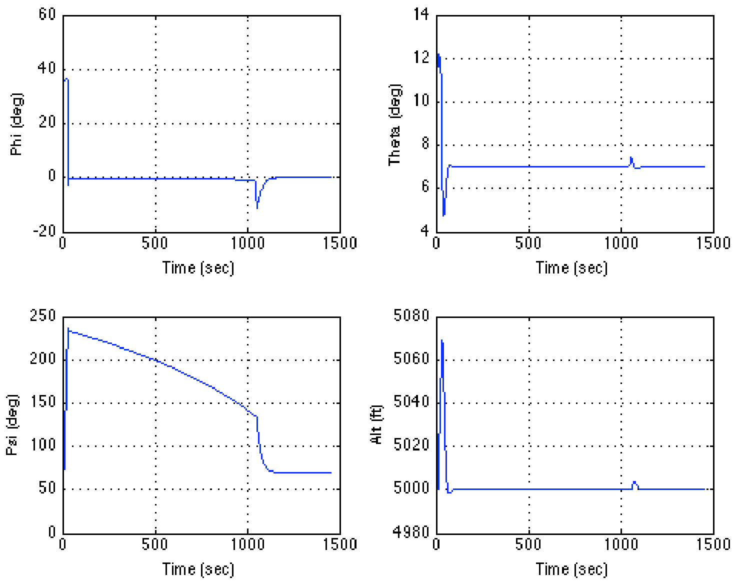

Figure 4.5: Aircraft orientation of Rhumb-line control implemented on RQ-2 Pioneer nonlinear aircraft model.

4.1.3 Dubins Curve Trajectory with Sliding Mode Controller

The sliding mode controller described in Section 3.4.2 was implemented on the Dubins vehicle simplified model. Simulations were run in order to verify that the model could accurately track the trajectory determined from the Dubins path generation algorithm in Section 3.3.2. As was done in Section 4.1.2, the sliding mode navigation controller was run for a variety of initial and final conditions. Table 4.2 gives the conditions used for the figures included in this section. Figure 4.6 shows the tracking of the Dubins simplified model with sliding mode control compared to the desired trajectory. The errors for longitude, latitude and bearing angle are shown in Figure 4.7. Considering these two figures, it can be seen that the Dubins simplified model tracks the desired trajectory well. The final steady-state error for bearing angle is 0°, indicating that the vehicle is traveling along the desired 70° Rhumb-line. The errors in longitude and latitude at the end of the simulation are a result of the vehicle traveling along the path, though slightly behind the desired point for each time step. The control effort from the sliding mode controller is shown in Figure 4.8. Figure 4.9 displays a rescaled plot of Figure 4.8.

Table 4.2: Dubins Simplified Model with Sliding Mode Control Initial and Final Conditions

Initial Longitude	$-117.7161°$
Initial Latitude	34.9557°
Initial Bearing	40°
Way-point Longitude	$-117.8624°$
Way-point Latitude	34.91629°
Way-point Bearing Angle	70°
V_T	115.2944 ft/s
ρ	800 ft
δt	0.01 s

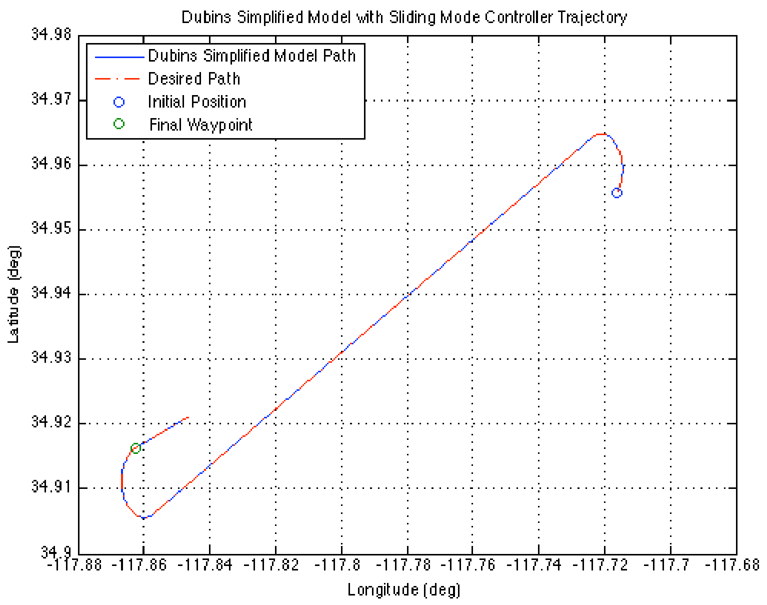

Figure 4.6: Trajectory of sliding mode controller implemented on Dubins simplified vehicle.

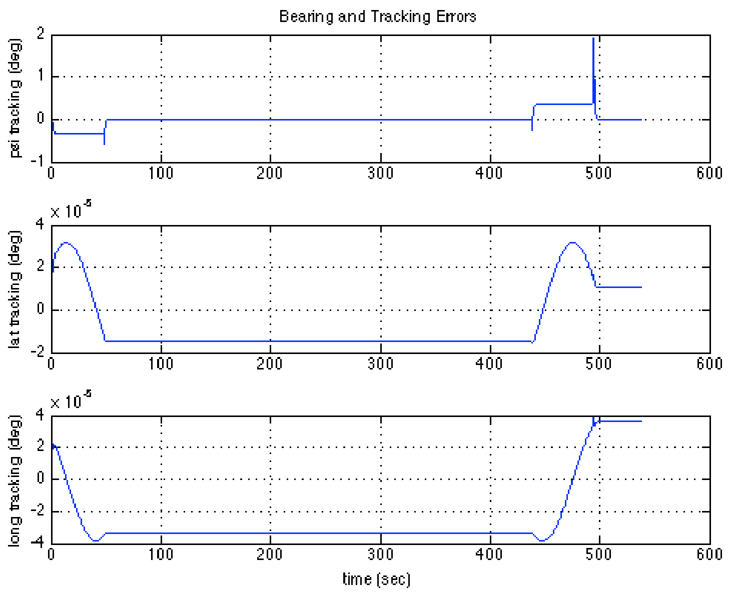

Figure 4.7: Tracking and bearing angle error for sliding mode controller implemented on Dubins simplified vehicle.

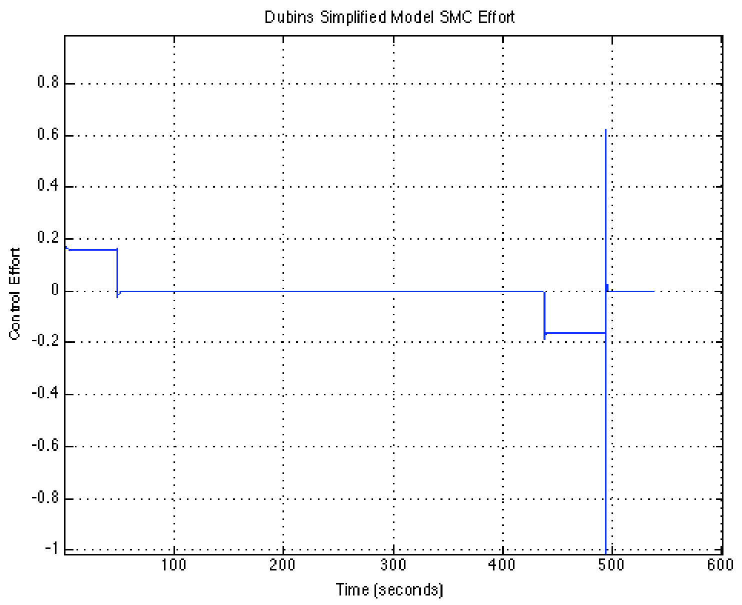

Figure 4.8: Control effort for sliding mode controller implemented on Dubins simplified vehicle.

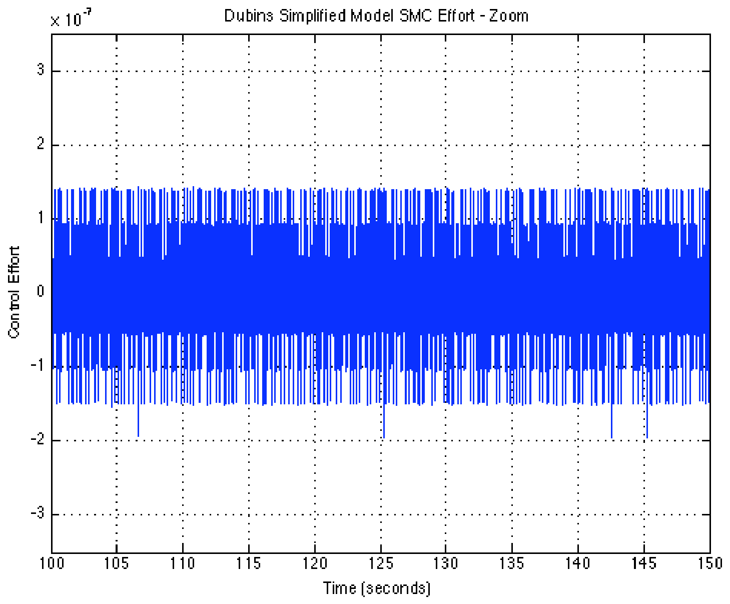

Figure 4.9: Control effort for sliding mode controller implemented on Dubins simplified vehicle.

4.1.4 Rhumb-Line Controller vs. Dubins Controller

A disadvantage of the Rhumb-Line controller is that a simulation needs to be run in order to determine the time to way-point. When the Dubins path is generated, a close estimate of the time to reach the way-point can be calculated by dividing the path length by V_{avg}. This was the original reason for switching to the Dubins path control over the Rhumb-line controller. End conditions for the comparison simulations are given in table 4.3.

Table 4.3: Simulation parameters and final conditions for Dubins path and Rhumb-line controller comparison.

Way-point Longitude	$-117.86243°$
Way-point Latittude	$-34.9163°$
Way-point ψ	$70°$
ρ (turning radius)	$800\ ft$
V_T	$115.2944\ ft/s$
δt	$0.01\ s$

As can be seen by the figures included in this section, the Dubins path is more direct in reaching the way-point than the path taken by the Rhumb-line controller. Table 4.4 shows a comparison of simulation times between the Dubins path and Rhumb-line simulations. Figures 4.10 and 4.11 show a comparison of the trajectory plots for cases 1 and 6, respectively. In case 1, the advantage of the Dubins path over the Rhumb-line controller can clearly be seen. As shown in Table 4.4, the time to reach the way-point for Rhumb-line navigation is almost twice that of the Dubins path. For case 6, although the difference in time to way-point between the two methods is not as large, the advantage can be clearly seen from Figure 4.11. The Dubins path provides a more direct trajectory to reach the way-point.

Table 4.4: Comparison of path lengths and travel times for selected Rhumb-Line and Dubins Curve trajectories.

Case	Initial Longitude (degrees)	Initial Latitude (degrees)	Initial Bearing (degrees)	Time to Waypoint (seconds)	
1	-117.6161	34.9890	10	R	1210.24
				D	710.84
2	-117.6161	34.9890	130	R	1169.61
				D	710.49
3	-117.6161	34.7557	40	R	1384.28
				D	830.18
4	-117.6161	34.7557	210	R	1390.55
				D	830.90
5	-118.1161	34.7557	-220	R	966.07
				D	830.50
6	-118.1161	34.7557	40	R	958.12
				D	813.77
7	-118.1161	35.0890	210	R	1235.98
				D	859.82
8	-118.1161	35.0890	-30	R	1257.63
				D	884.00

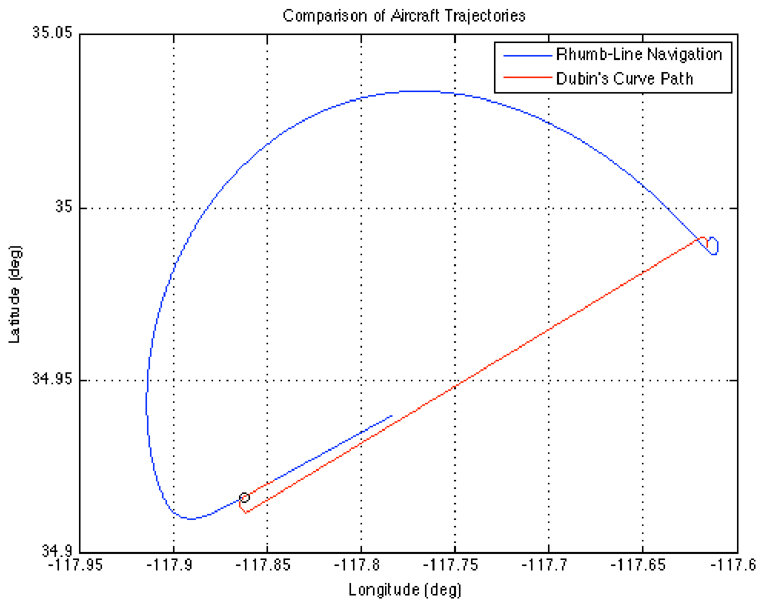

Figure 4.10: Comparison of Rhumb-line navigation and Dubins trajectory paths for case 1 from table 4.4.

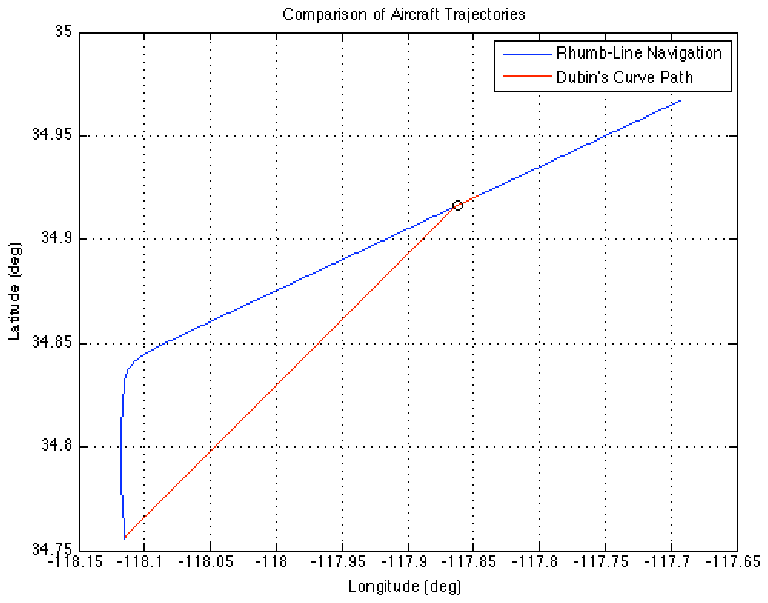

Figure 4.11: Comparison of Rhumb-line navigation and Dubins trajectory paths for case 6 from table 4.4.

4.1.5 Combined Dubins Path/Rhumb-Line Controller on Nonlinear Model

The combined Dubins/Rhumb-line controller was simulated using the nonlinear RQ-2 Pioneer aircraft model. It utilizes the calculation of the t and p segment endpoints from the Dubins path calculation, as described in section 3.5. The trajectory calculation by the Dubins algorithm is plotted against the nonlinear model simulation result in figure 4.12. Tracking along the p, straight path, segment is excellent while during the turning segments the aircraft does not follow the Dubins path. This is expected, as the inputs to the combined controller are the locations of the end of the straight segment and the final way-point. Figures 4.13 and 4.14 show the orientation, altitude, velocity and angular rates of the aircraft during the simulation. Note that in figure 4.13 the aircraft stays well within the allowed $60°$ bank angle limit. For the example provided, the predicted Dubins trajectory time was 494.71 seconds an the simulated time to waypoint was 481.81 seconds, a difference of 2.6%.

Multiple simulations were run to compare the predicted closed-form Dubins path time estimate to the time to waypoint using the hybrid controller. The simulated time to waypoint was found to be within 5% of the predicted time for a variety of scenarios. Incorporating a wind gust disturbance affected the time to waypoint, as wind gusts were assumed to be head or tail winds, but did not affect the aircrafts ability to follow the trajectory. Figure 4.15 shows the trajectory of the nonlinear aircraft with a 20 ft/s head wind incorporated. The time to reach the waypoint increased to 529.61 seconds due to the head wind, however the aircraft was able to follow the trajectory.

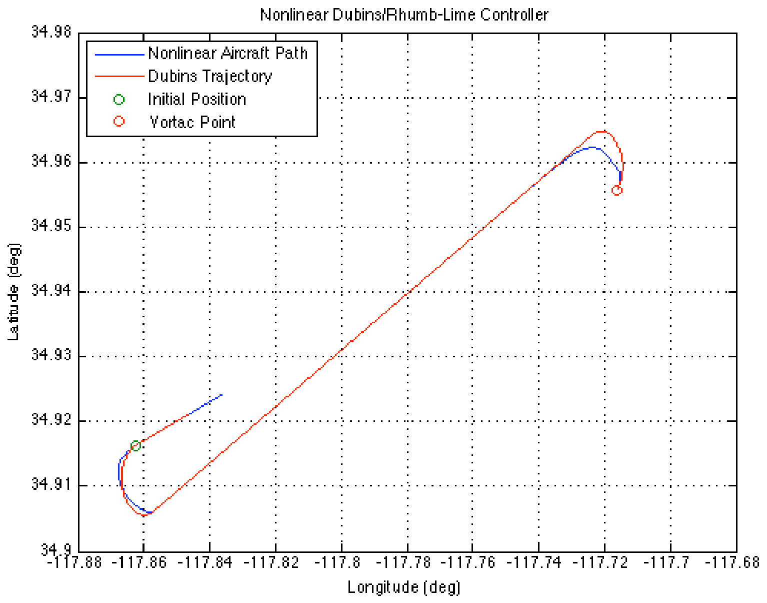

Figure 4.12: Trajectory of Dubins/Rhumb-Line controller on nonlinear model.

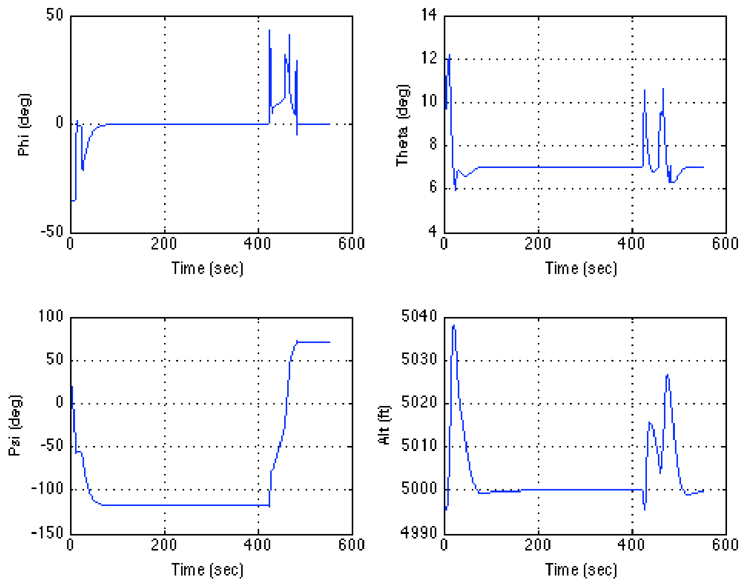

Figure 4.13: Orientation and altitude for Dubins/Rhumb-Line controller.

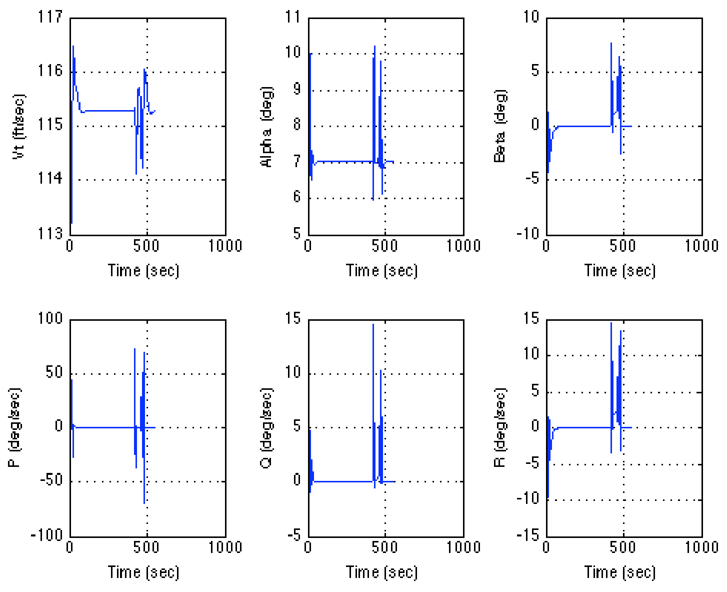

Figure 4.14: Stability axis and angular rate plots for Dubins/Rhumb-Line controller.

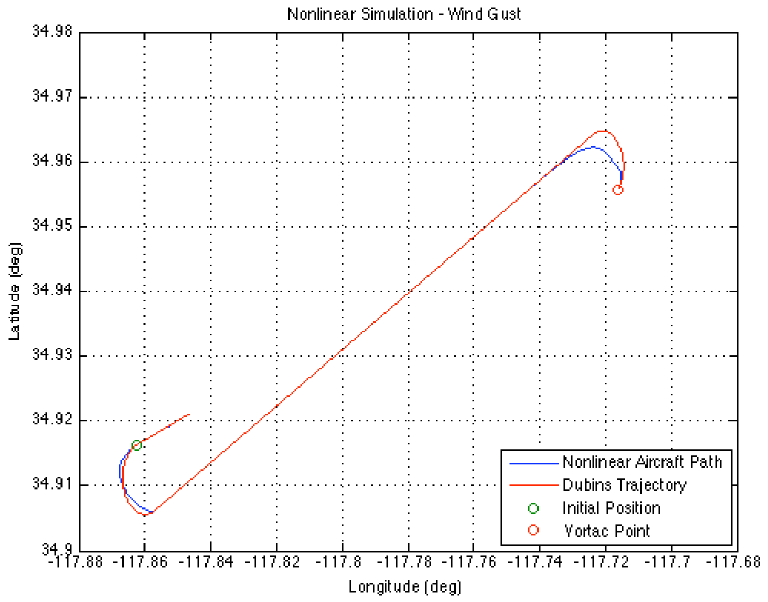

Figure 4.15: Trajectory of Dubins/Rhumb-Line controller on nonlinear model with wind gust.

4.1.6 Autonomous Landing Control

Autonomous landing control simulations were run with the nonlinear F-16 aircraft model. Simulations were run using a model that is autocoded to C by the MATLAB Real Time Workshop. This is done in order to decrease computation time due the large Simulink model for the F-16 aircraft. Note that for this simulation, the Rhumb-line controller and not the hybrid Dubins curve/Rhumb-line controller was used. Table 4.5 gives the locations and bearing angles of the way-points for the simulation. The desired bearing angle at the vortac point and runway threshold are the same as the aircraft is traveling along the glideslope during this time. Figure 4.16 shows a plot of the aircraft's longitude and latitude trajectory as well as the way-points it must pass through. More detailed plots of the aircraft's position throughout the simulation are shown in figure 4.17. The green lines on the latitude and longitude subplots represent the location of the runway threshold. The threshold was reached at a time of around 310 seconds. The ground track shows the bearing angle of the aircraft versus time. Note that on the altitude graph, the simulation ends at a height of 250 ft. This is because for testing purposes on the actual aircraft, the landing was simulated at a height of 250 ft above the runway. Figure 4.18

shows a plot of the corrected radar altitude data versus simulation time (radar altitude minus 250 ft).

Figure 4.19 shows orientation angle data and the acceleration along the z_b aircraft body axis throughout the simulation. The bank angle, ϕ, stays within the limit required during the turning maneuvers. Note that the acceleration along the z_b axis is near constant at just over -1 G during travel along the glideslope. The aircraft body angular rates, as well as velocity and control surface inputs to the horizontal tail and rudder are shown in figure 4.20. As can be seen in the figure, after the aircraft reaches the glideslope the angular rates remain at 0 deg/s until the aircraft performs the flare maneuver, which can be seen from the graph of Q, rotation about the y_b axis. The plot of V_T shows that the aircraft is traveling at a near constant rate from the initial position to the vortac point and then slowing down slightly along the glideslope. The drop off in velocity after the aircraft reaches the threshold point is a result of the flare maneuver, where the throttle is scaled back.

The throttle input to the aircraft, determined by the velocity controller, is shown in figure 4.21. The necessary throttle inputs to hold aircraft at a steady velocity can be seen from the graph as well as the decreased command for travel along the glideslope. The throttle scaling back during the flare maneuver is shown by the sharp drop-off in throttle command around 310 seconds. A plot of γ, defined as the difference between pitch angle θ and angle-of-attack α, is shown in figure 4.22. The aircraft tracks the glideslope, defined as a -3 deg path, well as shown on the graph. During the portion of the simulation where the aircraft is traveling along the glideslope, it meets the specified requirement. Calculated airspeed during the simulation is shown in figure 4.23. Note that calculated airspeed remains nearly constant until the flare maneuver is performed, after which it drops off sharply. Figure 4.24 shows a plot of the vertical rate. Fluctuations occur while the aircraft is navigating to the vortac point, after which it increases to a rate of around 15 ft/s while on the glideslope. At the point of touchdown, just after the threshold time of 310 seconds, the aircraft has a vertical rate close the desired value of 5 ft/s.

From these figures, it can be seen that the aircraft meets the required specifications for navigation to the vortac point and also performing autonomous landing.

Table 4.5: Initial, vortac and runway threshold way-points and bearing angles for F-16 landing simulation.

Initial Longitude	$-117.6161°$
Initial Latitude	$34.9890°$
Initial Bearing Angle	$-140°$ $(220°)$
Vortac Longitude	$-117.7326°$
Vortac Latitude	$34.9824°$
Vortac Bearing Angle	$238.24°$
Runway Threshold Longitude	$-117.8624°$
Runway Threshold Latitude	$34.9163°$
Runway Threshold Bearing Angle	$238.24°$

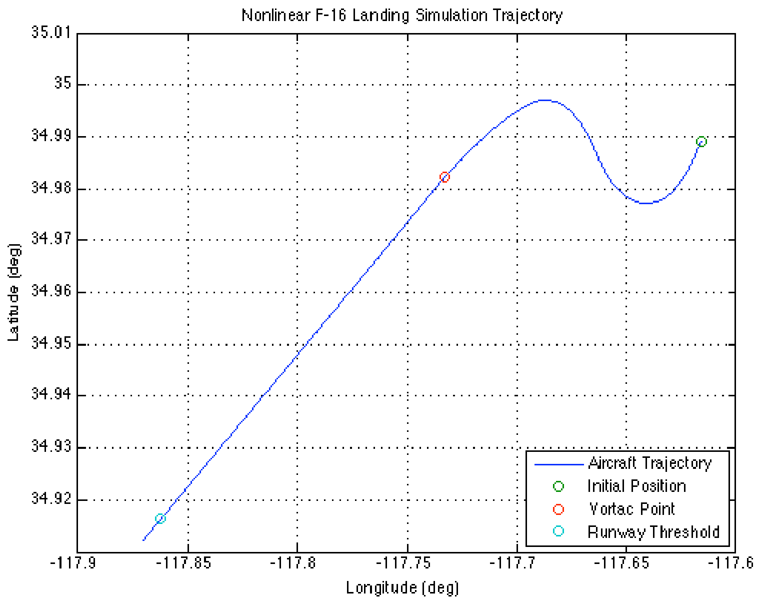

Figure 4.16: Trajectory plot for nonlinear F-16 model autonomous landing simulation.

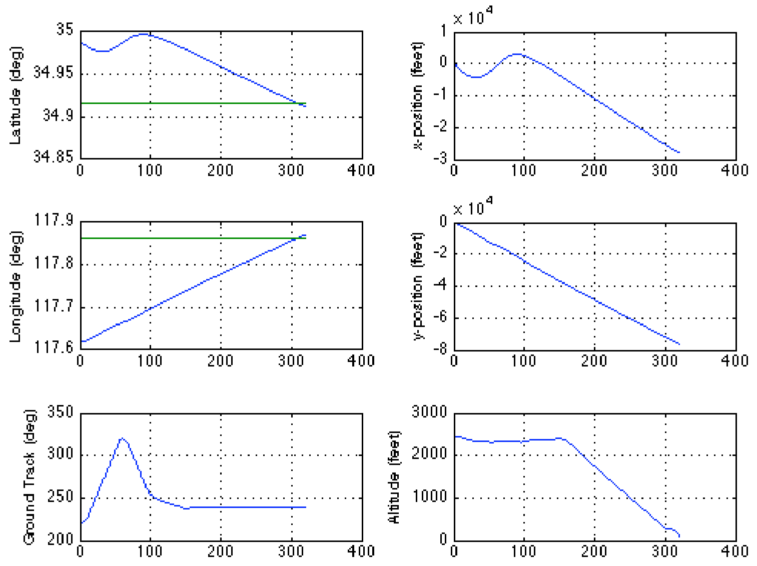

Figure 4.17: Position data for nonlinear F-16 model autonomous landing simulation.

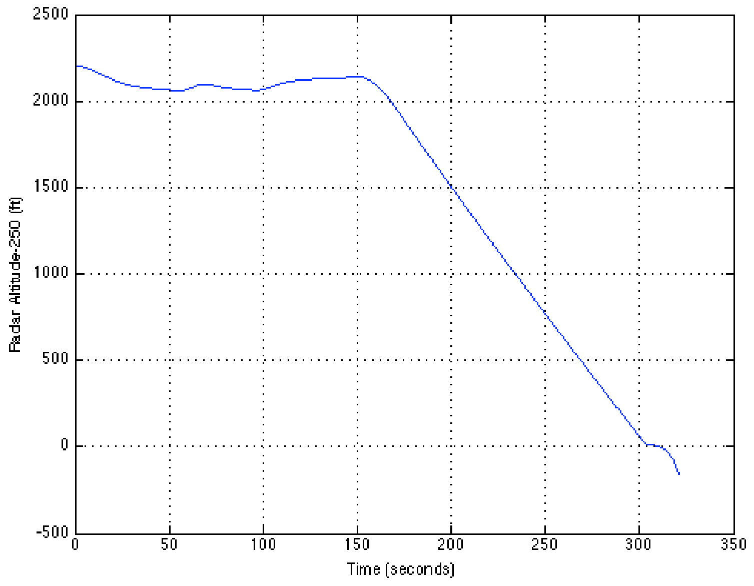

Figure 4.18: Correct radar altitude data for nonlinear F-16 model autonomous landing simulation.

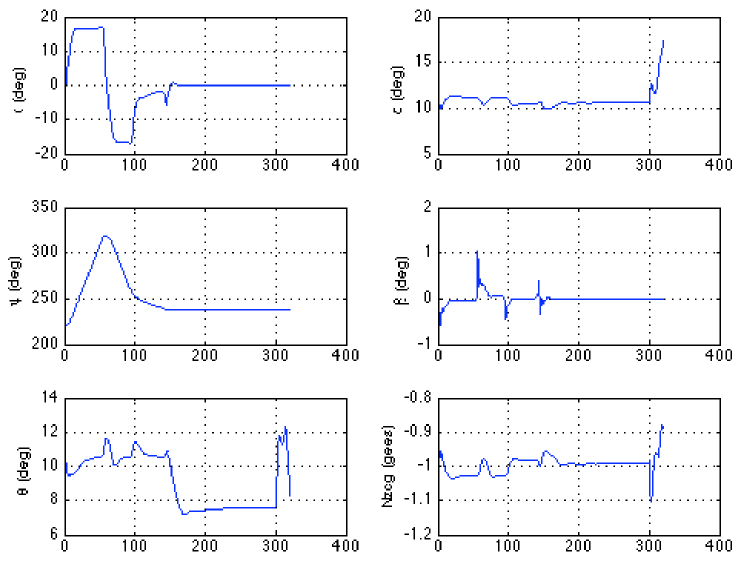

Figure 4.19: Orientation data for nonlinear F-16 model autonomous landing simulation.

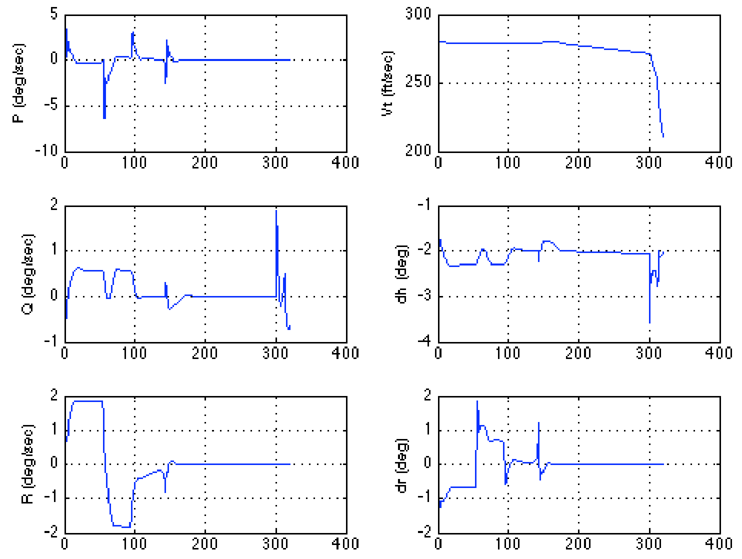

Figure 4.20: Angular rate, velocity and horizontal tail and rudder data for nonlinear F-16 model autonomous landing simulation.

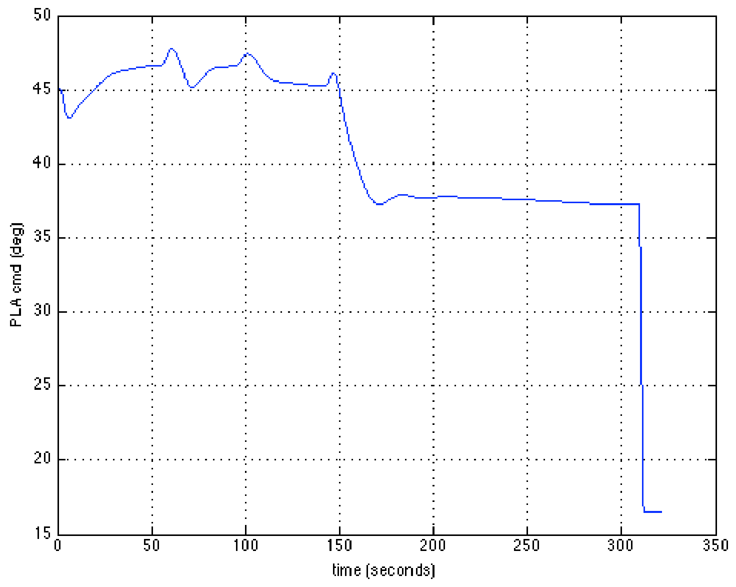

Figure 4.21: Throttle input command for nonlinear F-16 model autonomous landing simulation.

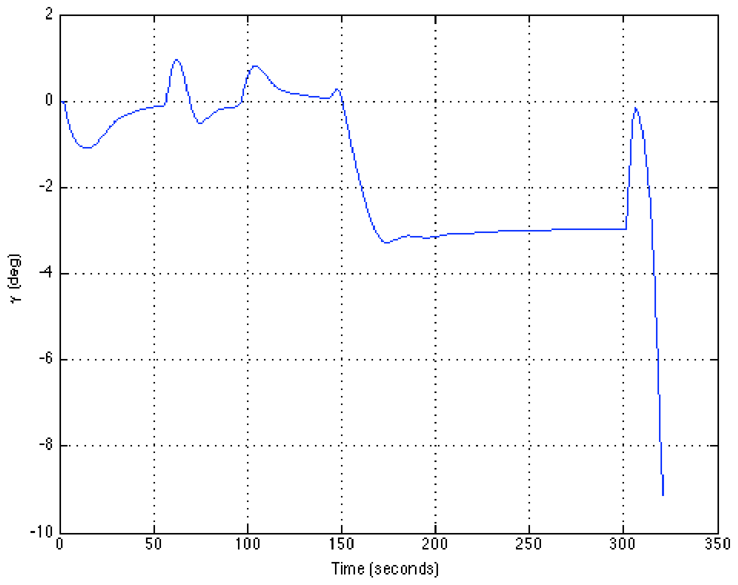

Figure 4.22: Plot of γ, defined as $\theta - \alpha$, for nonlinear F-16 model autonomous landing simulation.

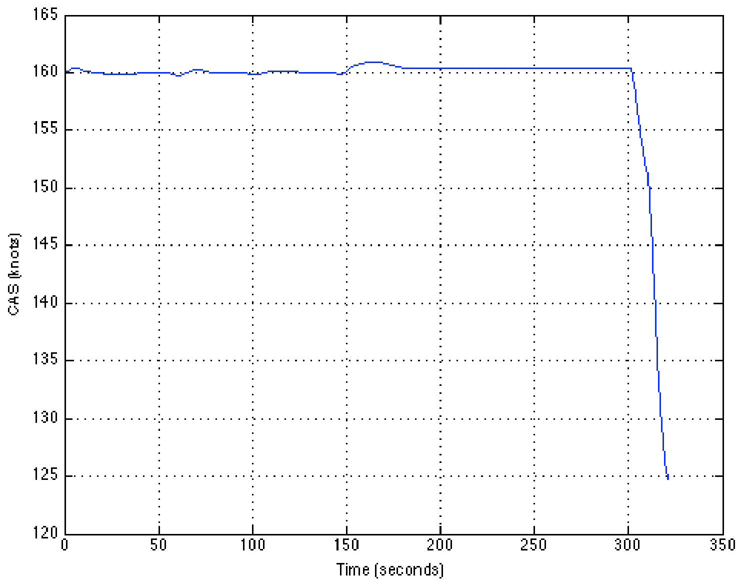

Figure 4.23: Calculated airspeed (in knots) data for nonlinear F-16 model autonomous landing simulation.

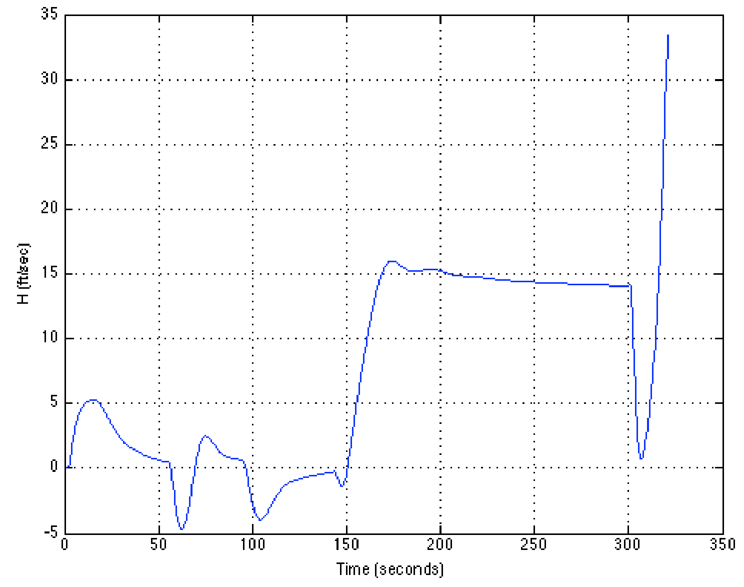

Figure 4.24: Vertical rate, $\dot{H}$ (ft/s), data for nonlinear F-16 model autonomous landing simulation.

4.1.7 Implementation on VISTA F-16 Aircraft

The autonomous landing controller was implemented and test on the Variable In-flight Stability Test Aircraft (VISTA) F-16. Equation 4.1 shows the incorrect longitude and latitude conversion used, while equation 4.2 shows the correct conversion for converting from degrees-minutes-seconds to degrees. Table 4.6 shows the incorrect and correct converted latitude and longitude coordinates for Edwards Air Force Base and the desired vortac point. Figure 4.25 shows the latitude and longitude tracking of VISTA. As displayed by the figure, the aircraft reached the desired waypoints, however due to the incorrect conversion the aircraft was not located at the correct runway point. Figure 4.26 displays the airspeed, radar altitude and angle-of-attack recorded from the aircraft during this test. Note that the runway touchdown point corresponds to the 250 ft point above the actual runway as described in the previous section.

$$\begin{array}{lcllllcrcr} Longitude & = & 117 & 51' & 44.730"W & = & -117+(51.44773)/60 & = & -117.857445°W \\ Latitude & = & 34 & 54' & 58.630"N & = & 34+(54.58630)/60 & = & 34.909771°N \end{array} \quad (4.1)$$

$$\begin{array}{lcllllcrcr} Longitude & = & 117 & 51' & 44.730"W & = & -117+[51+44.7730/60]/60 & = & -117.862425°W \\ Latitude & = & 34 & 54' & 58.630"N & = & 34+[54+58.630/60]/60 & = & 34.916286°N \end{array} \quad (4.2)$$

Table 4.6: Comparison of latitude and longitude values for VISTA flight testing.

	FAA Database (deg-min-sec)	Incorrect Waypoints (deg)	Correct Waypoins (deg)
EDW Vortac	34 58′ 56.500”N 117 43′ 57.380”W	34.976080° −117.72623°	34.98236° −117.732605°
Rwy22	34 54′ 58.630” 117 51′ 44.730”W	34.909771° −117.857455°	34.916286° −117.862425°

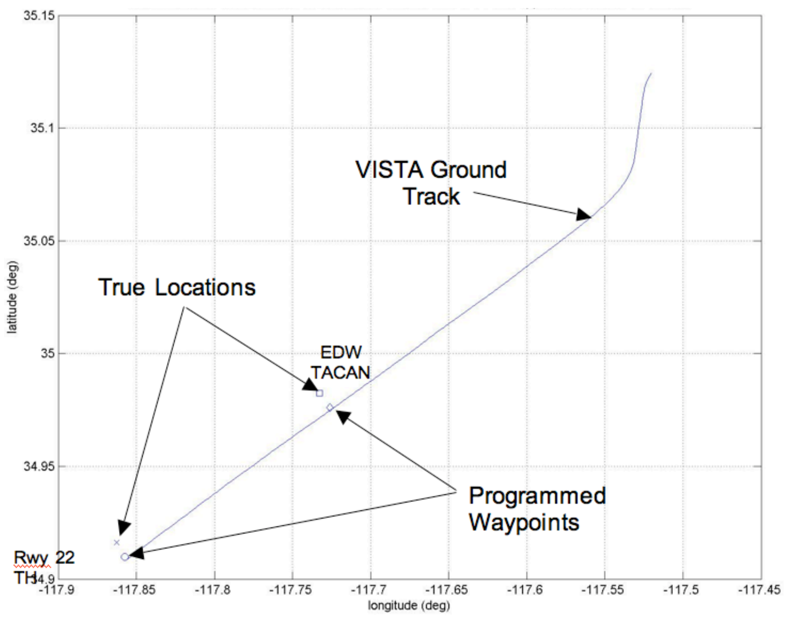

Figure 4.25: Outer loop controller ground track during autoland approach.

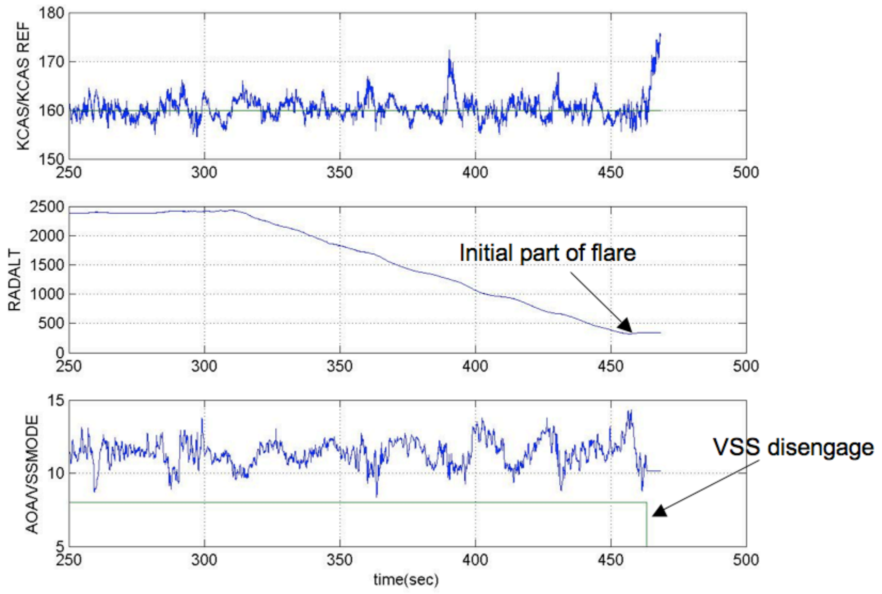

Figure 4.26: Outer loop controller airspeed, radar altitude and angle-of-attack during autoland approach.

Chapter 5

Conclusion & Future Work

5.1 Conclusion

This work led to the development of autonomous navigation and landing controllers for implementation on pre-existing aircraft. Techniques developed within this work were compared to previously used methods for path determination and control for tracking the desired trajectory. It was shown that the newly developed methods provide comparable performance to previous work. The results of this work are beneficial as the role of UAVs increases in both military and commercial applications.

The Rhumb-line navigation controller and associated velocity and altitude hold controllers were implemented on a Lockheed-Martin F-16 aircraft. Results from the non-linear simulation showed that the aircraft was able to successfully navigate from various initial positions to a final way-point (vortac point). Once the vortac point was reached, the controller was able to guide the aircraft along the pre-determined glideslope for the purpose of autonomous landing.

The Rhumb-line controller also formed the basis of work for the guidance and control system implemented on the RQ-2 Pioneer. With navigation and velocity hold controllers similar to those on the F-16, the Pioneer aircraft was modeled as a Dubins vehicle constrained to a two-dimensional surface. This proved to be beneficial in the development and refinement of the Dubins path controller, as computation time for simulations was significantly decreased. The Dubins path generation algorithm proved to be favorable over the Rhumb-line controller as the aircraft takes a more direct path to the way-point. Another strength of the Dubins path algorithm was that it provided a closed-form estimation of the time to way-point, which the Rhumb-line controller is unable to provide without running a simulation with the desired initial and final conditions. Sliding mode control was used on the Dubins simplified model as the already designed Rhumb-line controller was inadequate for guiding the aircraft along the Dubins path. The hybrid Dubins path/Rhumb-line controller provide a method for using the quick Dubins path on the nonlinear aircraft model. Results from the Dubins path/Rhumb-line controller showed that the nonlinear aircraft tracks the straight-path segment and arrives at the way-point within 5% of the predicted time. The combined controller also proved acceptable in guiding the aircraft to the final way-point in the presence of external disturbance in the form of a wind gust.

The end result of this work are methods for autonomous navigation and landing of a UAV for two types of aircraft. The implementation of the control scheme for the F-16 allows the aircraft to act as a drone without the expense of modifying internal flight code, while on the RQ-2 Pioneer it allows for a decrease in the role of human interaction.

5.2 Future Work

The products of this work were two techniques for autonomous navigation and landing of unmanned aircraft. The simulations completed demonstrate that the control techniques work for nonlinear aircraft models. Future related work involves implementation and simulation of the sliding mode controller onto a nonlinear aircraft model for improved accuracy in trajectory tracking and the closed-form estimate of time to way-point. Refinements can also be made in regard to the Dubins path generation algorithm. At present, the turning radius must be specified by the user. The effect of varying the turning radius on performance is an area open to investigation. The short-path case solution is also worth consideration for comparison to the Rhumb-line navigation method when the aircraft is already relatively close to the desired way-point.

Bibliography

[1] Surveillance USAF Intelligence, USAF Intelligence and Reconnaissance. Air Force unmanned aerial system flight plan 2009-2047. *http://www.af.mil/shared/media/document/AFD-090723-034.pdf*, July 2009.

[2] B. Clough. Unmanned aerial vehicles: Autonomous control challenges, a researcher's perspective. *Proceedings of 2nd AIAA Unmanned Unlimited Systems, Technologies and Operations - Aerospace Conference*, 2003.

[3] T. Redling. Integrated flight control systems - a new paradigm for an old art. *IEEE AES Systems Magazine*, pages 17–22, 2001.

[4] Y. Wang, X. Li, and Y. Huang. Navigation system of pilotless aircraft via GPS. *IEEE AES Systems Magazine*, pages 16–20, 1996.

[5] C.A. Baird and F.B. Snyder. Terrain-aided altitude computations on the AFTI/F-16. *IEEE Position Location and Navigation Symposium*, pages 474–481, 1990.

[6] S. Snyder et. al. Differential GPS/inertial navigation approach/landing flight test results. *IEEE AES Systems Magazine*, pages 3–11, 1992.

[7] L.E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[8] V. Lumelsky A.M. Shkel. Classification of the Dubins set. *Robotics and Autonomous Systems*, 34:179–292, 2001.

[9] S. Jeyaraman, A. Tsourdos, and R. Zbikowski. Formalised hybrid control scheme for a UAV group using Dubins set and model checking. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 5246–5251, 2004.

[10] G. Tang, Z. Wang, and A.L. Williams. On the construction of an optimal feedback control law for the shortest path problem for the Dubins car-like robot. *Proceedings of the 30th Southeatern Symposium on System Theory*, pages 280–284, 1998.

[11] A. Balluchi, A. Bicchi, B. Piccoli, and P. Soueres. Stability and robustness of optimal synthesis for route tracking by Dubins' vehicles. *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 581–586, 2000.

[12] H. Chitsaz and S.M. LaValle. Time-optimal paths for a Dubins airplane. *IEEE Conference on Decision and Control*, pages 2379–2384, 2007.

[13] A. Bhatia, M. Graziano, S. Karaman, R. Naldi, and E. Frazzoli. Dubins trajectory tracking using commercial off-the-shelf autopilots. *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2008.

[14] O. Yakimenko and I. Kaminer. Near-optimal trajectory generation for autonomous aircraft landing. *Proceedings of International Symposium on Computer Aided Control System Design*, pages 445–450, 1999.

[15] L. Shaoyan and C. Zongji. Research of autonomous landing control of unmanned combat air vehicle. *Proceedings of Fifth International Symposium on Instrumentation and Control Technology*, pages 769–773, 2003.

[16] A.A. Pashilkar, N. Sudararajan, and P. Saratchandran. A fault-tolerant neural aided controller for aircraft auto-landing. *Aerospace Science and Technology*, pages 49–61, 2006.

[17] R.C. Nelson. *Flight Stability and Automatic Control*. McGraw-Hill, 1998.

[18] S. Park and C. Kee. Enhanced method for single-antenna GPS-based attitude determination. *Aircraft Engineering and Aerospace Technology*, 78(3):236–243, 2006.

[19] C.A. Furuti. Map projections: Properties: Preserving directions. *http://www.progonos.com.br/furuti/MapProj/Dither/CartProp/Rhumb/rhumb.html*, 29 July 2009.

[20] J.J. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, 1991.

[21] H. Sira-Ramirez and L.I.L. Villeda. Sliding mode, delta-modulation and output feedback control of dynamic systems. In A. Sabanovic, L.M. Fridman, and S. Spurgeon, editors, *Variable Structure Systems: from Principles to Implementation*, pages 157–218. Institution of Electrical Engineers, 2004.

Appendix A

Switching Functions

$$
\begin{array}{lll}
\text{Class } a_{11} & : & \text{unique solution} \\
\text{Class } a_{12} & : & S_{12} = f(p_{rsr}, p_{rsl}, q_{rsl}) = p_{rsr} - p_{rsl} - 2(q_{rsl} - \pi) \\
\text{Class } a_{13} & : & S_{13} = g(t_{rsr}) = t_{rsr} - \pi \\
\text{Class } a_{14} & : & \begin{array}{l} S_{14}^{1} = g(t_{rsr}) = t_{rsr} - \pi \\ S_{14}^{2} = g(q_{rsr}) = q_{rsr} - \pi \end{array} \\
\text{Class } a_{21} & : & S_{21} = f(p_{lsl}, p_{rsl}, t_{rsl}) = p_{lsl} - p_{rsl} - 2(t_{rsl} - \pi) \\
\text{Class } a_{22} & : & \begin{array}{l} \text{if} \alpha > \beta, \text{then} S_{22}^{1} = f(p_{lsl}, p_{rsl}, t_{rsl}) = p_{lsl} - p_{rsl} - 2(t_{rsl} - \pi) \\ \text{if} \alpha < \beta, \text{then} S_{22}^{2} = f(p_{rsr}, p_{rsl}, q_{rsl}) = p_{rsr} - p_{rsl} - 2(q_{rsl} - \pi) \end{array} \\
\text{Class } a_{23} & : & \text{unique solution} \\
\text{Class } a_{24} & : & S_{24} = g(q_{rsr}) = q_{rsr} - \pi \\
\text{Class } a_{31} & : & S_{31} = g(q_{lsl}) = q_{lsl} - \pi \\
\text{Class } a_{32} & : & \text{unique solution} \\
\text{Class } a_{33} & : & \begin{array}{l} \text{if} \alpha < \beta, \text{then} S_{33}^{1} = f(p_{rsr}, p_{lsr}, t_{lsr}) = p_{rsr} - p_{lsr} - 2(t_{lsr} - \pi) \\ \text{if} \alpha > \beta, \text{then} S_{33}^{2} = f(p_{lsl}, p_{lsr}, q_{lsr}) = p_{lsl} - p_{lsr} - 2(q_{lsr} - \pi) \end{array} \\
\text{Class } a_{34} & : & S_{34} = f(p_{rsr}, p_{lsr}, t_{lsr}) = p_{rsr} - p_{lsr} - 2(t_{lsr} - \pi) \\
\text{Class } a_{41} & : & \begin{array}{l} S_{41}^{1} = g(t_{lsl}) = t_{lsl} - \pi \\ S_{41}^{2} = g(q_{lsl}) = q_{lsl} - \pi \end{array} \\
\text{Class } a_{42} & : & S_{42} = g(t_{lsl}) = t_{lsl} - \pi \\
\text{Class } a_{43} & : & S_{43} = f(p_{lsl}, p_{lsr}, q_{lsr}) = p_{lsl} - p_{lsr} - 2(q_{lsr} - \pi) \\
\text{Class } a_{44} & : & \text{unique solution}
\end{array}
\tag{A.1}
$$

Appendix B

Simulink Diagrams

B.1 F-16 Control System

B.1.1 Outer Loop Conrol Law

Figure B.1 shows the overall architecture of the outer-loop controller built in Simulink®. The subsystem "Outer Loop Auto CLAW" (blue box in diagram is the developed controller containing the altitude hold, velocity hold and Rhumb-line control navigation system. The three yellow subsystems are the pitch, roll and yaw reverse gradient and breakouts.

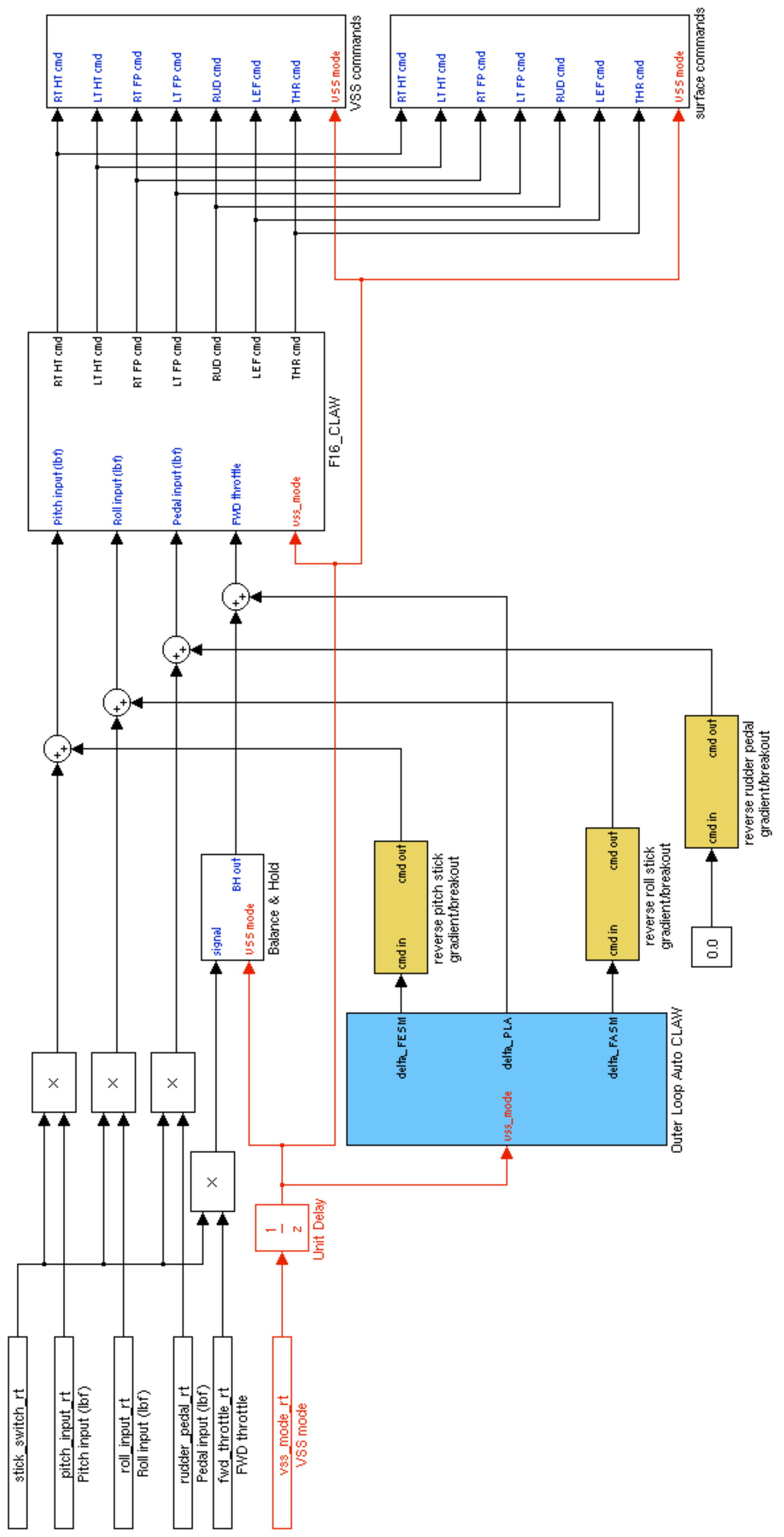

Figure B.1: Nonlinear Simulink model of F-16 outer-loop control architecture.

B.1.2 Reverse Gradient/Gain and Breakout

Reverse Pitch Gradient and Breakout

Figure B.2 displays a Simulink® diagram of the pitch reverse gradient and break out subsystem. Figure B.3 shows the "reverse_pitch_gradient" subsystem in figure B.2.

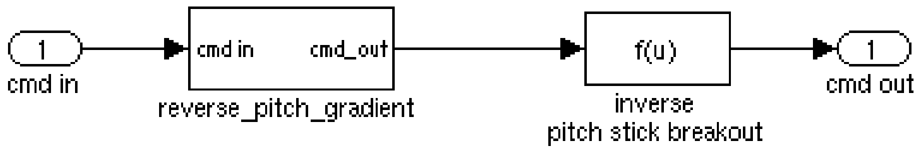

Figure B.2: Pitch reverse gradient and breakout.

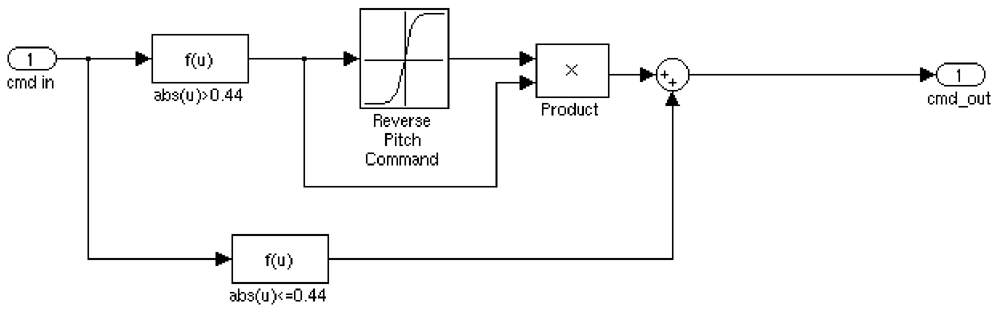

Figure B.3: Pitch gradient subsystem.

Reverse Roll Gradient and Breakout

Figure B.4 displays a Simulink® diagram of the roll reverse gradient and break out subsystem. Figure B.5 shows the "reverse_roll_gradient" subsystem in figure B.4. Note that figure B.5 contains two gradients based on the value of the commanded input.

Reverse Rudder Pedal Gain and Breakout

Figure B.6 displays a Simulink® diagram of the rudder reverse gain and breakout.

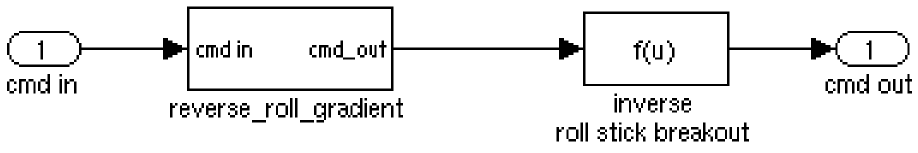

Figure B.4: Roll reverse gradient and breakout.

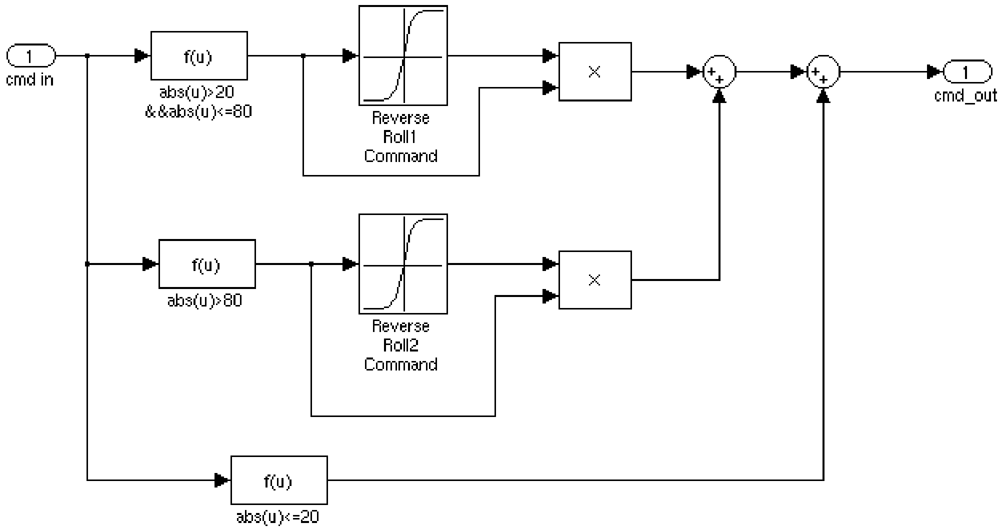

Figure B.5: Roll gradient subsystem.

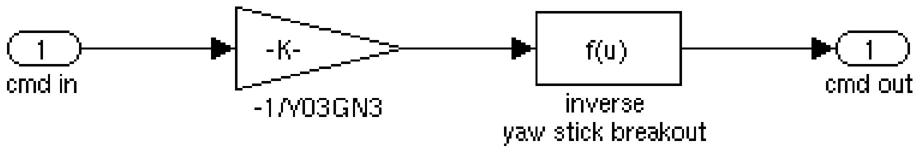

Figure B.6: Rudder reverse gain and breakout.

B.2 Open Loop Nonlinear Aircraft Model

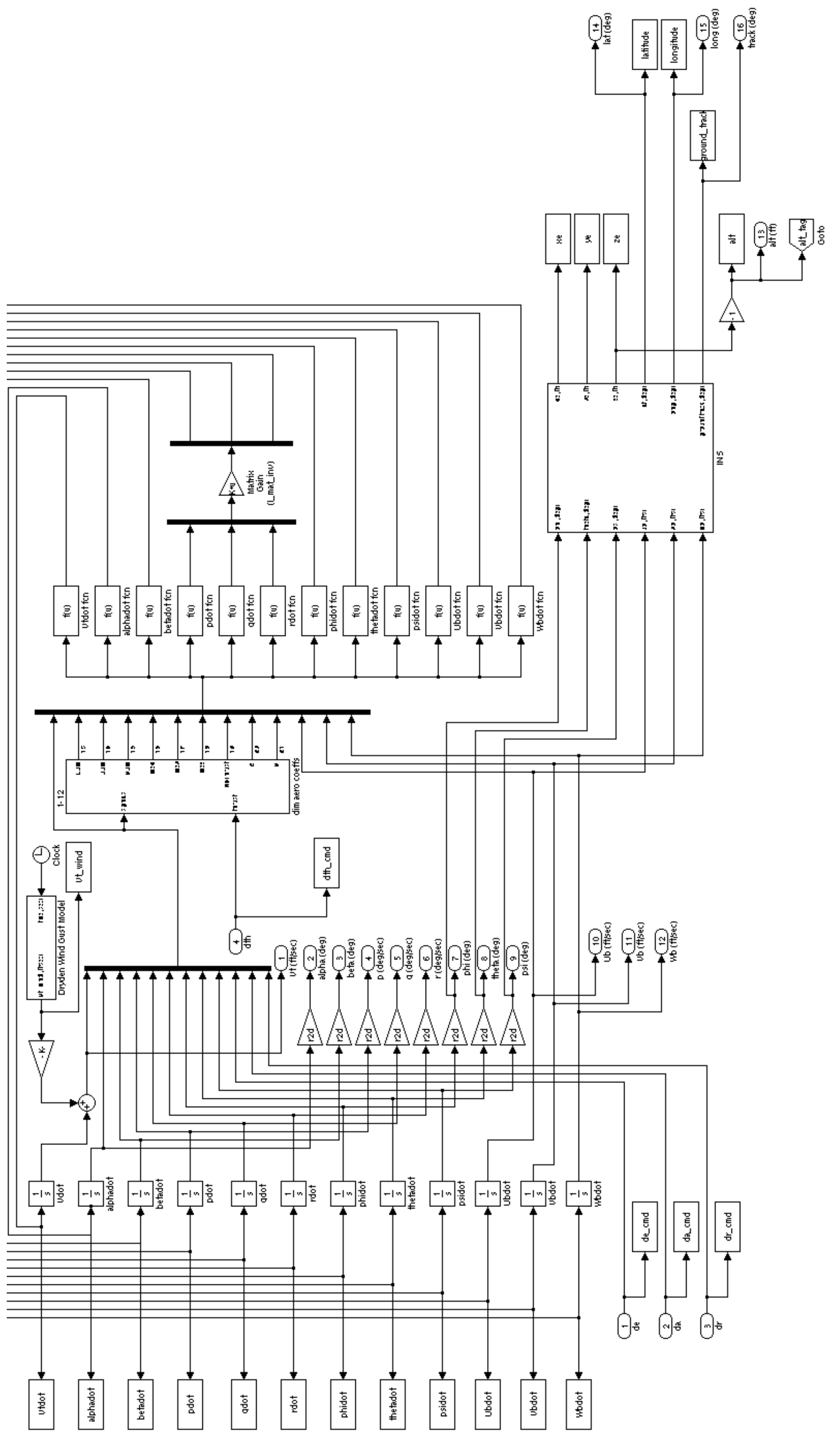

Figure B.7: Non-linear simulink model.

B.3 Rhumb-Line Navigation Controller

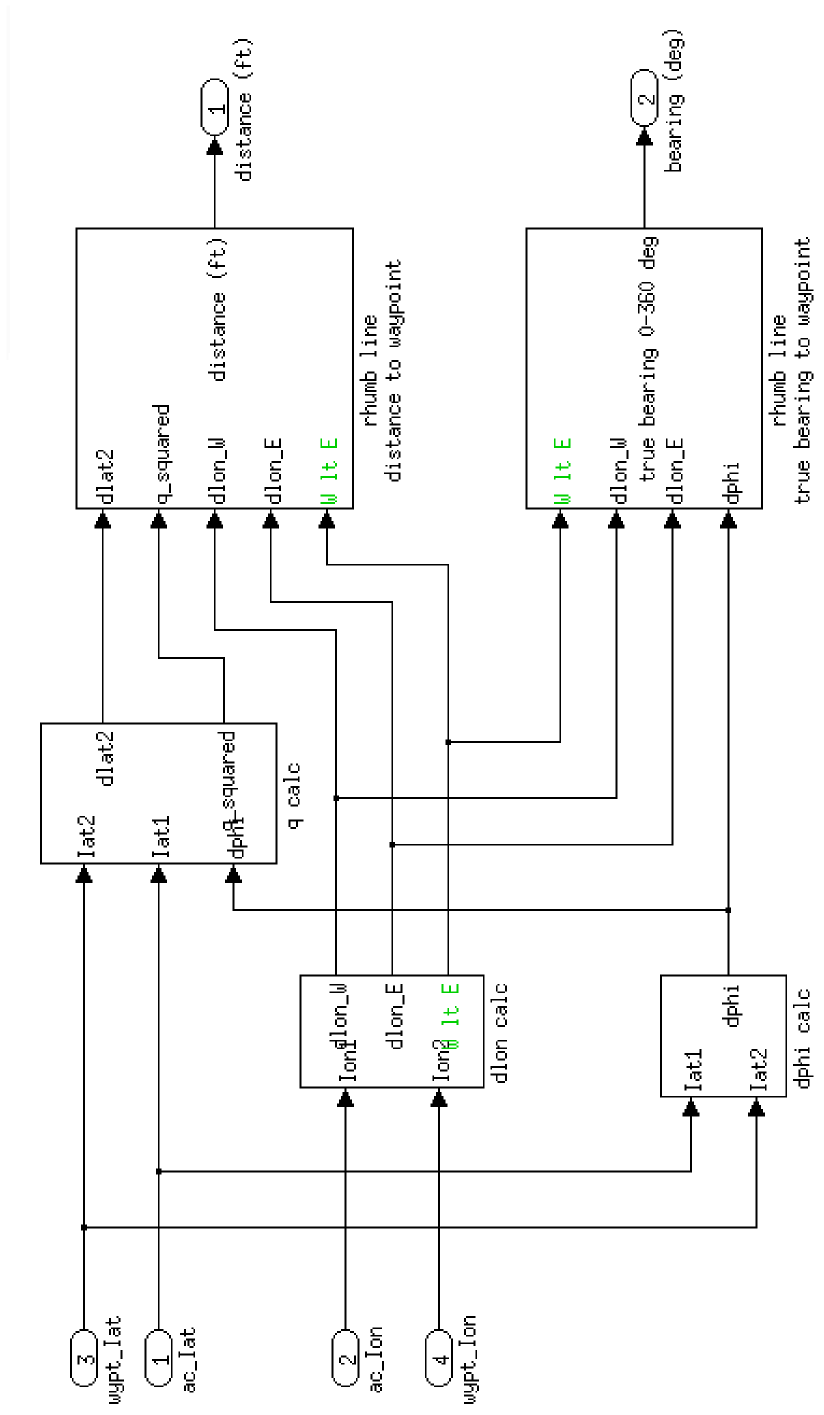

Figure B.8: Rhumb-line controller bearing angle calculation.

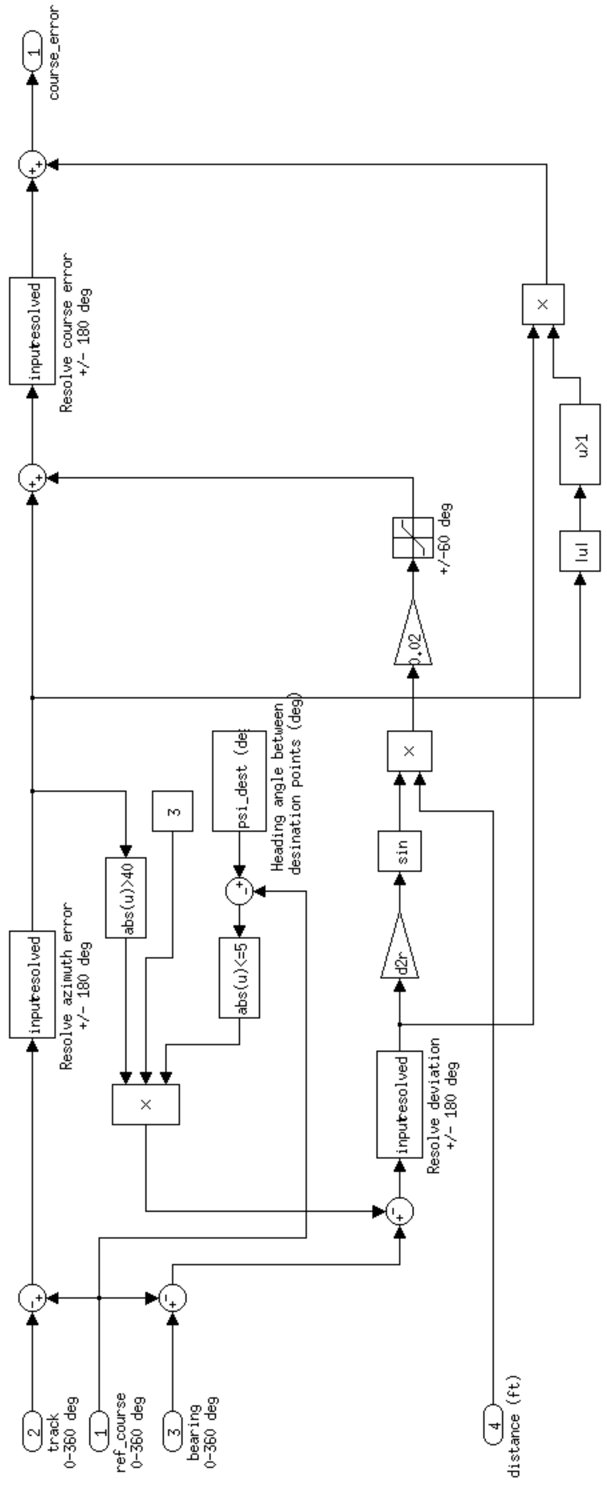

Figure B.9: Rhumb-line controller course error calculation.

B.4 RQ-2 Pioneer Control System

The RQ-2 Pioneer navigation control system is shown in figure B.10. The altitude hold inputs are *altitude*, θ and ϕ. The input to the velocity hold controller is V_T. The outputs of the system are commands to the aileron and elevator control surfaces (δa and δe) and the throttle command (δth).

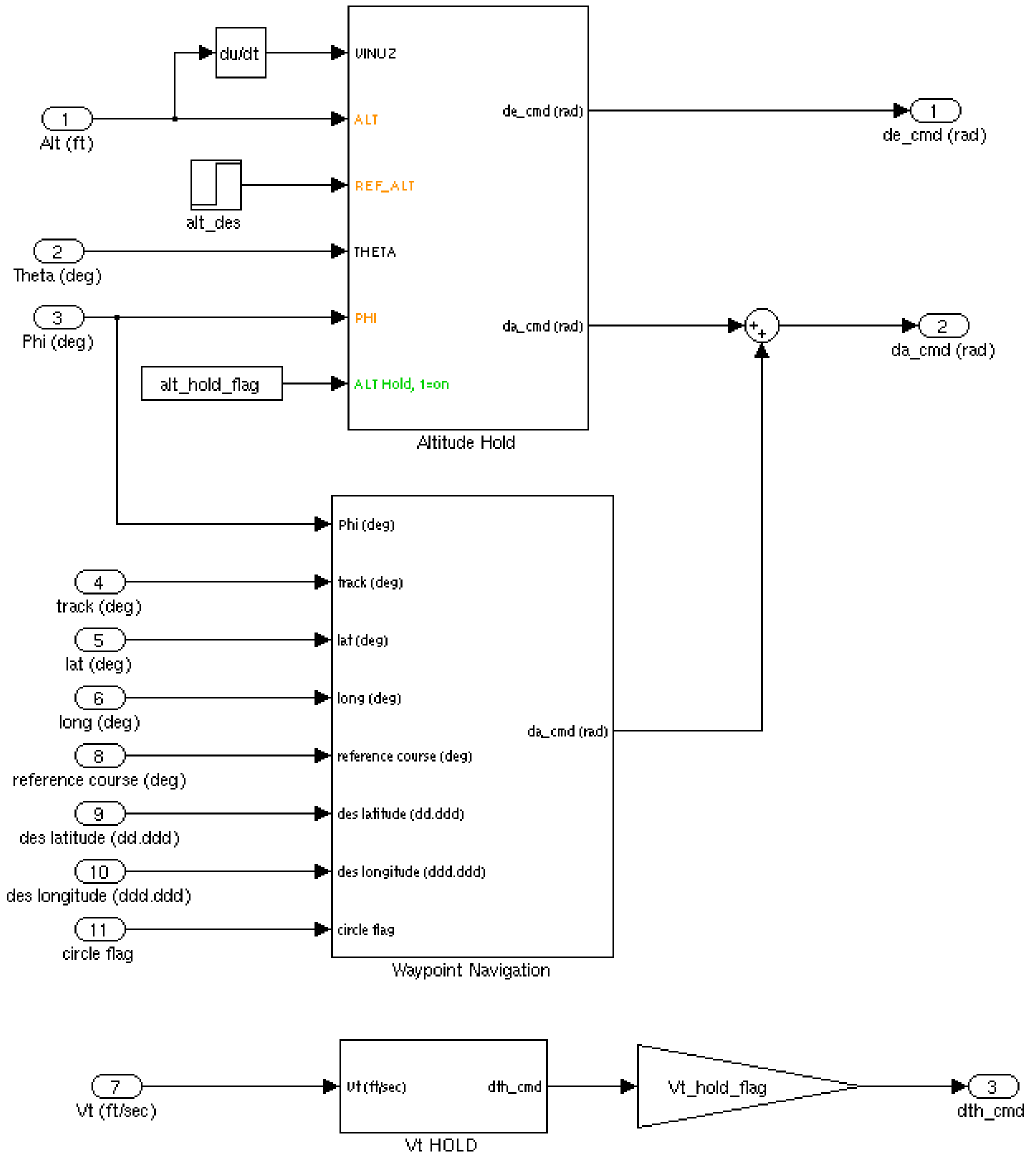

Figure B.10: RQ-2 Pioneer Rhumb-line navigation controller.

B.5 Dubins Aircraft Model

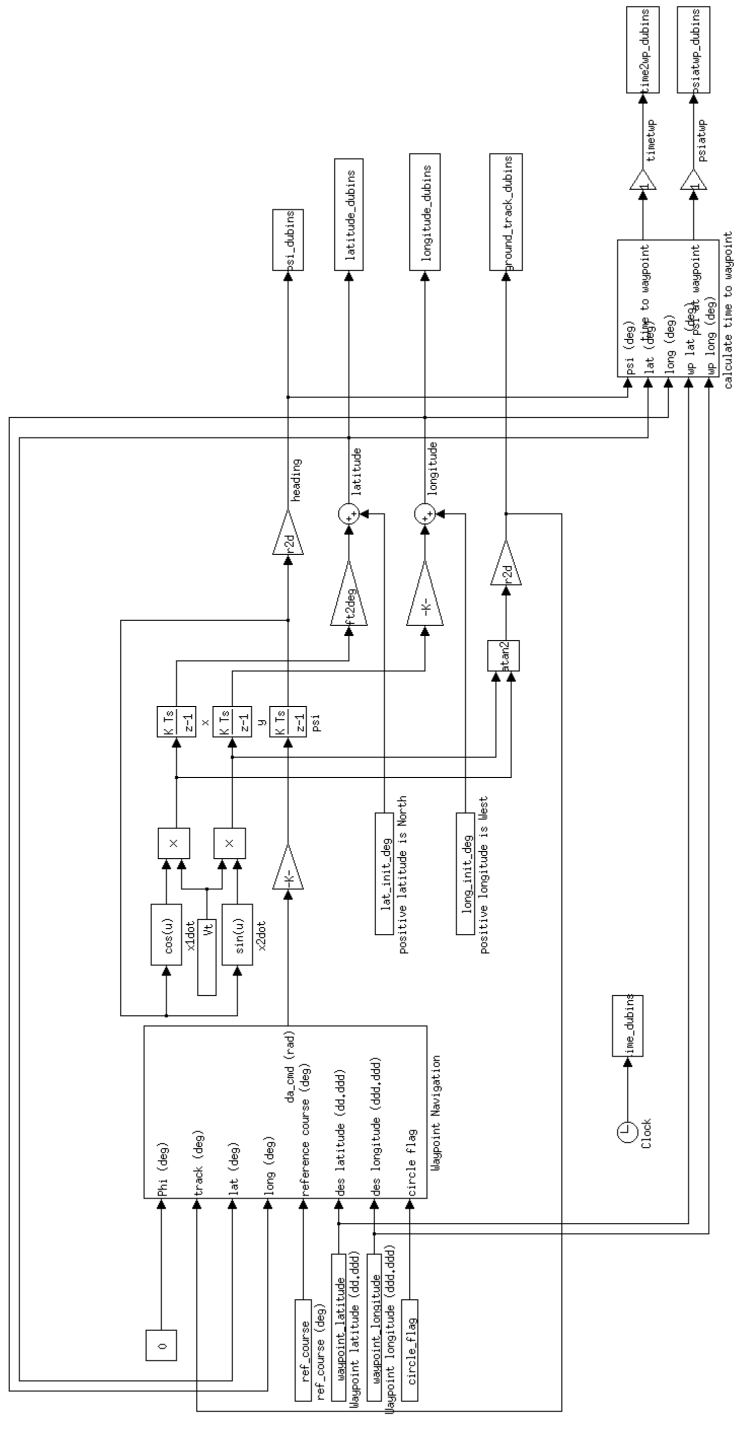

Figure B.11: Dubins aircraft model with Rhumb-line navigation controller. Note: "Waypoint Navigation" is identical to the nonlinear Waypoint Navigation subsystem.

B.6 Sliding Mode Controller

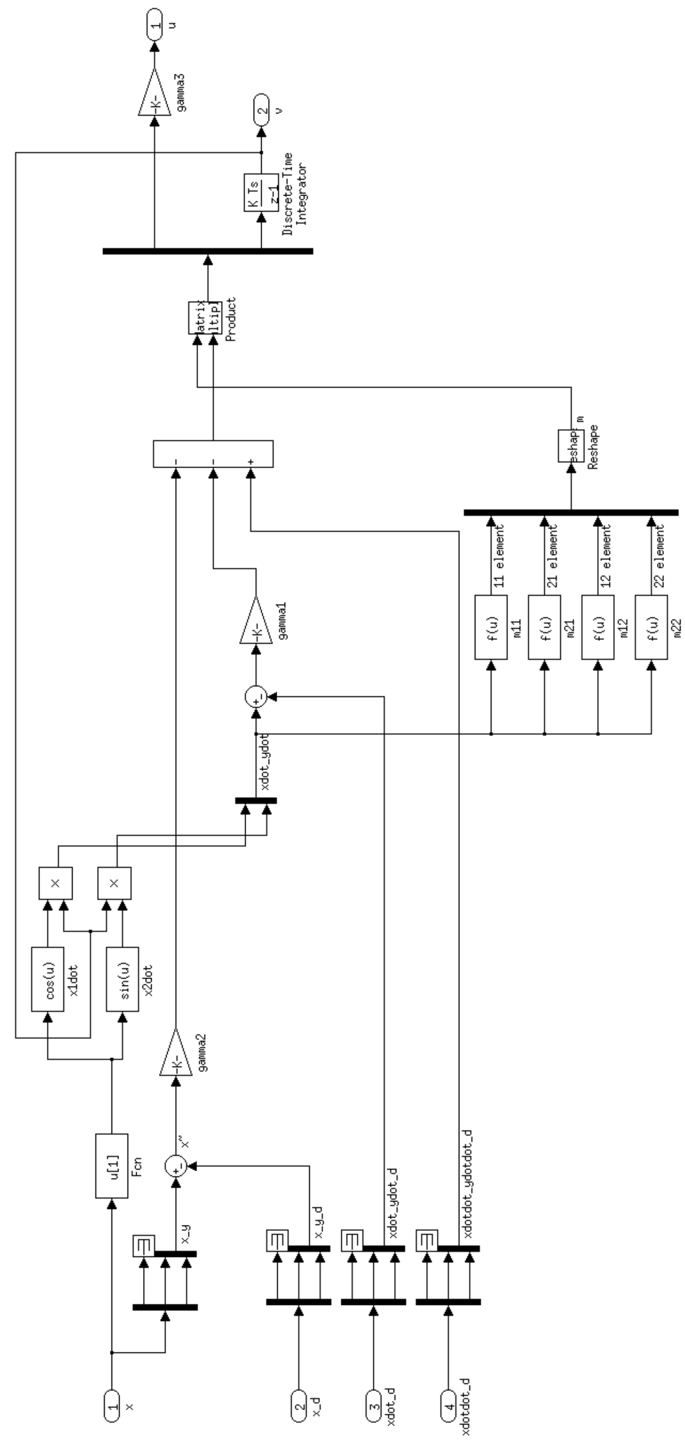

Figure B.12: Dubins aircraft model sliding mode navigation controller.

Appendix C

RQ-2 Handling Qualities

C.1 Additional Plots

Step-Input to Elevator

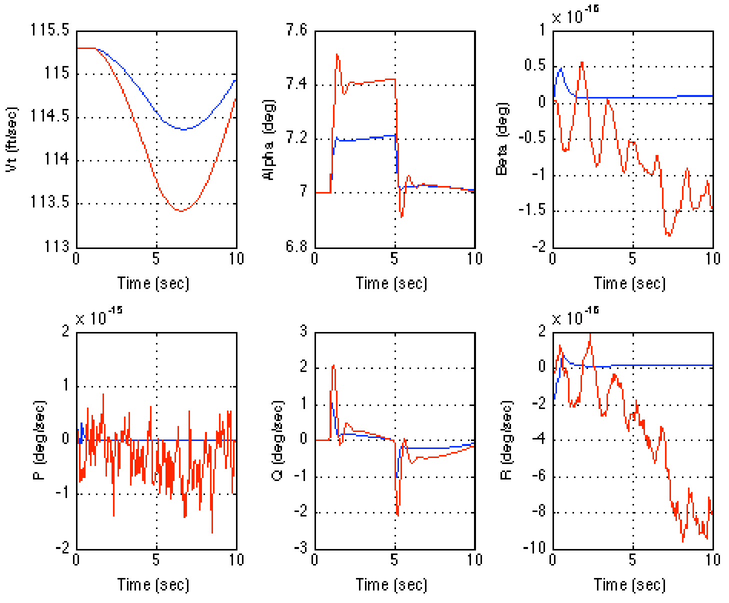

Figure C.1: Orientation and rotation rates for δe elevator step-input. Closed-loop is shown in blue, open-loop in red.

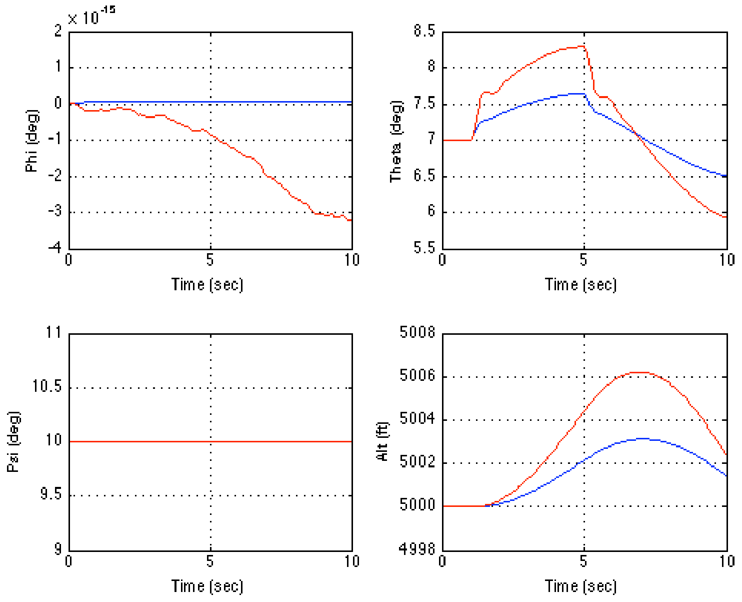

Figure C.2: Orientation and altitude for δe elevator step-input. Closed-loop is shown in blue, open-loop in red.

Rudder Doublet

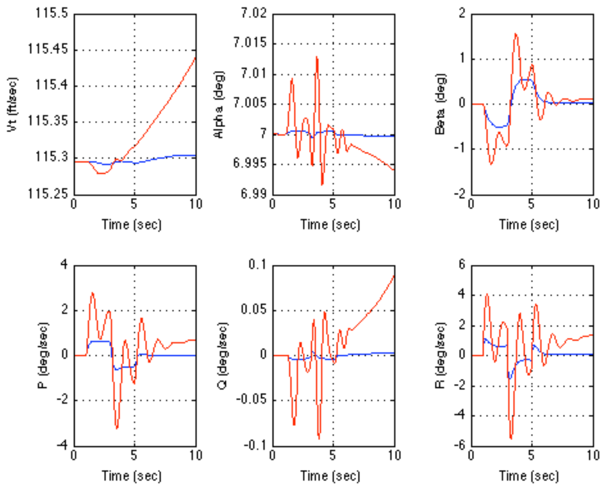

Figure C.3: Orientation and rotation rates for δr rudder doublet. Closed-loop is shown in blue, open-loop in red.

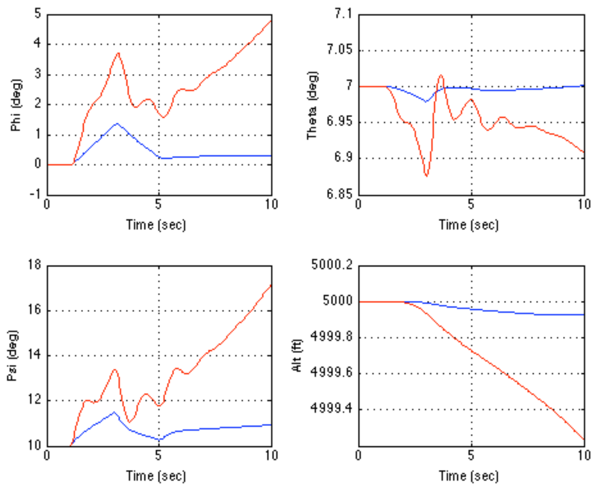

Figure C.4: Orientation and altitude for δr rudder doublet. Closed-loop is shown in blue, open-loop in red.