6-2021

Six Degree-of-Freedom Mission Planning for Reentry Trajectories

Peter Davis

OPTIMAL CONTROL BASED SIX DEGREE-OF-FREEDOM MISSION-PLANNING FOR REENTRY TRAJECTORIES

THESIS

Peter Davis

AFIT-ENY-MS-21-J-097

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENY-MS-21-J-097

OPTIMAL CONTROL BASED SIX DEGREE-OF-FREEDOM MISSION-PLANNING FOR REENTRY TRAJECTORIES

THESIS

Presented to the Faculty

Department of Aeronautical and Astronautical Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Aeronautical Engineering

Peter Davis, BS, BA

May 17, 2021

AFIT-ENY-MS-21-J-097

OPTIMAL CONTROL BASED SIX DEGREE-OF-FREEDOM MISSION-PLANNING FOR REENTRY TRAJECTORIES

THESIS

Peter Davis, BS, BA

Committee Membership:

Major Robert A. Bettinger, Ph.D
Chair

Major Joshuah A. Hess, Ph.D
Member

Dr. Richard G. Cobb
Member

Abstract

Traditional reentry dynamics and planning has typically explored 3 Degrees-of-Freedom (3 DOF) or pseudo 6-DoF problem formulations. This research expands upon previous work and presents a path-constrained optimal control formulation of a fully 6 Degrees-of-Freedom (3 DOF) dynamic system for an unpowered Reentry Vehicle (RV).

In a full 6-DoF dynamic system, the translation, rotation and rotational rates are continually tracked. A system of equations of motion are developed to express the dynamics of the RV in terms of defined states and the RV's physical control deflections. A neural-network is used to approximate the aerodynamic database of an exemplary RV. The resulting highly non-linear dynamic system is generalized such that it could be directly adapted to a given reentry body such as Maneuvering Reentry Vehicle (MaRVs) or a Hypersonic Glide Vehicles (HGV) with appropriate inputs.

This research lays the foundation for the integration of the real control parameters into mission planning. The newly developed equations of motion are verified against existing work. The system of highly non-linear equations and constraints are used to express an optimal control problem that investigates the minimum time and minimum control trajectories that impact a mission specified terminal conditions. Using GPOPS-II, an optimal control profile for each case is found within the specified conditions. It is also demonstrated that the multi-dimensional aerodynamic database can be approximated by a series of Artificial Neural Networks (ANN) with acceptable error bounds.

Soli Deo Gloria.

Acknowledgements

Firstly, I would like to thank Major Robert Bettinger for his persistent support throughout this research process. Your mentorship has helped me immensely, from the beginning of this research three years ago to the final document. I am grateful to Major Joshuah Hess and Dr. Richard Cobb for your scholarly advice and invaluable feedback.

To my team at work: Mark Bellott, Jeremy Suel and Ray Baxter. Your boundless support and patience over the last four years made this possible. You decided to risk hiring and accepted the cost of me taking time-off to work on thesis. I hope I've met your expectations and I will forever be thankful for your help, guidance and inspiration. Alex Freeman: thank you for being a great listener, good friend and an astute, witty proofreader. Also, Jared Augsburger for shining light to the nebulous world of artificial intelligence and machine learning. And to all other members of the Impulse Team: the inspiration I've took from all of you has had a meaningful impact on this research.

My family: Pappa, Amma, Johnu, Kochumol, Ammama, Ammachi and Kunjamma. For their support, guidance, prayers and love towards all these years. Thank you all and this would not have been possible without you all. And to Anu, for your care and words of motivation that kept me from plunging to bowels of lethargy.

-Peter *Valiyaveettil* Davis

Why is it called atmospheric reentry when the vehicle in its whole enters the atmosphere only once?

Table of Contents

Page

List of Figures

List of Tables

OPTIMAL CONTROL BASED SIX DEGREE-OF-FREEDOM
MISSION-PLANNING FOR REENTRY TRAJECTORIES

I. Introduction

1.1 Motivation

Atmospheric reentry is a fascinating problem. It presents complex engineering challenges for manned spaceflight, interplanetary exploration, end-of-life space system disposal, and strategic defense. For any reentry mission, trajectory planning is crucial. In case of manned reentry vehicles such as the Space Shuttle or Apollo capsules, the path of the vehicle to the landing site has to be pre-computed and must ensure the survivability of passengers. Similarly for a missile system, it is critical that a planned path is taken to hit the target at the right conditions.

Conventionally, reentry mission planning is done by commanding an angle-of-attack and bank profile for a desired trajectory. It is assumed that the physical control deflections can easily achieve the guidance commands. In essence, the angle of attack, bank angle or their derivatives are treated as control inputs for mission planning. And very often, the angle of side-slip is neglected. Additionally, the aerodynamic model used in most existing body of research is often simplistic. Most aerodynamic models assume that the coefficients of lift and drag are solely the functions of Mach number and angle of attack. Such a model does not factor in the actual control deflections needed to achieve the said angle of attack and Mach number. [1] [2] [3] [4].

The bulk of the existing studies do not model or track attitude rates. Most studies assume that the Reentry Vehicle (RV) is at trim state during the flight. To be at

"trim" implies that the inertial roll rate, pitch rate and yaw rate are close to zero [5]. In fact the roll, pitch and yaw rates directly influence the moment and often the force coefficients [6] [7].

The following questions were the motivation for the research: How do actual control deflections of an aerodynamic reentry vehicle effect the trajectory of a reentry mission? Is it possible to develop a high-fidelity simulation environment that tracks the attitude and attitude rates of an RV, where the control inputs are the actual control deflections of the system? If so, can the control profile for a given mission be optimized?

1.2 Research Objectives

The goals of this research are two-fold:

1. Build a high-fidelity simulation environment that characterizes translational and rotational trajectory of the RV over the course of reentry.

    - Define state variables and control inputs. The control inputs must be the deflections from the aerodynamic control surfaces of the RV.
    - Incorporate a full six degrees-of-freedom (6-DoF) aerodynamic database is employed to characterize the force and moment coefficients due to said control deflections. An Artificial Neural Network (ANN) will be used to approximate multi-dimensional lookup tables that define aerodynamic coefficients.
    - Develop the model of a highly non-linear and coupled dynamic system that represents the reentry dynamics of a non-thrusting RV.
    - Create a modular simulation environment which can be adapted to emulate any aerodynamically controlled reentry vehicle with a properly defined

aerodynamic database.

2. Use the developed simulation environment to investigate the optimal control profile for a given scenario. GPOPS-II [8] is used to find the control profile that minimizes the flight time from an initial state to a final state.

The Apollo reentry capsules and most early InterContinental Ballistic Missiles had limited aerodynamic control [5]. This is not true in case for most current reentry vehicles. The ability to aerodynamically control the vehicle mid-flight enhances precision, extends range capabilities and helps radar evasion. As we move into the age of high-performance Maneuvering Reentry Vehicles (MaRV) and Hypersonic Glide Vehicles (HGVs), the impact of aerodynamic control surfaces becomes increasingly important in trajectory planning. This research lays the foundation for the integration of the real control parameters into mission planning.

1.3 Assumptions and Limitations

The following assumptions are made to aid the analysis:

- A spherical Earth that rotates at a constant rate.
- Gravity is always directed towards Earth's geometric center. See 2.4.1.
- A model based on Standard Atmosphere 1976 is used to emulate atmospheric effects. Ideal gas assumption is made to compute speed of sound at different altitudes. This is detailed in Section 2.4.2.
- Variations in atmosphere due to wind effects are assumed to be negligible and hence not modeled.
- Controller response dynamics are ignored. In the model, the control demands are achieved instantaneously.

- Guidance and navigation errors are not modeled. It is assumed, for the scope of this study, that the vehicle has accurate knowledge of its state.
- Mass and geometry changes in the RV which occur during reentry due to heating and ablation effects are ignored. See Section 2.5.
- It is also assumed any change in moments or products of inertia due to control deflections is negligible.

1.4 Thesis Overview

Chapter II outlines existing body of relevant research and current work that were leveraged in this study. Chapter III documents the derivation of equations of motion, formulation of the dynamic system, mission constraints and the presentation of the optimal control problem. Chapter IV introduce the vehicle model used for the study, as well as detail the performed analysis and simulation-based results. In Chapter V, results of the study and its significance will be discussed. Finally, the ideas for future research based on this study are laid out.

II. Background and Literature Review

This chapter examines the current body of research and analyze different approaches to solve the problem at hand.

2.1 Reentry Equations of Motion

The equations of motion are the basis to the simulation environment. They can be broadly categorized into two groups. 3 Degrees of Freedom (3-DoF) which only tracks the translation, i.e, position, velocity and acceleration of the reentering body. Such an environment assumes a point-mass, which typically is the center-of-mass for rigid reentry bodies. In a 6 Degrees of Freedom (6-DoF) freedom simulation, the rotational motion of the body is also tracked along with translation. In contrast to the point-mass model, the orientation of the body with respect to the center-of-mass is also tracked.

Early pioneers like Chapman, Loh, and Vinh developed non-dimensional equations of motion that better suited the computational capabilities and limitations of that era [2] [9]. With the growth of computing power and resources available to researchers, there is no longer the need to compromise dimensionality to solve the equations of motion for atmospheric reentry [10]. Novel methods such as dynamic programming, artificial intelligence, and high performance computing are being used to study reentry trajectories subject to design a variety of boundary and path constraints [11] [3].

The early papers from the 1960s formulate the reentry equations of motion in non-dimensional form [9] [2]. The non-dimensional analysis was advantageous as the computing power of that age was limited. Today, with advanced computing capabilities, one could easily afford to model the system with dimensional equations of motion. Dimensional equations are intuitive and easier to troubleshoot. Moreover,

the purpose of this study is mission planning and not on-board guidance. Therefore, the time required for the computation of on-board control is not a limiting factor.

2.2 6-DoF Equations of Motion

Most existing research on reentry dynamics use a 3-DoF model [2] [10] . The attitude of the body is not captured with that approach unlike a truly 6-DoF simulation. A 6-DoF model needs much more a-priori information such as a full suite of aerodynamics. Angle of attack, bank angle or side-slip angle are used as the control inputs for their dynamic system [2] [11] [12] [13]. These parameters will be defined in Chapter III. In this study, it will be shown that angle of attack, bank angle and side-slip angle are functions of the vehicle's state and control deflections. Bollino [5] has derived the rate of change of the three parameters as a function of state values and control inputs for a highly simplified scenario, as described in Section 4.4.1. This study will derive a more rigorous and generalized form of those equations within the assumptions stated in Section 1.3.

2.3 Optimal Control Problem

GPOPS-II, a MATLAB based software, translates the given problem into a Non-Linear Problem that is passed on to a solver such snopt or ipopt to compute the optimal control profile for the specified cost function and path and state constraints [8]. Other programs such as DIDO are also commonly used to solve optimal control problem [5]. Section 3.5 details how the reentry dynamics for the studies cost functions are fed into GPOPS-II.

Novel methods including artificial intelligence and machine learning techniques have been employed to solve optimal control problem. Lee has used a reinforcement learning method to compute minimum time trajectories for a notional hypersonic

glide vehicle in a 3-DoF simulation environment [11].

2.4 Planetary Model

This study assumes a spherical rotating Earth with a radius of 6378.14 kilometers [14]. The axis of rotation is along the planet's North Pole. It is assumed that Earth rotates at a constant rate of 7.292115×10^{-5} radians per second. Any effects due to precession and nutation are negligible in the timespan of a typical reentry [5].

2.4.1 Gravity Model

For a spherical Earth, the gravity always pulls the body towards Earth's geometric center.

$$\mathbf{g} = -\frac{\mu_{\oplus}}{r^{\mathbf{2}}}\mathbf{x_P} \tag{2.4.1}$$

were $\mu_{\oplus}$ is Earth's gravitational parameter equal to $3.986004418 \times 10^{14} \frac{m^3}{s^2}$ [14], r is the radial distance between the vehicle's center of mass and Earth's center, in meters and x_p is the direction of the instantaneous position vector measured from Earth's geometric center.

2.4.2 Atmospheric Model

There are several different models used in literature to estimate Earth's atmosphere. The simplest class of models assume exponentially decaying atmospheric density; such models have are used by Hicks [2] and Buseman et al. [10]. This model is good for preliminary simulations as it is computationally efficient [14]. The 1976 NASA Standard Atmosphere [15] is used to model atmospheric effects in this study. Crassidis and Markeley [14] have described more complex and recent atmospheric models, such as Harris-Priester Model wich attempts to model density in terms of

atmospheric temperature, and the purely empirical GOST atmosphere constructed using data from the Cosmos spacecraft.

Normally in simulations, the 1976 Standard Atmosphere is employed as a lookup table [16]. But for the ease of the optimal control solver to take derivatives, the atmospheric data is approximated as continuous functions of altitude [17]. It is shown that the only relevant data from the 1976 Standard Atmosphere is the atmospheric density and ambient temperature for calculating speed of sound. The adapted atmospheric models and their comparisons to standard models are detailed in subsections 2.4.2.1 and 2.4.2.2.

2.4.2.1 Atmospheric Density

Atmospheric density is typically estimated as a function of geocentric altitude alone. It is common to approximate density with a simple atmospheric model such as Hicks [2]. However, the error of that model relative to Standard Atmosphere can exceed 50% as depicted in Figure 2.4.1. A continuously differentiable atmospheric model was created by adding four Gaussian correction terms to estimate the residual of a basic exponential atmospheric density model. This model predicts density with a maximum error of 10% compared to Standard Atmosphere 1976 (Figure 2.4.1). The adapted atmospheric density ρ is expressed as :

$$
\begin{aligned}
\rho(h) &= \rho_0(h) + \Delta\rho_1(h) + \Delta\rho_2(h) + \Delta\rho_3(h) + \Delta\rho_4(h) \\
\rho_0(h) &= 1.23\ exp(-\frac{h}{7.25}) \\
\Delta\rho_1(h) &= 0.05\ exp\left(-\left(\frac{h-4.2}{3}\right)^2\right) + 0.11\ exp\left(-\left(\frac{h-8.7}{6.4}\right)^2\right) \\
\Delta\rho_2(h) &= 0.7\times10^{-2}\ exp\left(-\left(\frac{h-19}{2.8}\right)^2\right) + 0.2\times10^{-2}\ exp\left(-\left(\frac{h-23}{1.8}\right)^2\right) \\
\Delta\rho_3(h) &= -1.4\times10^{-3}\ exp\left(-\left(\frac{h-37.2}{11.4}\right)^2\right) - 0.27\times10^{-3}\ exp\left(-\left(\frac{h-48.7}{10.3}\right)^2\right) \\
\Delta\rho_4(h) &= -1.2\times10^{-4}\ exp\left(-\left(\frac{h-53.8}{6.4}\right)^2\right) - 0.14\times10^{-4}\ exp\left(-\left(\frac{h-60.2}{6.4}\right)^2\right)
\end{aligned}
\tag{2.4.2}
$$

where h is the geocentric altitude in kilometers and ρ is atmospheric density in kg/m^3. The curve fit was derived for geocentric altitudes from 0 to 80 kilometers. The comparison of the proposed model to Standard Atmosphere 1976 is illustrated in Figure 2.4.1.

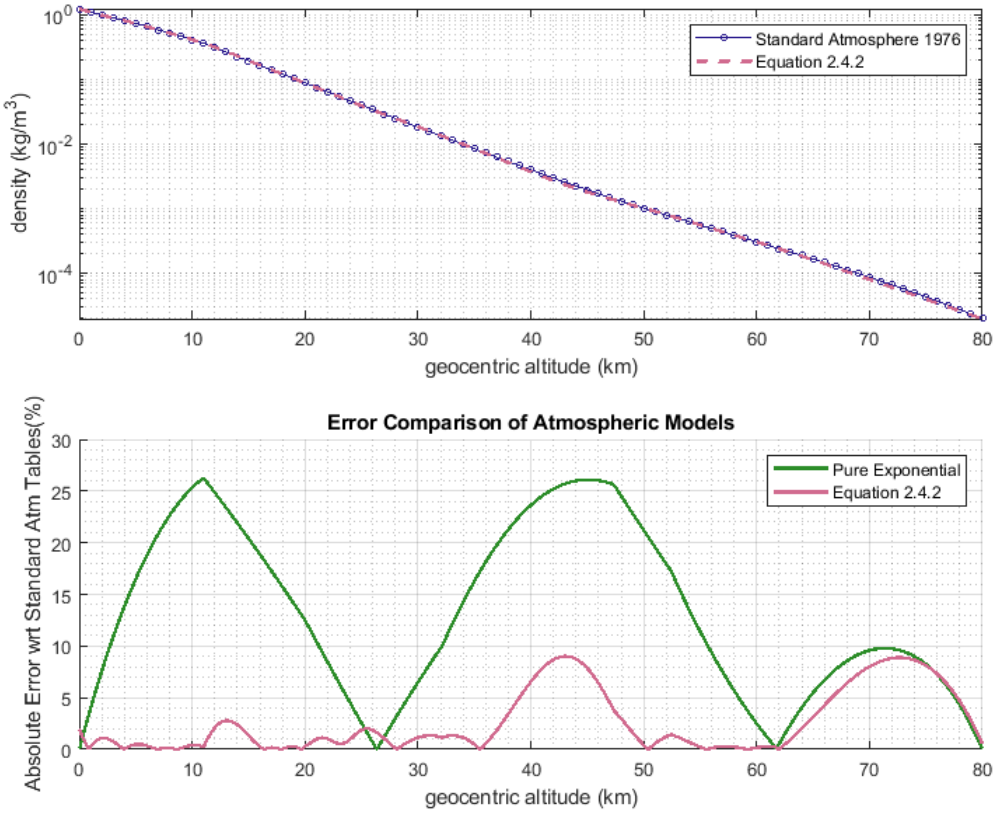

Figure 2.4.1: Atmospheric Model

2.4.2.2 Speed of Sound

The speed of sound is key to compute the Mach number, which dictates the ambient aerodynamics of the vehicle. Speed of sound can be approximated as a function of atmospheric temperature T as :

$$v_S = \sqrt{\tilde{\gamma}\hat{R}T}$$

where $\tilde{\gamma}$ is the ratio of specific heats, $\hat{R}$ is the ideal gas constant and T is the ambient temperature. Using the Standard Atmosphere 1976 tables for atmospheric temper-

ature, the speed of sound is approximated as a fifth-order polynomial function of geocentric altitude.

$$v_s = (3.791 \times 10^{-7})h^5 - (5.8 \times 10^{-5})h^4 + (1.11 \times 10^{-3})h^3 + (0.144)h^2 - (5.4)h + 341.9 \tag{2.4.3}$$

where v_s is the speed of sound in m/s and h is the geocentric altitude in kilometers. When compared with the speed of sound computed from Standard Atmosphere 1976, the absolute value of relative error does not exceed 3% as shown in Figure 2.4.2.

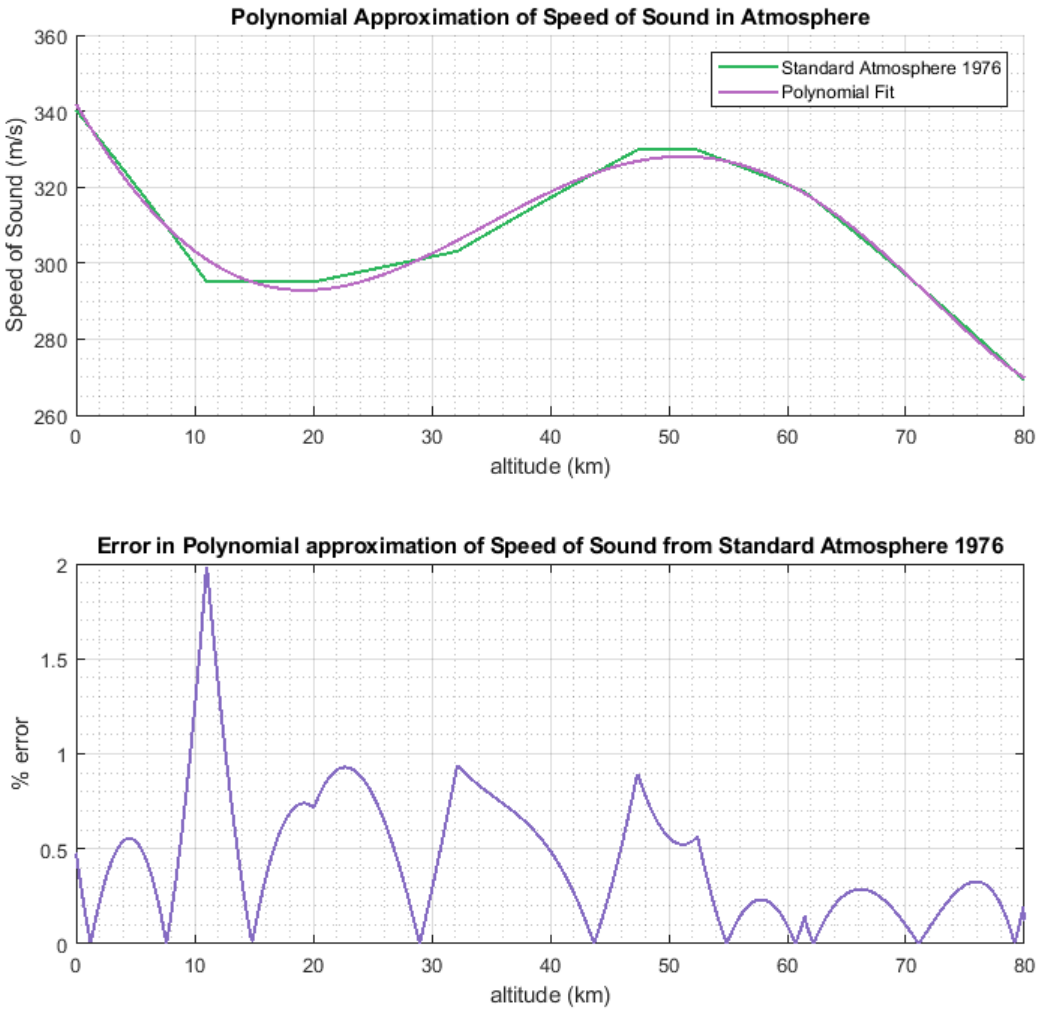

Figure 2.4.2: Speed of Sound Comparisons

2.5 Vehicle Mass and Physical Properties

As the RV descends into Earth at high speed, its mass and geometry change slightly due to heating and ablation. This change is considered negligible for the purpose of this study. Hence, the following physical properties of the vehicle are considered to be non-varying throughout the simulation.

- m, Mass: Change in mass due to heating and ablative effects are assumed to be negligible. Hence the mass is treated to be a constant throughout reentry. Expressed in kilograms.

- Center of Mass: The center of mass is also assumed to be constant throughout reentry.

- **I**, Inertia Tensor : As stated earlier, the mass properties of the reentering body is assumed to be constant. Plus, small control deflections do not affect the products and moments of inertia in any significant way. Hence, the inertia tensor is assumed to be constant. Inertia tensor is a positive definite, symmetric 3-by-3 matrix that contains the moments and products of inertia.

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}$$

  The diagonal terms represent moments of inertia and non-diagonal terms are products of inertia. Each element of the tensor is expressed in kilograms-times-square-meters in SI units. For an axisymmetric vehicle, the products of inertia are zero when expressed in reference to its principal axes [18]. Most aerodynamic vehicles have one plane of symmetry, leading to two of the products being zero. This is explored in Section 3.3.3.

2.6 6-DoF Aerodynamic Database

The aerodynamic database tabulates the force and moment coefficients of an aerodynamic body with respect to Mach number, angle of attack, angle of sideslip and control deflections. The coefficients are computed from wind tunnel measurements or

predicted from numerical simulation software such as MISSILE DATCOM [5]. For the mission planner, it is assumed that a full 6-DoF aerodynamic database for the reentry vehicle is available *a priori*. There are two ways the force coefficients are represented:

- with Respect to the Body Axes: axial, normal and side coefficients are represented along the directions illustrated in Figure 3.2.8. This is the most direct measurement from wind-tunnel data [19].
- with Respect to the Wind Direction: The coefficients are represented as components opposite to the instantaneous velocity vector, along the lift vector, and normal to the lift-drag plane. This system is commonly used for airplanes and cruise missiles [6].

The moment coefficients are almost always measured with respect to the vehicle's center of gravity along the body axes [18] [20]. The aerodynamic coefficients are measured by varying one independent variable while keeping the others constant. Due to the discrete nature of the data, the values of the coefficients within the tabulated points have to be interpolated from the tabulated points in a multi-dimensional sense. The multi-dimensional interpolation is complex and computationally intensive. To alleviate the computational burden, the coefficients can be decomposed into individual increments based on specific independent variables. It is assumed that each increment is independent of the others at all times [6]. An example of such an aerodynamic model is given in Equations 2.6.1 and 2.6.2. The subscript $_0$ refers to base coefficients. They can be estimated as a function of Mach number and angle of attack. Stability and control increments with respect to the independent variables are added to the base coefficients.

$$
\begin{aligned}
C_x &= C_{x_0} + \Delta C_{x_q} + \Delta C_{x_{\delta a}} + \Delta C_{x_{\delta e}} \\
C_y &= C_{y_0} + \Delta C_{y_p} + \Delta C_{y_r} + \Delta C_{y_{\delta e}} + \Delta C_{y_{\delta r}} \\
C_z &= C_{z_0} + \Delta C_{z_q} + \Delta C_{z_{\delta a}} + \Delta C_{z_{\delta e}}
\end{aligned} \tag{2.6.1}
$$

Similarly, the moment coefficients from the aerodynamic database can be summed as:

$$
\begin{aligned}
C_l &= C_{l_0} + \Delta C_{l_p} + \Delta C_{l_r} + \Delta C_{l_{\delta a}} + \Delta C_{l_{\delta r}} \\
C_m &= C_{m_0} + \Delta C_{m_q} + \Delta C_{m_{\delta e}} \\
C_n &= C_{n_0} + \Delta C_{n_p} + \Delta C_{n_r} + \Delta C_{n_{\delta a}} + \Delta C_{n_{\delta r}}
\end{aligned} \tag{2.6.2}
$$

The subscripts p, q, r for stability coefficients refer to roll rate, pitch rate and yaw rate respectively. The control increments, which describe how each increment changes with respect to control deflection are given in terms of effective roll, pitch and yaw deflections $(\delta a, \delta e, \delta r)$. A full list of base stability and control increments used in the model and their dependencies is shown in Tables 2.6.1, 2.6.2 and 2.6.3.

The aerodynamic database contains the full suite of stability and control derivatives from which the force and moment coefficients are computed. The full contents of aerodynamic coefficients in the database is detailed in the following tables:

Table 2.6.1: Base Aerodynamic Coefficients

Coefficient Description	**Notation**	**Function of:**
Base Axial Force Coefficient	C_{x_0}	α, β, M
Base Side Force Coefficient	C_{y_0}	β
Base Normal Force Coefficient	C_{z_0}	α, β, M
Base Roll Moment Coefficient	C_{l_0}	M
Base Pitch Moment Coefficient	C_{m_0}	M
Base Yaw Moment Coefficient	C_{n_0}	M

Table 2.6.2: Stability Increments

Increment Description	**Notation**	**Function of:**
Change in Axial Force due to Pitch Rate	ΔC_{x_q}	α, M, q
Change in Side Force due to Roll Rate	ΔC_{y_p}	α, M, p
Change in Side Force due to Yaw Rate	ΔC_{y_r}	α, M, r
Change in Normal Force due to Pitch Rate	ΔC_{z_q}	α, M, q
Change in Roll Moment due to Roll Rate	ΔC_{l_p}	M, p
Change in Roll Moment due to Yaw Rate	ΔC_{l_r}	M, r
Change in Pitch Moment due to Pitch Rate	ΔC_{m_q}	M, q
Change in Yaw Moment due to Roll Rate	ΔC_{n_p}	M, p
Change in Yaw Moment due to Yaw Rate	ΔC_{n_r}	M, r

Table 2.6.3: Control Increments

Increment Description	**Notation**	**Function of:**
Change in Axial Force due to Roll Deflection	$\Delta C_{x_{\delta a}}$	$\alpha, M, \delta a$
Change in Axial Force due to Pitch Deflection	$\Delta C_{x_{\delta e}}$	$\alpha, M, \delta e$
Change in Side Force due to Pitch Deflection	$\Delta C_{y_{\delta e}}$	$\alpha, M, \delta e$
Change in Side Force due to Yaw Deflection	$\Delta C_{y_{\delta r}}$	$\alpha, M, \delta r$
Change in Normal Force due to Roll Deflection	$\Delta C_{z_{\delta a}}$	$\alpha, M, \delta a$
Change in Normal Force due to Pitch Deflection	$\Delta C_{z_{\delta e}}$	$\alpha, M, \delta e$
Change in Roll Moment due to Roll Deflection	$\Delta C_{l_{\delta a}}$	$M, \delta a$
Change in Roll Moment due to Yaw Deflection	$\Delta C_{l_{\delta r}}$	$M, \delta r$
Change in Pitch Moment due to Pitch Deflection	$\Delta C_{m_{\delta e}}$	$M, \delta e$
Change in Yaw Moment due to Roll Deflection	$\Delta C_{n_{\delta a}}$	$M, \delta a$
Change in Yaw Moment due to Yaw Deflection	$\Delta C_{n_{\delta r}}$	$M, \delta r$

Equations 2.6.1 and 2.6.2 is one example of a typical aerodynamic model. Based on the mission requirements and vehicle properties, the aerodynamic model can be expanded or simplified. Examples of aerodynamic models with varying complexities can be found in [6]. The aerodynamic model used for this study is described in Equation 3.6.1.

2.7 Artificial Neural Networks (ANN)

Section 2.6 illustrates that a high-fidelity aerodynamic database involves summing coefficient contributions from multiple table look-ups and cumulating each increments to find respective coefficients. This is a tedious process for a human, as well as a computer. In the early days of this research, GPOPS failed to converge to a solution when look-up tables were used for computing aerodynamic data. Moreover, Masternak has

reported the inefficiency of using look-up tables for problems that use Optimization problems. It is not guaranteed that the derivative of the discrete lookup table data is continuous. Most gradient based solvers are incompatible with discontinuous derivatives [17].

Approximating the speed of sound and atmospheric density as a continuous function of their inputs in Section 2.4.2 was simple as each of them only involved one independent variable. However, for a 6-DoF aerodynamic database that involves almost a dozen independent variables for each coefficient, more advanced methods are to be used for function-approximation.

An Artificial Neural Network (ANN), in its basic form, is an aggregate of functions that emulates the mapping of inputs to outputs in a "truth" data set. A neural network can be divided into an input layer, an output layer and a number of hidden layers specified by the user. Each layer contains a number of nodes. Each input and output of the problem represents the node in their respective layers. The number of nodes in each of the hidden layers is typically specified by the user informed from problem specifications. Each node has an activation function associated with it. The activation function defines the how the node is interconnected to the other nodes based on weights and biased applied to it. A learning algorithm is used to compute the values of weights and biased at each node that approximates the training data within specified uncertainty levels [21] [22].

2.8 Reentry Heat Flux

During the course of reentry, a large amount of energy is dissipated as heat. The shock wave created by a RV moving at hypersonic speeds causes extremely high temperatures that dissociate air molecules into ions [23]. This has a direct effect on the design considerations for the survivability of the airframe. Hence, it is preferable

to keep the heat flux below a desired limit determined by the structural engineers. For the purpose of mission planning, it is desirable to express the heat flux as a function of ambient conditions, the vehicle's speed and its geometry. For a blunt body, the maximum heat transfer occurs at the stagnation point [13]. Hence, an estimation of heat flux at stagnation point is a conservative measure of the heat experienced by the airframe.

It can be safely assumed that the heat transfer is solely due to convective effects. For most vehicles the heat flux due to radiative effects is considered negligible [24]. Finke [25] has formulated an empirical relationship between the vehicle's geometry, free stream conditions and convective heat rate. The relationship, converted to SI units, can be stated as follows:

$$Q = \frac{5.75 \times 10^{-5}}{\sqrt{R_n}} \sqrt{\frac{\rho}{\rho_0}} v^{3.15} \tag{2.8.1}$$

where ρ_0 is the atmospheric density at sea level, ρ is the instantaneous atmospheric density, R_n is radius of the curvature at the stagnation point (nose radius) specified in meters and v is the air relative speed in m/s; the heat flux is given in W/m^2.

2.9 Chapter Conclusion

Based on the works mentioned in Sections 2.1 and 2.2, the equations of motion that characterize an unpowered reentry trajectory are developed in Chapter III. Models described in Section 2.4 is used to characterize gravity and atmospheric effects. Subsequently, a high-fidelity aerodynamic model of an RV is defined using the concepts from Section 2.6 in Section 3.6. A series of ANNs are used to estimate the aerodynamic database. The final dynamic system is translated into an optimal control problem with define states and path constraints including Equations 2.8.1.

III. Research Methodology

3.1 Chapter Overview

In this chapter, the high-fidelity dynamic model of the non-thrusting reentry vehicle (RV) is defined. The chapter begins with establishing different reference frames of interest. The relationships between the reference frames is used to derive the 6-DoF equations of motion. The dynamic model is expressed as highly non-linear functions of states and the control deflections. Later in the chapter, the optimal control problem is stated based on the dynamic model and defined constraints. Finally, the settings used for the optimal control solver are outlined.

3.2 Reference Frames

The dynamics of the reentry problem is described using the following reference frames.

- Earth Centered Inertial (ECI)
- Earth Centered Earth Fixed (ECEF)
- Vehicle Pointing Reference Frame (VPF)
- Flight Path Reference Frame (FPF)
- Aerodynamic Reference Frame (ARF)
- Body Reference Frame (BRF)

This section defines each of the named frames and describes their relationship with each other.

3.2.1 Earth Centered Inertial Reference Frame (ECI)

This is an inertial frame fixed at the center of the Earth. The x-axis, represented by unit vector $\mathbf{x_I}$, points to the intersection of the equator and prime meridian (0° latitude, 0° longitude) at epoch. Epoch for this study is defined at the start of the simulation. Unit vector $\mathbf{z_I}$ is aligned with the axis of Earth's rotation [12]. The y-axis completes a right-handed coordinate system : $\mathbf{y_I} = \mathbf{z_I} \times \mathbf{x_I}$, and is illustrated in Figure 3.2.1.

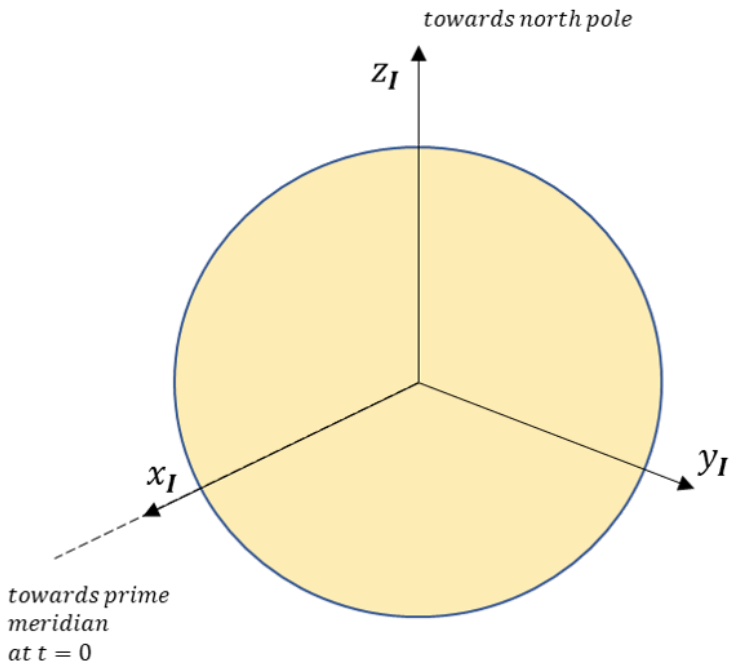

Figure 3.2.1: ECI Reference Frame. Vectors $\mathbf{x_I}$ and $\mathbf{y_I}$ span the equatorial plane.

3.2.2 Earth Centered Earth Fixed Reference Frame (ECEF)

ECEF is a non-inertial frame that rotates with the planet. The frame originates at Earth's center with $\mathbf{z_E}$ pointing along the direction of Earth's rotation. The frame's x-axis passes through 0° latitude 0° longitude at all times. As before, the y-axis completes a right handed coordinate system : $\mathbf{y_E} = \mathbf{z_E} \times \mathbf{x_E}$. Let Earth's rotation

rate be $\omega_\oplus$. A vector in ECI frame is related to ECEF frame by a rotation along the common z-axis by $\omega_\oplus \Delta t$ where Δt is the time that has passed since epoch. The transformation of a vector $\mathbf{u_I}$ in ECI frame to $\mathbf{u_E}$, its expression in ECEF frame, can be formulated as follows:

$$\mathbf{u_E} = \mathbf{T_I^E u_I} = \begin{bmatrix} \cos(\omega_\oplus \Delta t) & \sin(\omega_\oplus \Delta t) & 0 \\ -\sin(\omega_\oplus \Delta t) & \cos(\omega_\oplus \Delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{u_I} \tag{3.2.1}$$

The angular velocity of frame ECEF with respect to ECI frame can be expressed in ECI frame as

$$\omega_\mathbf{I}^{\mathbf{E/I}} = \omega_\oplus \mathbf{z_I} = \omega_\oplus \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.2.2}$$

Figure 3.2.2 depicts the relationship between frames ECI and ECEF. In the image, Vectors $\mathbf{x_I}$,$\mathbf{y_I}$,$\mathbf{x_E}$ and $\mathbf{y_E}$ spans the equatorial plane. Axes $\mathbf{z_I}$ and $\mathbf{z_E}$ are collinear.

3.2.3 Vehicle Pointing Frame (VPF)

The Vehicle Pointing Reference Frame, as defined by Hicks [2], is a non-inertial frame centered at Earth's geometric center. The x-axis $\mathbf{x_P}$ points toward the current position of the RV. The y-axis is constructed such that it is parallel to the equatorial plane and z-axis forms a right-handed coordinate system

$$\begin{aligned} \mathbf{x_P} &= \frac{\mathbf{r_E}}{||\mathbf{r_E}||} \\ \mathbf{y_P} &= \mathbf{z_E} \times \mathbf{x_P} \\ \mathbf{z_P} &= \mathbf{x_P} \times \mathbf{y_P} \end{aligned} \tag{3.2.3}$$

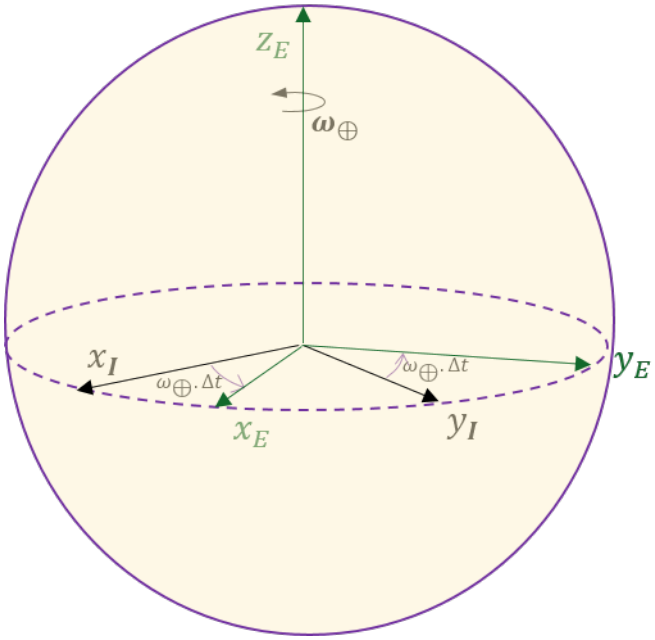

Figure 3.2.2: Illustration of ECEF Reference Frame.

where r_E is the position vector expressed in terms of ECEF frame. The frame can also be described as a rotation along $\mathbf{z_E}$ by the longitude μ followed by a rotation of λ about $-\mathbf{y_P}$. Thus, the transformation of a vector $\mathbf{u_E}$ in ECI frame to $\mathbf{u_P}$ in Vehicle Pointing Frame can be expressed as:

$$\begin{aligned}
\mathbf{u_P} &= T_E^P \mathbf{u_E} = \mathbf{R_y}(-\lambda)\mathbf{R_z}(\mu)\mathbf{u_E} \\
&= \begin{bmatrix} cos(\lambda) & 0 & sin(\lambda) \\ 0 & 1 & 0 \\ -sin(\lambda) & 0 & cos(\lambda) \end{bmatrix} \begin{bmatrix} cos(\mu) & sin(\mu) & 0 \\ -sin(\mu) & cos(\mu) & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{u_E} \\
&= \begin{bmatrix} cos(\lambda)cos(\mu) & cos(\lambda)sin(\mu) & sin(\lambda) \\ -sin(\mu) & cos(\mu) & 0 \\ -sin(\lambda)cos(\mu) & -sin(\lambda)sin(\mu) & cos(\lambda) \end{bmatrix} \mathbf{u_E}
\end{aligned} \tag{3.2.4}$$

In Figure 3.2.3, vectors $\mathbf{x_E}$, $\mathbf{y_E}$ and $\mathbf{y_P}$ are contained in the equatorial plane. $\mathbf{x_P}$ points towards the vehicle position and $\mathbf{z_P}$ completes the right-handed triad. The

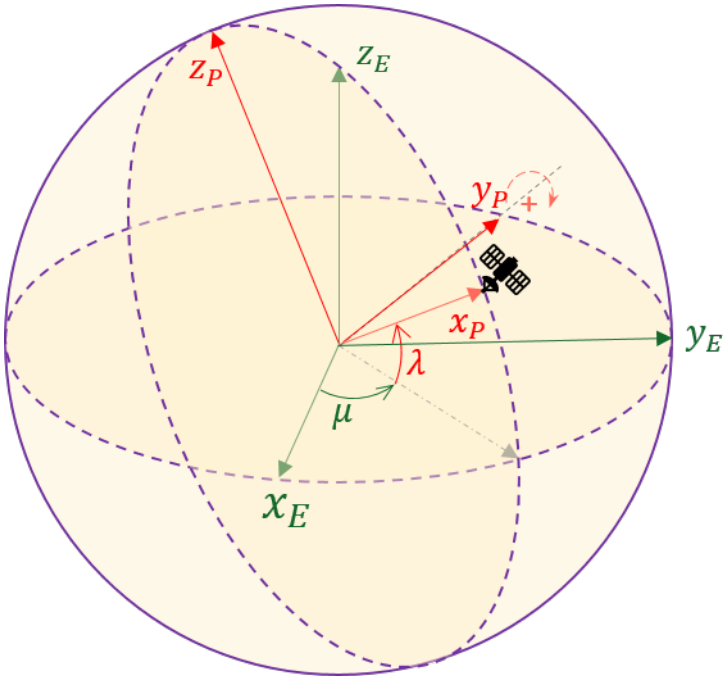

Figure 3.2.3: The Vehicle Pointing Frame (VPF).

angular velocity of the vehicle pointing frame with respect to ECEF expressed in ECEF is:

$$
\begin{aligned}
\omega^{\mathbf{P/E}} &= -\dot{\lambda}\mathbf{y_P} + \dot{\mu}\mathbf{z_E} \\
\omega_{\mathbf{E}}^{\mathbf{P/E}} &= -\dot{\lambda}T_P^E \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \dot{\mu} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\
&= -\dot{\lambda} \begin{bmatrix} -sin(\mu) \\ cos(\mu) \\ 0 \end{bmatrix} + \dot{\mu} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\
\omega_{\mathbf{E}}^{\mathbf{P/E}} &= \begin{bmatrix} \dot{\lambda}sin(\mu) \\ -\dot{\lambda}cos(\mu) \\ \dot{\mu} \end{bmatrix}
\end{aligned}
\tag{3.2.5}
$$

$\mathbf{y_P}$ is parallel to the equatorial plane so that it points east and $\mathbf{z_P}$ points North at every instant in time. The plane formed by $\mathbf{y_P}$ and $\mathbf{z_P}$ can be interpreted as the local horizontal plane. Similarly, the plane formed by $\mathbf{x_P}$ and $\mathbf{z_P}$ defines the local vertical plane, parallel to the vehicle position at each time.

3.2.4 Flight-Path Frame (FPF)

Flight-Path Frame is a non-inertial frame originating from the center of mass of the vehicle. The x-axis points along the earth-relative velocity vector $\mathbf{v_R}$. The y-axis is defined to be normal to the plane that contains the ECEF position and velocity vectors. The z-axis completes the right-handed coordinate system. Note that both the position and velocity vectors lie in the x-z plane formed by the Flight Path Frame.

$$\begin{aligned} \mathbf{x_F} &= \frac{\mathbf{v_R}}{||\mathbf{v_R}||} \\ \mathbf{y_F} &= \mathbf{x_P} \times \mathbf{x_F} \\ \mathbf{z_F} &= \mathbf{x_F} \times \mathbf{y_F} \end{aligned} \tag{3.2.6}$$

To transform a given vector $\mathbf{u_P}$ from VPF to FPF, it is preferable to first re-align the axes $\{\mathbf{x_P}, \mathbf{y_P}, \mathbf{z_P}\}$ such that the new axis $\mathbf{z'_P}$ points against the gravity vector, $\mathbf{x'_P}$ points downrange and $\mathbf{y'_P}$ points to the crossrange direction [12]. This can be accomplished by a rotation about axis $\mathbf{y_P}$ by an angle of $\frac{\pi}{2}$ radians followed by a rotation about $\mathbf{z'_P}$ by the same angle [12].

$$\begin{aligned} \mathbf{u_{P'}} = T_P^{P'}\mathbf{u_P} &= \mathbf{R_z}\left(\frac{\pi}{\mathbf{2}}\right)\mathbf{R_y}\left(\frac{\pi}{\mathbf{2}}\right)\mathbf{u_P} \\ &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}\mathbf{u_P} \end{aligned} \tag{3.2.7}$$

The intermediate frame is depicted in Figure 3.2.4. Here, $\mathbf{z'_P}$ is the radial position vector. Axes $\mathbf{x'_P}$ and $\mathbf{y'_P}$ are contained in the local horizontal plane. The vector $\mathbf{x'_P}$ is the intersection of the equatorial and local horizontal planes.

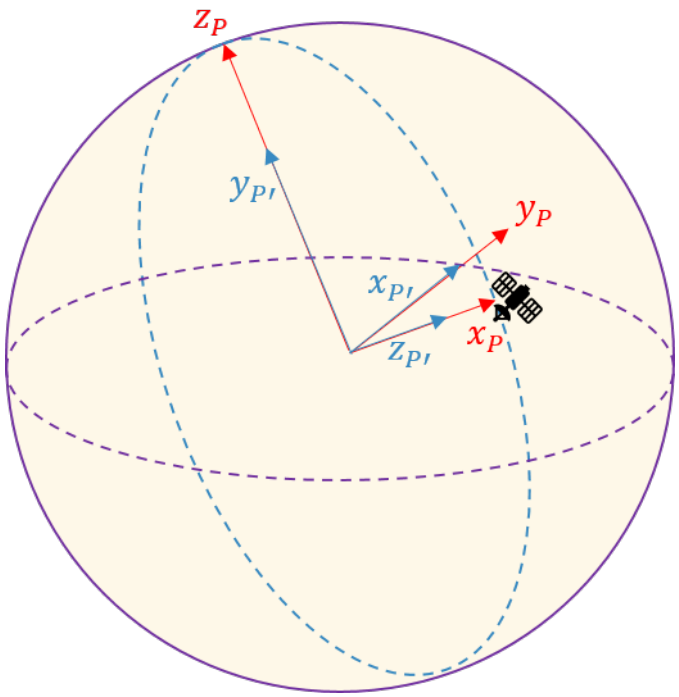

Figure 3.2.4: The intermediate frame.

In the intermediate frame, the local-horizontal plane is spanned by the vectors $\mathbf{x_{P'}}$ and $\mathbf{y_{P'}}$. The heading angle ψ describes the orientation of the velocity vector with respect to the equatorial plane. Vector $\mathbf{v_h}$ is the projection of the velocity vector into the local horizontal plane. Recall that the vector $\mathbf{y_P}$ is contained in the same plane, which by definition is parallel to the equatorial plane. From Equation 3.2.7, it can be deduced that $\mathbf{y_P}$ and $\mathbf{x_{P'}}$ are parallel. Therefore, the angle between $\mathbf{v_h}$ and $\mathbf{x_{P'}}$ is the heading angle. First, rotate $\mathbf{x_{P'}}$ by ψ so that the new $\mathbf{x_{P''}}$ axis aligns with $\mathbf{v_h}$. At this point the velocity vector is entirely contained in the plane defined by $\mathbf{x_{P''}}$ and $\mathbf{z_{P''}}$. Now rotate about $\mathbf{y_{P''}}$ so that the new x-axis $\mathbf{x_F}$ is aligned with the velocity vector by an angle of γ. γ, known as the flight-path angle [20], measures

the difference between velocity vector and the local-horizontal plane. Note that the aforementioned is a negative rotation about axis $\mathbf{y_{P''}}$ by angle γ. The series of Euler rotations from the intermediate frame to flight-path frame is illustrated in Figure 3.2.5. In this figure, the shaded region represents the local horizontal plane. Axis $\mathbf{z_{P''}}$ protrudes outwards in the left image. In the right, axis $\mathbf{y_F}$ goes into the page and is contained in the local horizontal plane.

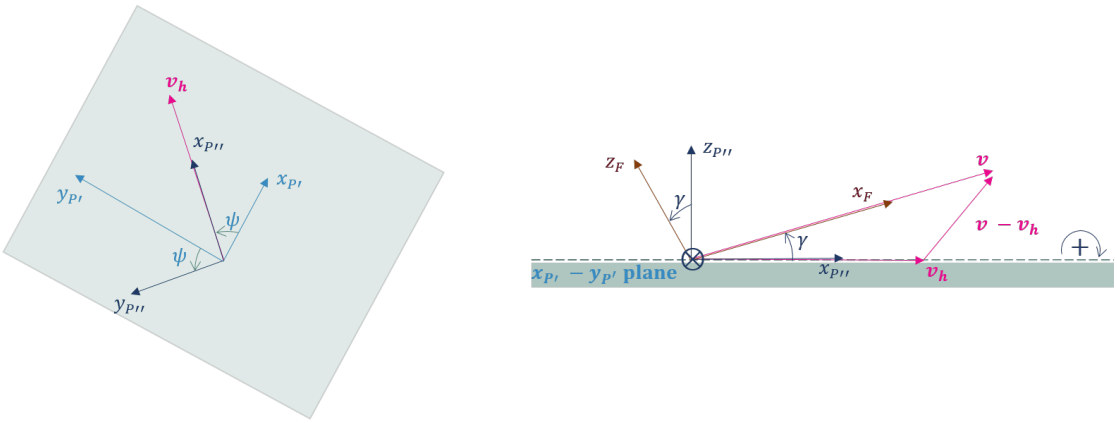

Figure 3.2.5: The path from intermediate frame to FPF.

Using the definitions of γ and ψ, the transformation of vector $\mathbf{u_P}$ in VPF to its equivalent $\mathbf{u_F}$ in FPF can be expressed as:

$$\begin{aligned}\mathbf{u_F} &= T^F_{P'}\mathbf{u_{P'}} = \mathbf{R_y}(-\gamma)\mathbf{R_z}(\psi)\mathbf{u_{P'}} \\ &= \begin{bmatrix} cos(\gamma) & 0 & sin(\gamma) \\ 0 & 1 & 0 \\ -sin(\gamma) & 0 & cos(\gamma) \end{bmatrix} \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{u_{P'}}\end{aligned}$$

$$
\begin{aligned}
\mathbf{u_F} &= T_P^F \mathbf{u_P} = \mathbf{T_{P'}^F T_P^{P'}} \mathbf{u_P} \\
&= \begin{bmatrix} cos(\gamma) & 0 & sin(\gamma) \\ 0 & 1 & 0 \\ -sin(\gamma) & 0 & cos(\gamma) \end{bmatrix} \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \mathbf{u_P} \\
&= \begin{bmatrix} sin(\gamma) & cos(\gamma)cos(\psi) & cos(\gamma)sin(\psi) \\ 0 & -sin(\psi) & cos(\psi) \\ cos(\gamma) & -sin(\gamma)cos(\psi) & -sin(\gamma)sin(\psi) \end{bmatrix} \mathbf{u_P}
\end{aligned}
\tag{3.2.8}
$$

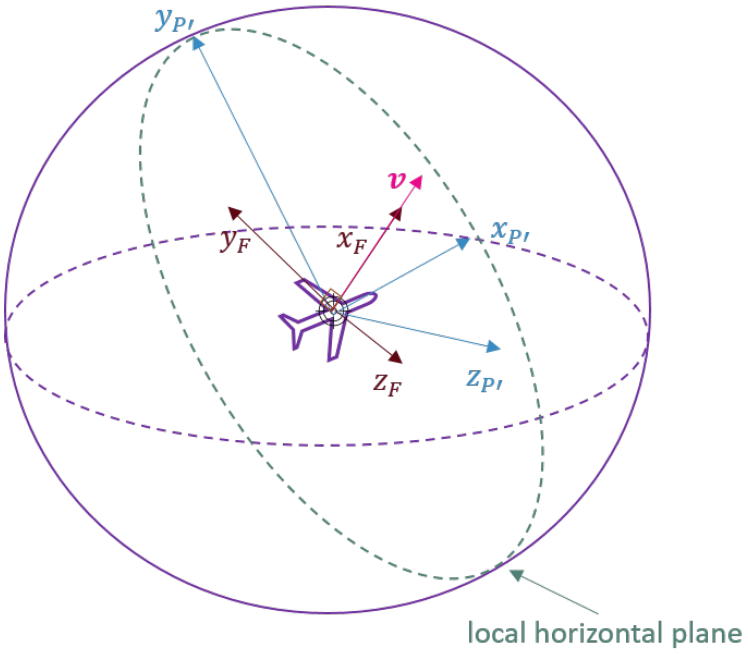

Figure 3.2.6: The Flight-Path Frame. $\mathbf{x_F}$ points towards the relative velocity vector.

The angular velocity of Flight-Path Frame with respect to VFP can be expressed

as:

$$\begin{aligned}
\omega^{\mathbf{F/P}} &= -\dot{\gamma}\mathbf{y_F} + \dot{\psi}\mathbf{z_{P'}} \\
\omega_{\mathbf{P}}^{\mathbf{F/P}} &= -\dot{\gamma}T_F^P \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \dot{\psi}T_{P'}^P \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\
\omega_{\mathbf{P}}^{\mathbf{F/P}} &= \begin{bmatrix} \dot{\psi} \\ \dot{\gamma}sin(\psi) \\ -\dot{\gamma}cos(\psi) \end{bmatrix}
\end{aligned} \tag{3.2.9}$$

3.2.5 Aerodynamic Reference Frame (ARF)

In the previous section, the flight-path frame was established where the x-axis $\mathbf{x_F}$ is aligned to the vehicle's velocity vector, $\mathbf{y_F}$ is normal to the plane that contains both position and velocity vector. It is known that the total lift vector acts normal to the direction of the vehicle's velocity [20]. Hence, the total lift vector is completely contained in the plane spanned by $\mathbf{y_F}$ and $\mathbf{z_F}$. The Aerodynamic Reference Frame (ARF) is constructed by rotating $\mathbf{y_F}$ and $\mathbf{z_F}$ around $\mathbf{x_F}$ by the bank angle so that the the z-axis of the new frame is aligned with the total lift vector, as shown in Figure 3.2.7.

The bank angle defines the orientation of the lift vector with respect to the r-v plane: the plane spanned by the instantaneous position and velocity vectors. The bank angle, denoted by σ, is measured from axis $+\mathbf{z_F}$ to the total lift vector. A positive σ is a counter-clockwise rotation in $\mathbf{y_F}$-$\mathbf{z_F}$ plane. Hence a negative rotation around axis $\mathbf{x_F}$ will align the z-axis of the new frame with the lift vector. The

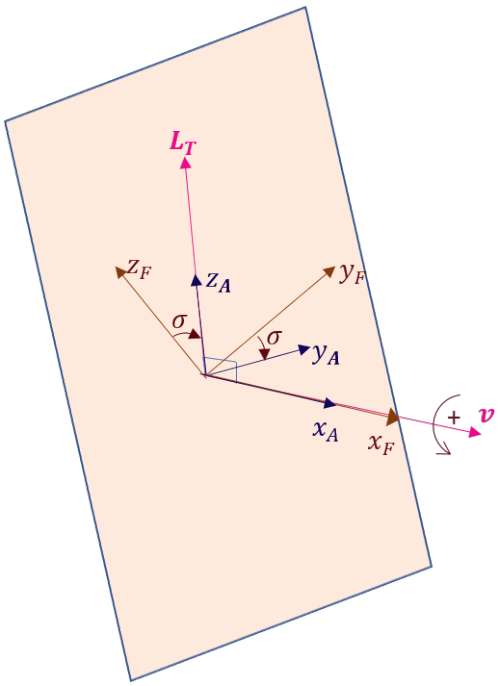

Figure 3.2.7: Aerodynamic Reference Frame. $\mathbf{L_T}$ is the total lift vector.

transformation from FPF to ARF can be expressed as follows:

$$\begin{aligned}\mathbf{u_A} &= T_F^A \mathbf{u_F} = \mathbf{R_x}(-\sigma)\mathbf{u_F} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\sigma) & -sin(\sigma) \\ 0 & sin(\sigma) & cos(\sigma) \end{bmatrix} \mathbf{u_F}\end{aligned} \tag{3.2.10}$$

The expression for angular velocity of ARF in Flight-Path Reference Frame is:

$$\begin{aligned}\omega^{\mathbf{A/F}} &= -\dot{\sigma}\mathbf{x_F} \\ \omega_{\mathbf{F}}^{\mathbf{A/F}} &= \begin{bmatrix} -\dot{\sigma} \\ 0 \\ 0 \end{bmatrix}\end{aligned} \tag{3.2.11}$$

During a planar reentry, it is assumed that there are no external accelerations acting

out of the $r - v$ plane. In that case, the bank angle σ is either 0 or 180 degrees.

3.2.6 Body Reference Frame (BRF)

BRF is a non-inertial reference frame positioned at the vehicle's center of mass. The x-axis points towards the vehicle's nose. The x-z axis of the Body Reference Frame forms the vehicle's plane of symmetry. The positive z-axis points to the nadir of the body. Finally, the y-axis completes a right-handed triad, as illustrated in Figure 3.2.8. Let the velocity vector in expressed in body frame be $\mathbf{v}$.

$$\mathbf{v} = a\mathbf{x_B} + s\mathbf{y_B} + u\ \mathbf{z_B} \tag{3.2.12}$$

where a, s, u are the components of relative velocity that acts along axial, side and up directions respectively.

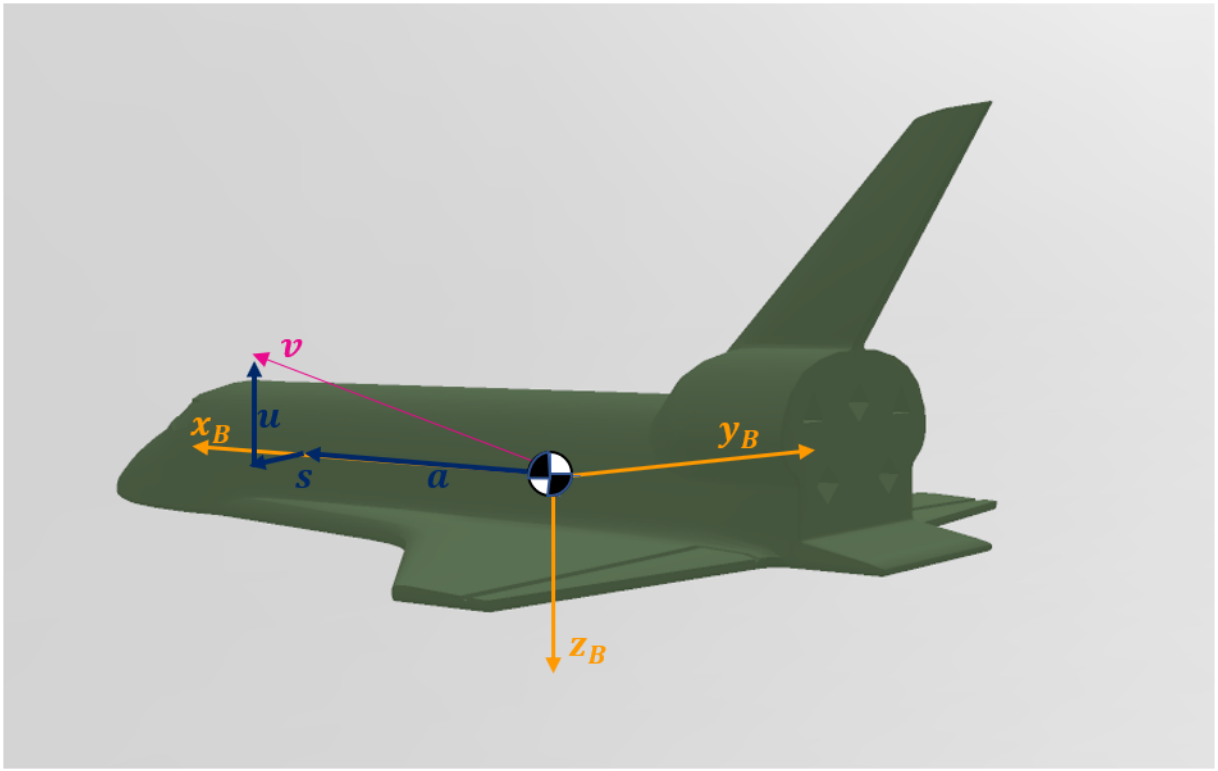

Figure 3.2.8: The Body Reference Frame.

The Sideslip angle β is defined as the angle between the velocity vector and its projection in the body's x-z plane.

$$\beta = sin^{-1}\left(\frac{s}{||\mathbf{v}||}\right) \tag{3.2.13}$$

A positive value of β is a counter-clockwise rotation measured from the velocity vector. The angle-of-attack, α, is the angle between velocity projection into x-z plane and the body x-axis. Hence,

$$\alpha = tan^{-1}\left(\frac{u}{a}\right) \tag{3.2.14}$$

A positive value of α is measured counterclockwise from the velocity vector in the $\mathbf{x_B}$-$\mathbf{z_B}$ plane. To align the Body Reference Frame with the Aerodynamic Reference Frame, the following steps are taken.

1. Let $\mathbf{v_b}$ be the projection of the velocity vector into the body x-z plane as shown in Figure 3.2.9. Rotate $\mathbf{x_B}$ around $\mathbf{y_B}$ such that $\mathbf{x_B}$ is concurrent with the projection of the velocity vector in the x-z plane. From Figure 3.2.10., it is clear that this is a rotation by an angle of $-\alpha$ around axis $\mathbf{y_B}$. Let the intermediate frame be labeled $\{\mathbf{x_B}' - \mathbf{y_B}' - \mathbf{z_B}'\}$. Note that $\mathbf{y_B}'$ coincides with $\mathbf{y_B}$.

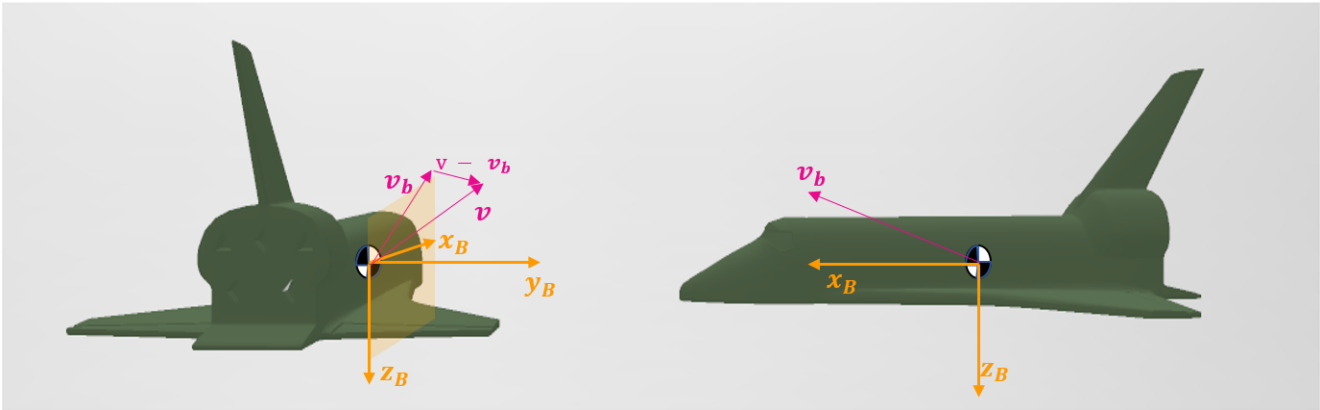

Figure 3.2.9: Velocity vector projected to the body x-z plane

2. Now, the velocity vector is completely contained in the plane spanned by $\mathbf{y_B}'$

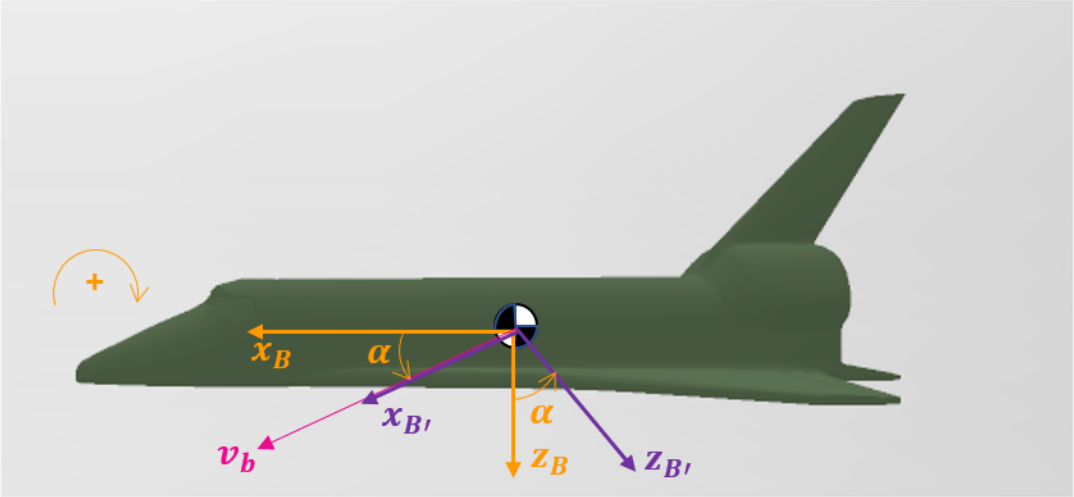

Figure 3.2.10: Rotation of the body x-axis by Angle of attack.

and $\mathbf{x_B}'$. In other words, the velocity vector and axes $\{\mathbf{x_B}' - \mathbf{y_B}'\}$ are normal to axis $\mathbf{z_B}'$. By design, rotating $\mathbf{x_B}'$ by an angle of $+\beta$ around $\mathbf{z_B}'$ aligns it to the velocity vector. Figure 3.2.11 illustrates this rotation. The new intermediate axis system is labelled as $\{\mathbf{x_B}" - \mathbf{y_B}" - \mathbf{z_B}"\}$. The shaded region represents the plane that contains $\mathbf{x_B}', \mathbf{y_B}', \mathbf{x_B}$ and $\mathbf{y_B}$.

3. At the end of the previous rotation, the body x-axis is concurrent with the velocity vector. Recall that the aerodynamic frame is defined such that $\mathbf{x_A}$ coincides with the velocity vector and $\mathbf{z_A}$ points along the lift vector. The planes spanned by axes $\{\mathbf{y_A}\text{-}\mathbf{z_A}\}$ and axes $\{\mathbf{y_B}"\text{-}\mathbf{z_B}"\}$ are normal to the velocity vector. Given that there can only be one plane normal to the velocity vector, axes $\{\mathbf{y_A}\text{-}\mathbf{z_A}\}$ and axes $\{\mathbf{y_B}"\text{-}\mathbf{z_B}"\}$ must be contained in the same plane. A rotation of the intermediate body axes along the x axis will completely align it with the ARF as shown in Figure 3.2.12.

   Figure 3.2.12 shows that the total lift vector $\hat{\mathbf{L}}$ makes an angle of η with axis $\mathbf{y_B}"$. A positive η is measured counter-clockwise from $\mathbf{z_B}"$. Therefore, a rotation along $\mathbf{x_B}"$ by an angle of $-\eta$ is performed to align the $\mathbf{z_B}"$ with the $\hat{\mathbf{L}}$. At the

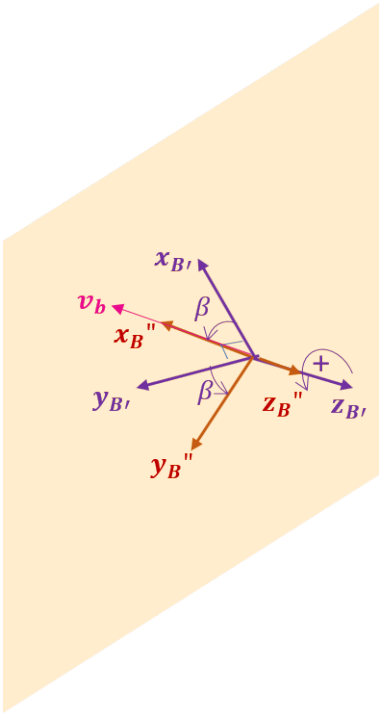

Figure 3.2.11: Rotation by sideslip angle.

end of this rotation, BRF is aligned with ARF.

Angle η is called the lift-roll angle. It describes the orientation of the Lift with respect to the projection of the axis $\mathbf{z_b}$ into the plane normal to the velocity vector. In Section 3.3.4, the calculation of η and its heavy dependence on the vehicle's aerodynamics will be explained.

To sum up, the transformation of a vector $\mathbf{u_B}$ from Body Reference Frame to Aerodynamic Reference Frame can be expressed as follows :

$$\begin{aligned}\mathbf{u_A} &= T_B^A \mathbf{u_B} = \mathbf{R_x}(-\eta)\mathbf{R_z}(\beta)\mathbf{R_y}(-\alpha)\mathbf{u_B} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\eta) & -sin(\eta) \\ 0 & sin(\eta) & cos(\eta) \end{bmatrix} \begin{bmatrix} cos(\beta) & sin(\beta) & 0 \\ -sin(\beta) & cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \mathbf{u_B}\end{aligned} \tag{3.2.15}$$

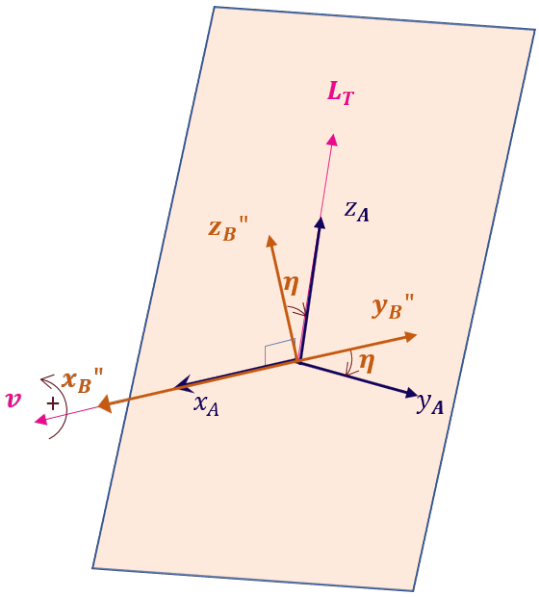

Figure 3.2.12: Rotation by $-\eta$

Simplifying the above expression for the transformation matrix T_B^A yields the following:

$$T_B^A = \begin{bmatrix} c(\beta)c(\alpha) & s(\beta) & c(\beta)s(\alpha) \\ s(\eta)s(\alpha) - c(\eta)s(\beta)c(\alpha) & c(\eta)c(\beta) & -s(\eta)c(\alpha) - c(\eta)s(\beta)s(\alpha) \\ -c(\eta)s(\alpha) - s(\eta)s(\beta)c(\alpha) & s(\eta)c(\beta) & c(\eta)c(\alpha) - s(\eta)s(\beta)s(\alpha) \end{bmatrix}$$

where $c()$ and $s()$ are shorthand for cosine and sine functions respectively. The inverse transformation from ARF to BRF can be written as:

$$\mathbf{u_B} = \mathbf{T_A^B u_A} = \left(\mathbf{T_B^A}\right)^{\mathbf{T}} \mathbf{u_A} \tag{3.2.16}$$

A direct transformation from Body Frame to Flight Path Frame can be made if the angles σ, η, β and α are known. The BRF to FRF transformation matrix can be written as:

$$\mathbf{T_B^F} = \mathbf{T_A^F T_B^A} = \left(\mathbf{T_F^A}\right)^{\mathbf{T}} \mathbf{T_B^A} \tag{3.2.17}$$

The angular velocity of the body frame with respect to the flight-path frame can be formulated as follows:

$$
\begin{aligned}
\omega^{\mathbf{B/A}} &= -\omega^{\mathbf{A/B}} = -\left(-\dot{\alpha}\mathbf{y_B} + \dot{\beta}\mathbf{z_{B'}} - \dot{\eta}\mathbf{x_A}\right) \\
&= \dot{\alpha}\mathbf{y_B} - \dot{\beta}\mathbf{z_{B'}} + \dot{\eta}\mathbf{x_A} \\
&= \dot{\alpha}T_B^A \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - \dot{\beta}T_{B"}^A \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \dot{\eta} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\
\omega_{\mathbf{A}}^{\mathbf{B/A}} &= \dot{\alpha}T_B^A \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - \dot{\beta} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\eta) & -sin(\eta) \\ 0 & sin(\eta) & cos(\eta) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \dot{\eta} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\
&= \dot{\alpha} \begin{bmatrix} sin(\beta) \\ cos(\eta)cos(\beta) \\ sin(\eta)cos(\beta) \end{bmatrix} - \dot{\beta} \begin{bmatrix} 0 \\ -sin(\eta) \\ cos(\eta) \end{bmatrix} + \dot{\eta} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\
\omega_{\mathbf{A}}^{\mathbf{B/A}} &= \begin{bmatrix} sin(\beta) & 0 & 1 \\ cos(\eta)cos(\beta) & sin(\eta) & 0 \\ sin(\eta)cos(\beta) & -cos(\eta) & 0 \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\eta} \end{bmatrix}
\end{aligned}
\tag{3.2.18}
$$

$\omega_{\mathbf{A}}^{\mathbf{B/A}}$ is the angular velocity of BRF with respect to ARF expressed in Aerodynamic Reference Frame. The same quantity can be expressed in BRF as:

$$
\begin{aligned}
\omega_{\mathbf{B}}^{\mathbf{B/A}} &= T_A^B \omega_{\mathbf{A}}^{\mathbf{B/A}} = \left(\mathbf{T_B^A}\right)^{\mathbf{T}} \omega_{\mathbf{A}}^{\mathbf{B/A}} \\
&= \begin{bmatrix} 0 & sin(\alpha) & cos(\alpha)cos(\beta) \\ 1 & 0 & sin(\beta) \\ 0 & -cos(\alpha) & sin(\alpha)cos(\beta) \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\eta} \end{bmatrix}
\end{aligned}
\tag{3.2.19}
$$

In body reference frame, $\omega_{\mathbf{A}}^{\mathbf{B/A}}$ is only a function on α, β and the angular rates.

3.3 Equations of Motion

In this section, the evolution of the state variables in time as function of the state and control inputs is formulated. The equations of motion, aerodynamic relationships and inertial properties of the system are used to define the state dynamics of the reentry problem.

3.3.1 Kinematic Equations

From Section 3.2.4, recall that the velocity vector is parallel to axis $\mathbf{x_F}$. Using Equation 3.2.8 a velocity vector of magnitude v acting along $\mathbf{x_F}$ can be expressed in Vehicle Pointing frame as:

$$\begin{aligned}
\mathbf{v_P} = \mathbf{T_F^P v_F} = v\left(\mathbf{T_P^F}\right)^{\mathbf{T}} &\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\
= v &\begin{bmatrix} sin(\gamma) & 0 & cos(\gamma) \\ cos(\gamma)cos(\psi) & -sin(\psi) & -sin(\gamma)cos(\psi) \\ cos(\gamma)sin(\psi) & cos(\psi) & -sin(\gamma)sin(\psi) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\
= v &\begin{bmatrix} sin(\gamma) \\ cos(\gamma)cos(\psi) \\ cos(\gamma)sin(\psi) \end{bmatrix}
\end{aligned} \tag{3.3.1}$$

$\mathbf{v_P}$ as expressed above is the relative velocity with respect to moving Earth, and is the velocity measured by an observer on the rotating planet-fixed system. The vector $\mathbf{v_P}$ can also be expressed in relation to frame ECEF. Consider position vector $\mathbf{r}$ in ECEF frame. $\mathbf{v_P}$ is the rate of change of $\mathbf{r}$ expressed in Vehicle Pointing Frame, the

derivative of $\mathbf{r}$ in VPF can be written as:

$$\begin{aligned}\mathbf{v_P} &= [\dot{r_E}]_P \\ &= \dot{\mathbf{r_P}} + \omega_P^{P/E} \times \mathbf{r_P} \\ &= \dot{\mathbf{r_P}} + T_E^P \omega_E^{P/E} \times \mathbf{r_P}\end{aligned} \tag{3.3.2}$$

A position vector of magnitude r is easily expressed in VPF as

$$\mathbf{r_P} = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix}$$

Drawing in the results from Equations 3.2.4 and 3.2.5 to Equation 3.3.2:

$$\mathbf{v_P} = \begin{bmatrix} \dot{r} \\ 0 \\ 0 \end{bmatrix} + T_E^P \omega_E^{P/E} \times \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} \dot{r} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} cos(\lambda)cos(\mu) & cos(\lambda)sin(\mu) & sin(\lambda) \\ -sin(\mu) & cos(\mu) & 0 \\ -sin(\lambda)cos(\mu) & -sin(\lambda)sin(\mu) & cos(\lambda) \end{bmatrix} \begin{bmatrix} \dot{\lambda}sin(\mu) \\ -\dot{\lambda}cos(\mu) \\ \dot{\mu} \end{bmatrix} \times \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{v_P} = \begin{bmatrix} \dot{r} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\mu}\,(sin(\lambda)) \\ -\dot{\lambda} \\ \dot{\mu}cos(\lambda) \end{bmatrix} \times \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix}$$

The above equation can be simplified to:

$$\mathbf{v_P} = \begin{bmatrix} \dot{r} \\ r\dot{\mu}cos(\lambda) \\ r\dot{\lambda} \end{bmatrix} \tag{3.3.3}$$

Equating Equations 3.3.1 and 3.3.3, the kinematic equations of motion are derived:

$$\begin{aligned} \dot{r} &= v\sin(\gamma) \\ \dot{\mu} &= \frac{v\cos(\gamma)\cos(\psi)}{r\cos(\lambda)} \\ \dot{\lambda} &= \frac{v\cos(\gamma)\sin(\psi)}{r} \end{aligned} \tag{3.3.4}$$

v is the magnitude of the Earth-relative velocity vector.

3.3.2 Force Equations

Abrahamson [12] derives the force equations of motion relative to VPF, is expressed in the Flight-Path frame as follows:

$$\begin{aligned}
\begin{bmatrix} \dot{v} \\ v\dot{\psi}cos(\gamma) \\ v\dot{\gamma} \end{bmatrix} &= \frac{\mathbf{F}}{m} \\
&- 2v\omega_{\oplus} \begin{bmatrix} 0 \\ sin(\lambda)cos(\gamma) - cos(\lambda)sin(\gamma)sin(\psi) \\ -cos(\lambda)cos(\psi) \end{bmatrix} \\
&- \frac{v^2cos^2(\gamma)}{r} \begin{bmatrix} 0 \\ cos(\psi)tan(\lambda) \\ -1 \end{bmatrix} \\
&- r\omega_{\oplus}^2cos(\lambda) \begin{bmatrix} -cos(\lambda)sin(\gamma) + sin(\lambda)cos(\gamma)sin(\psi) \\ sin(\lambda)cos(\psi) \\ -cos(\lambda)cos(\gamma) - sin(\lambda)sin(\gamma)sin(\psi) \end{bmatrix}
\end{aligned} \tag{3.3.5}$$

The first term in Equation 3.3.5 represents the inertial acceleration acting on the body. Because the RV is assumed to be non-thrusting, the only external forces acting on the body considered in this study are gravity and aerodynamic forces. Furthermore, the change in mass of the body over the course of reentry is assumed to be negligible in this study. Hence the mass m is treated as a constant value.

- Gravitational Force: As mentioned in Section 2.4.1 gravity always acts opposite to the position vector. Therefore, gravity can be expressed in Flight Path Frame

as:

$$\begin{aligned}
\mathbf{g_F} &= \frac{\mu_\oplus}{r^2}\left(-\mathbf{x_P}\right) = \frac{\mu_\oplus}{r^2} T_P^F \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \\
\mathbf{g_F} &= -\frac{\mu_\oplus}{r^2} \begin{bmatrix} sin(\gamma) \\ 0 \\ cos(\gamma) \end{bmatrix}
\end{aligned} \tag{3.3.6}$$

- Aerodynamic Forces: The aerodynamic forces acting on the body are functions of vehicle state and control inputs. The accelerations expressed in BRF can be written as follows:

$$\begin{aligned}
\mathbf{a_F} &= \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \\
\mathbf{a_F} &= \left(\frac{\rho v^2}{2}\frac{S}{m}\right) \mathbf{T_B^F} \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix}
\end{aligned} \tag{3.3.7}$$

$\boldsymbol{T_B^F}$ is the transformation matrix from body to Flight-Path Frame. ρ is the atmospheric density. It is primarily a function of the geodetic altitude $h = r - r_\oplus$ and can be obtained from the atmospheric model. v is the magnitude of the Earth relative velocity. S is the aerodynamic reference area, specified in the vehicle's aerodynamic database. And m is the vehicle mass. C_x, C_y, C_z are the axial,lateral and normal aerodynamic coefficients. They are tabulated in the aerodynamic database as a function of Mach number, angle of attack (α), sideslip

Angle (β), inertial attitude rates (p, q, r) and control deflections(δa,δe,δr).

To summarize, the inertial accelerations in the Flight-Path Frame are given by

$$\frac{\mathbf{F}}{m} = -\frac{\mu_\oplus}{r^2}\begin{bmatrix}\sin(\gamma)\\0\\\cos(\gamma)\end{bmatrix} + \left(\frac{\rho v^2}{2}\frac{S}{m}\right)\mathbf{T}_{\mathbf{B}}^{\mathbf{F}}\begin{bmatrix}C_x\\C_y\\C_z\end{bmatrix} \tag{3.3.8}$$

An expansion of the elements of Matrix $\boldsymbol{T}_{\boldsymbol{B}}^{\boldsymbol{F}}$ in the above equation can be found in Appendix A.

After collecting all the terms, the force equations of motion is summarized below:

$$\begin{aligned}\begin{bmatrix}\dot{v}\\v\dot{\psi}\cos(\gamma)\\v\dot{\gamma}\end{bmatrix} &= \left(\frac{\rho v^2}{2}\frac{S}{m}\right)\mathbf{T}_{\mathbf{B}}^{\mathbf{F}}\begin{bmatrix}C_x\\C_y\\C_z\end{bmatrix} - \frac{\mu_\oplus}{r^2}\begin{bmatrix}\sin(\gamma)\\0\\\cos(\gamma)\end{bmatrix}\\ &- 2v\omega_\oplus\begin{bmatrix}0\\\sin(\lambda)\cos(\gamma)-\cos(\lambda)\sin(\gamma)\sin(\psi)\\-\cos(\lambda)\cos(\psi)\end{bmatrix}\\ &- \frac{v^2\cos^2(\gamma)}{r}\begin{bmatrix}0\\\cos(\psi)\tan(\lambda)\\-1\end{bmatrix}\\ &- r\omega_\oplus^2\cos(\lambda)\begin{bmatrix}-\cos(\lambda)\sin(\gamma)+\sin(\lambda)\cos(\gamma)\sin(\psi)\\\sin(\lambda)\cos(\psi)\\-\cos(\lambda)\cos(\gamma)-\sin(\lambda)\sin(\gamma)\sin(\psi)\end{bmatrix}\end{aligned} \tag{3.3.9}$$

The third term in Equation 3.3.9 represents Coriolis acceleration due to Earth's rotation. The fourth term in Equation 3.3.10 is due to the Coriolis acceleration of the Flight-Path frame due to the rotation of Vehicle-Pointing Reference Frame. The fi-

nal term arises due to centripetal acceleration. Equation 3.3.9 can be simplified to calculate the rates of change of speed, heading and flight-path angle using Equation 1.0.1.

$$
\begin{aligned}
\dot{v} &= \frac{\rho v^2}{2}\frac{S}{m}\{C_y\sin(\beta) + C_x\cos(\alpha)\cos(\beta) + C_z\cos(\beta)\sin(\alpha)\} \\
&\quad - \frac{\mu_\oplus}{r^2}\sin(\gamma) \\
&\quad - r\omega_\oplus^2\cos(\lambda)\left(-\cos(\lambda)\sin(\gamma) + \sin(\lambda)\cos(\gamma)\sin(\psi)\right) \\
\dot{\psi} &= \Big(\frac{\rho v^2}{2}\frac{S}{m}\{(C_y(\cos(\sigma)\cos(\beta)\cos(\eta) + \cos(\beta)\sin(\sigma)\sin(\eta)) \\
&\quad + C_x(\cos(\sigma)(\sin(\alpha)\sin(\eta) - \cos(\alpha)\cos(\eta)\sin(\beta)) - \sin(\sigma)(\cos(\eta)\sin(\alpha) + \cos(\alpha)\sin(\beta)\sin(\eta))) \\
&\quad - C_z(\cos(\sigma)(\cos(\alpha)\sin(\eta) + \cos(\eta)\sin(\alpha)\sin(\beta)) - \sin(\sigma)(\cos(\alpha)\cos(\eta) - \sin(\alpha)\sin(\beta)\sin(\eta)))\} \\
&\quad - 2v\omega_\oplus\{\sin(\lambda)\cos(\gamma) - \cos(\lambda)\sin(\gamma)\sin(\psi)\} \\
&\quad - \frac{v^2\cos^2(\gamma)}{r}\cos(\psi)\tan(\lambda) - r\omega_\oplus^2\cos(\lambda)\sin(\lambda)\cos(\psi)\Big)\frac{1}{v\cos(\gamma)} \\
\dot{\gamma} &= \Big(\frac{\rho v^2}{2}\frac{S}{m}\{(C_y(\cos(\sigma)\cos(\beta)\sin(\eta) - \cos(\beta)\sin(\sigma)\cos(\eta)) \\
&\quad - C_x(\cos(\sigma)(\sin(\alpha)\cos(\eta) + \cos(\alpha)\sin(\eta)\sin(\beta)) + \sin(\sigma)(\sin(\eta)\sin(\alpha) - \cos(\alpha)\sin(\beta)\sin(\eta))) \\
&\quad + C_z(\cos(\sigma)(\cos(\alpha)\cos(\eta) - \sin(\eta)\sin(\alpha)\sin(\beta)) + \sin(\sigma)(\cos(\alpha)\sin(\eta) + \sin(\alpha)\sin(\beta)\cos(\eta)))\} \\
&\quad - \frac{\mu_\oplus}{r^2}\cos(\gamma) + 2v\omega_\oplus\{\cos(\lambda)\cos(\psi)\} \\
&\quad + \frac{v^2\cos^2(\gamma)}{r} + r\omega_\oplus^2\{\cos(\lambda)\cos(\gamma) + \sin(\lambda)\sin(\gamma)\sin(\psi)\}\Big)\frac{1}{v}
\end{aligned}
\tag{3.3.10}
$$

3.3.3 Attitude Rates

The inertial angular velocities with respect to axes $\mathbf{x_B}, \mathbf{y_B}, \mathbf{z_B}$ are denoted by p, q and r respectively. Etkin [18] derives the angular acceleration rates as:

$$\begin{aligned}
\dot{p} &= \frac{1}{I_{xx}} \left\{ L + I_{zx}(\dot{r} + pq) + (I_{yy} - I_{zz})qr + I_{yz}(q^2 - r^2) + I_{xy}(\dot{q} - rp) \right\} \\
\dot{q} &= \frac{1}{I_{yy}} \left\{ M + I_{zx}(r^2 - p^2) + (I_{zz} - I_{xx})rp + I_{yz}(\dot{r} - pq) + I_{xy}(\dot{p} + qr) \right\} \\
\dot{r} &= \frac{1}{I_{zz}} \left\{ N + I_{zx}(\dot{p} - qr) + (I_{xx} - I_{yy})pq + I_{yz}(\dot{q} + rp) + I_{xy}(p^2 - q^2) \right\}
\end{aligned}$$

I_{xx}, I_{yy}, I_{zz} are the vehicle's moments of inertia, and they are the diagonal terms of the inertia tensor. I_{zx}, I_{xy}, I_{yz} are the vehicle's products of inertia. They are expressed with respect to the vehicle center of mass in the Body Reference Frame. For the scope of this research, they are assumed to be constant throughout reentry. Most aerospace vehicles have a plane of symmetry. As mentioned in Section 3.2.6, the plane spanned by x-z defines the plane of symmetry. Thus, $I_{xy} = I_{yz} = 0$ and the previous equation can be simplified to:

$$\begin{aligned}
\dot{p} &= \frac{1}{I_{xx}} \left\{ L + I_{zx}(\dot{r} + pq) + (I_{yy} - I_{zz})qr \right\} \\
\dot{q} &= \frac{1}{I_{yy}} \left\{ M + I_{zx}(r^2 - p^2) + (I_{zz} - I_{xx})rp \right\} \\
\dot{r} &= \frac{1}{I_{zz}} \left\{ N + I_{zx}(\dot{p} - qr) + (I_{xx} - I_{yy})pq \right\}
\end{aligned} \tag{3.3.11}$$

where L, M, N are the torques in the body axial, lateral and normal directions respectively. The coupling between pitch and yaw rates is visible. A minor modification can be made to how $\dot{p}$ is formulated to express the attitude rates in terms of known

quantities.

$$
\begin{aligned}
I_{xx}\dot{p} &= L + I_{zx}(\dot{r} + pq) + (I_{yy} - I_{zz})qr \\
I_{xx}\dot{p} &= L + I_{zx}\left\{\frac{1}{I_{zz}}\left\{N + I_{zx}(\dot{p} - qr) + (I_{xx} - I_{yy})pq\right\}\right\} + I_{zx}pq + (I_{yy} - I_{zz})qr \\
I_{xx}\dot{p} &= L + \frac{I_{zx}}{I_{zz}}N + \frac{I_{zx}}{I_{zz}}I_{zx}(\dot{p} - qr) + \frac{I_{zx}}{I_{zz}}(I_{xx} - I_{yy})pq \\
&\quad + I_{zx}pq + (I_{yy} - I_{zz})qr
\end{aligned}
$$

The above equation is solved for $\dot{p}$ to obtain:

$$
\begin{aligned}
\dot{p} &= \frac{I_{zz}}{I_{xx}I_{zz} - I_{zx}^2}\left\{L + \frac{I_{zx}}{I_{zz}}N + (I_{yy} - I_{zz} - \frac{I_{zx}^2}{I_{zz}})qr + \left(1 + \frac{I_{xx} - I_{yy}}{I_{zz}}\right)I_{zx}pq\right\} \\
\dot{q} &= \frac{1}{I_{yy}}\left\{M + I_{zx}(r^2 - p^2) + (I_{zz} - I_{xx})rp\right\} \\
\dot{r} &= \frac{1}{I_{zz}}\left\{N + I_{zx}(\dot{p} - qr) + (I_{xx} - I_{yy})pq\right\}
\end{aligned}
\tag{3.3.12}
$$

L, M, N in the equation above denote the Roll Moment, Pitch Moment and Yaw Moment respectively in the BRF. They can be expressed as:

$$
\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \left(\frac{\rho v^2 S}{2}\right)\begin{bmatrix} bC_l \\ cC_m \\ bC_n \end{bmatrix} \tag{3.3.13}
$$

ρ, S, v are the atmospheric density, aerodynamic reference area and magnitude of the instantaneous Earth relative velocity respectively. b and c are respectively the wingspan and the reference chord length. C_l, C_m and C_n are obtained from the aerodynamic database using Equation 2.6.2. Hence the attitude rates are also a function of the state and control inputs.

3.3.4 Determination of Lift-Roll Angle

The lift-roll angle η was introduced in Section 3.2.6. It describes the orientation of the lift vector with respect to the projection of the axis $\mathbf{y_B}$ into the lift plane (Figure 3.3.1). The force coefficients $\{ C_x, C_y, C_z \}$ obtained from the aerodynamic database describe the relative magnitudes of aerodynamic acceleration along the body axial, lateral and normal directions respectively. The following series of rotations are performed to transform the coefficients into frame $\{ x_B", y_B", z_B" \}$:

$$\begin{bmatrix} C_x" \\ C_y" \\ C_z" \end{bmatrix} = \mathbf{R_z}(\beta)\mathbf{R_y}(-\alpha) \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \tag{3.3.14}$$

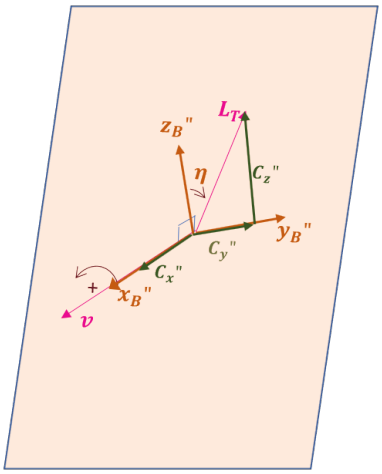

Figure 3.3.1: Depiction of Lift-Roll angle.

In Figure 3.3.1, the shaded region indicates the lift plane. Vectors $\mathbf{v}$,$\mathbf{C_X}$ and $\mathbf{x_B}$" are normal to the lift-plane. As illustrated in Section 3.2.6, the above rotation casts the coefficients along the lift and drag planes. C_x" describes the magnitude of the total aerodynamic acceleration experienced by the vehicle along the direction of the velocity vector. The vectors C_y" and C_z" span the lift plane. The total lift coefficient

C_L can be derived as:

$$C_L = \sqrt{(C_y")^2 + (C_z")^2} \tag{3.3.15}$$

The direction of the total lift in the intermediate frame can be expressed as:

$$\hat{\mathbf{L}} = \frac{C_y"}{C_L}\mathbf{y_B}" + \frac{C_z"}{C_L}\mathbf{z_B}" \tag{3.3.16}$$

The body y-axis projected to the lift plane in the intermediate frame $\{ x_B", y_B", z_B"\}$ is simply **[0 1 0]'**. Therefore, the lift-roll angle η as defined to be:

$$\eta = tan^{-1}\left(\frac{C_y"}{C_z"}\right) \tag{3.3.17}$$

3.3.4.1 Estimating $\dot{\eta}$

Since the lift roll angle η is part of the defined state, its rate of change has to be formulated to describe the system dynamics. Equation 3.3.17 can be re-written as:

$$tan(\eta) = \frac{C_y"}{C_z"}$$

Differentiating both sides with respect to time, we get :

$$\dot{\eta}\ \frac{1}{\cos^2(\eta)} = \frac{\dot{C_y}"C_z" - \dot{C_z}"C_y"}{{C_z"}^2}$$

where $\dot{C}_x, \dot{C}_y, \dot{C}_z$ are the instantaneous rate of change of the force coefficients in the body axes. Using Equation 3.3.14 to expand the above:

$$\dot{\eta} = \cos^2(\eta)\ \left\{\frac{\tilde{a}}{\tilde{d}}\ \dot{\alpha} + \frac{\tilde{b}}{\tilde{d}}\ \dot{\beta} + \frac{\tilde{c}}{\tilde{d}}\right\} \tag{3.3.18}$$

where $\tilde{a}$,$\tilde{b}$ and $\tilde{c}$ are shorthand for :

$$\tilde{a} = cos(\beta)\ \{C_x cos(\alpha) + C_z sin(\alpha)\} C_y - sin(\beta)\ (C_x^2 + C_z^2) \tag{3.3.19}$$

$$\begin{aligned} \tilde{b} = cos(\beta)\left\{(C_x^2 - C_z^2) cos(\alpha) sin(\alpha) + C_x C_z \left(sin^2(\alpha) - cos^2(\alpha)\right)\right\} \\ + C_y sin(\beta)\left\{C_x sin(\alpha) - C_z cos(\alpha)\right\} \end{aligned} \tag{3.3.20}$$

$$\begin{aligned} \tilde{c} = \dot{C}_x\left\{C_y sin(\alpha) cos(\beta) - C_z sin(\beta)\right\} \\ + \dot{C}_y\left\{C_z cos(\alpha) - C_x sin(\alpha)\right\} cos(\beta) \\ + \dot{C}_z\left\{C_x sin(\beta) - C_y cos(\alpha) cos(\beta)\right\} \end{aligned} \tag{3.3.21}$$

$$\tilde{d} = (C_z")^2 = \{C_z \cos(\alpha) - C_x \sin(\alpha)\}^2 \tag{3.3.22}$$

For simplicity, it is assumed that the values for $\dot{C}_x, \dot{C}_y, \dot{C}_z$ are approximately zero. In effect, $\dot{\eta}$ can be approximated by:

$$\dot{\eta} \approx \cos^2(\eta)\ \left\{ \frac{\tilde{a}}{\tilde{d}}\ \dot{\alpha} + \frac{\tilde{b}}{\tilde{d}}\ \dot{\beta} \right\} \tag{3.3.23}$$

Expressions for $\dot{\alpha}$ and $\dot{\beta}$ will be derived in Section 3.3.5.

3.3.5 Rates of Aerodynamic Angles

The last three entries in the state formulation defined in Section 3.4.1 are bank angle σ, angle of attack α and sideslip angle β. To completely define the state space, the time rate of change of these three parameters must be defined as a function of state quantities and control inputs. Recall that the inertial angular velocities p, q, r are expressed in the body frame. The angular velocities can be written as an elaborate function of the angular rates defined in Section 3.2 in terms of Body Reference frame

as:

$$\begin{aligned}\begin{bmatrix} p \\ q \\ r \end{bmatrix}_B &= \omega_{\mathbf{B}}^{\mathbf{E/I}} + \omega_{\mathbf{B}}^{\mathbf{P/E}} + \omega_{\mathbf{B}}^{\mathbf{F/P}} + \omega_{\mathbf{B}}^{\mathbf{A/F}} + \omega_{\mathbf{B}}^{\mathbf{B/A}} \\ &= \mathbf{T_A^B T_F^A T_P^F T_E^P T_I^E}\, \omega_{\mathbf{I}}^{\mathbf{E/I}} + \mathbf{T_A^B T_F^A T_P^F T_E^P}\, \omega_{\mathbf{E}}^{\mathbf{P/E}} \\ &\quad + \mathbf{T_A^B T_F^A T_P^F}\, \omega_{\mathbf{P}}^{\mathbf{F/P}} + \mathbf{T_A^B T_F^A}\, \omega_{\mathbf{F}}^{\mathbf{A/F}} + \omega_{\mathbf{B}}^{\mathbf{B/A}}\end{aligned} \tag{3.3.24}$$

Define the inertial rates of roll, pitch and yaw of the Flight-Path Frame as:

$$\begin{aligned}\begin{bmatrix} p_F \\ q_F \\ r_F \end{bmatrix}_B &= \omega_{\mathbf{B}}^{\mathbf{E/I}} + \omega_{\mathbf{B}}^{\mathbf{P/E}} + \omega_{\mathbf{B}}^{\mathbf{F/P}} \\ &= \mathbf{T_A^B T_F^A T_P^F T_E^P T_I^E}\, \omega_{\mathbf{I}}^{\mathbf{E/I}} + \mathbf{T_A^B T_F^A T_P^F T_E^P}\, \omega_{\mathbf{E}}^{\mathbf{P/E}} + \mathbf{T_A^B T_F^A T_P^F} \omega_{\mathbf{P}}^{\mathbf{F/P}}\end{aligned} \tag{3.3.25}$$

Further expansion of Equation 3.3.25 is omitted for brevity. The angular rates in the above expression are expressed relative to BRF. Equation 3.3.24 can then be written as:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix}_B = \begin{bmatrix} p_F \\ q_F \\ r_F \end{bmatrix}_B + T_A^B T_F^A\, \omega_{\mathbf{F}}^{\mathbf{A/F}} + \omega_{\mathbf{B}}^{\mathbf{B/A}}$$

The above expression is simplified using equations derived in Section 3.2 to obtain:

$$\begin{bmatrix} p - p_F \\ q - q_F \\ r - r_F \end{bmatrix}_B = \begin{bmatrix} -cos(\alpha)cos(\beta) \\ -sin(\beta) \\ -sin(\alpha)cos(\beta) \end{bmatrix} \dot{\sigma} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \dot{\alpha} + \begin{bmatrix} sin(\alpha) \\ 0 \\ -cos(\alpha) \end{bmatrix} \dot{\beta} + \begin{bmatrix} cos(\alpha)cos(\beta) \\ sin(\beta) \\ sin(\alpha)cos(\beta) \end{bmatrix} \dot{\eta} \tag{3.3.26}$$

In Section 3.3.4.1, an expression for $\dot{\eta}$ was derived as a function of $\dot{\alpha}$ and $\dot{\beta}$. Substituting Equation 3.3.23 into 3.3.26 yields:

$$\begin{aligned}
\begin{bmatrix} p - p_F \\ q - q_F \\ r - r_F \end{bmatrix}_B &= \begin{bmatrix} -cos(\alpha)cos(\beta) \\ -sin(\beta) \\ -sin(\alpha)cos(\beta) \end{bmatrix} \dot{\sigma} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \dot{\alpha} + \begin{bmatrix} sin(\alpha) \\ 0 \\ -cos(\alpha) \end{bmatrix} \dot{\beta} \\
&+ \begin{bmatrix} cos(\alpha)cos(\beta) \\ sin(\beta) \\ sin(\alpha)cos(\beta) \end{bmatrix} \cos^2(\eta) \left\{ \frac{\tilde{a}}{\tilde{d}} \dot{\alpha} + \frac{\tilde{b}}{\tilde{d}} \dot{\beta} \right\} \qquad (3.3.27) \\
\begin{bmatrix} p - p_F \\ q - q_F \\ r - r_F \end{bmatrix}_B &= \mathbf{G} \begin{bmatrix} \dot{\sigma} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix}
\end{aligned}$$

where G is a 3-by-3 matrix defined as:

$$\mathbf{G} = \begin{bmatrix} -cos(\alpha)cos(\beta) & \cos^2(\eta)cos(\alpha)cos(\beta)\frac{\tilde{a}}{\tilde{d}} & sin(\alpha) + \cos^2(\eta)cos(\alpha)cos(\beta)\frac{\tilde{b}}{\tilde{d}} \\ -sin(\beta) & 1 + \cos^2(\eta)sin(\beta)\frac{\tilde{a}}{\tilde{d}} & \cos^2(\eta)sin(\beta)\frac{\tilde{b}}{\tilde{d}} \\ -sin(\alpha)cos(\beta) & \cos^2(\eta)sin(\alpha)cos(\beta)\frac{\tilde{a}}{\tilde{d}} & -cos(\alpha) + \cos^2(\eta)sin(\alpha)cos(\beta)\frac{\tilde{b}}{\tilde{d}} \end{bmatrix} \qquad (3.3.28)$$

$\tilde{a}, \tilde{b}, \tilde{d}$ are obtained from Equations 3.3.19, 3.3.20 and 3.3.22 respectively. Finally, the rates of change of aerodynamic angles can be written in terms of states and controls as follows:

$$\begin{bmatrix} \dot{\sigma} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \mathbf{G^{-1}} \begin{bmatrix} p - p_F \\ q - q_F \\ r - r_F \end{bmatrix}_{\mathbf{B}} \qquad (3.3.29)$$

3.3.5.1 Invertibility of Matrix G

To ensure that a solution is plausible, it is necessary to establish that **G** is invertible. The determinant of matrix G is found to be:

$$\begin{aligned} \det(\mathbf{G}) &= \cos(\beta)\cos^2(\alpha) + \cos(\beta)\sin^2(\alpha) \\ \det(\mathbf{G}) &= \cos(\beta) \end{aligned} \tag{3.3.30}$$

For $\mathbf{G^{-1}}$ to exist [26],

$$\det(\mathbf{G}) \neq 0$$

For Equation 3.3.30, this means:

$$\beta \neq \frac{\pi}{2}(2j+1), \quad j \in \mathbb{Z} \tag{3.3.31}$$

where $\mathbb{Z}$ is the set of all integers.

For almost all practical applications of flight dynamics [18] [20] :

$$|\beta| \leq \frac{\pi}{4} = 45^\circ$$

Therefore, **G** is invertible and Equation 3.3.29 always has a solution for values of β under the assumptions maintained in this study.

3.4 High Fidelity Simulation Environment

The equations of motion derived in the previous section are used to define a dynamic system. The states and control elements of the system are identified in the following sections.

3.4.1 States

The 12 state variables of the system are defined as follows:

$$\mathbf{X} = [r\ \mu\ \lambda\ v\ \psi\ \gamma\ p\ q\ r\ \sigma\ \alpha\ \beta]^{\mathbf{T}} \tag{3.4.1}$$

3.4.2 Controls

The control vector consists of the deflections from each individual control surface on the RV. The aerodynamic database maps the control deflections in the respective ambient conditions to the incremental force and moment contributions. The number of control elements vary from system to system. For the Space Shuttle, there were six control surfaces [27]. The control deflections can be written as:

$$\mathbf{u} = \begin{bmatrix} \delta 1 & \delta 2, ... & \delta n \end{bmatrix}^{\mathbf{T}} \tag{3.4.2}$$

where n is the number of control surfaces. In some cases, the aerodynamic database may be expressed in terms of effective roll, pitch and yaw:

$$\mathbf{u} = \begin{bmatrix} \delta a & \delta e & \delta r \end{bmatrix}^{\mathbf{T}} \tag{3.4.3}$$

where $\delta a, \delta e, \delta r$ are effective roll, pitch, and yaw, respectively. A control mixing algorithm that maps individual deflections to effective values are usually pre-computed [28]. As shown in Section 3.6, this study will focus on a control vector similar to Equation 3.4.2.

3.4.3 System Dynamics

The state update function describes how the state variables are dynamically changing in terms of state and control variables.

$$\dot{\mathbf{X}} = \mathbf{F}(\mathbf{X}, \mathbf{u}, , \mathbf{t})$$

Consolidating the equations of motion from Section 3.3, a full 6-DoF high fidelity dynamic system can be expressed in Figure 3.4.1. It can seen that the equations are time-invariant and hence are functions of state and control variables alone.

$$
\begin{aligned}
\dot{r} &= v\sin(\gamma) \\
\dot{\mu} &= \frac{v\cos(\gamma)\cos(\psi)}{r\cos(\lambda)} \\
\dot{\lambda} &= \frac{v\cos(\gamma)\sin(\psi)}{r} \\
\dot{v} &= \frac{\rho v^2}{2}\frac{S}{m}\{C_y\sin(\beta) + C_x\cos(\alpha)\cos(\beta) + C_z\cos(\beta)\sin(\alpha)\} - \frac{\mu_\oplus}{r^2}\sin(\gamma) \\
&\quad - r\omega_\oplus^2\cos(\lambda)\left(-\cos(\lambda)\sin(\gamma) + \sin(\lambda)\cos(\gamma)\sin(\psi)\right) \\
\dot{\psi} &= \Big(\frac{\rho v^2}{2}\frac{S}{m}\{(C_y(\cos(\sigma)\cos(\beta)\cos(\eta) + \cos(\beta)\sin(\sigma)\sin(\eta)) \\
&\quad + C_x(\cos(\sigma)(\sin(\alpha)\sin(\eta) - \cos(\alpha)\cos(\eta)\sin(\beta)) \\
&\quad - \sin(\sigma)(\cos(\eta)\sin(\alpha) + \cos(\alpha)\sin(\beta)\sin(\eta))) \\
&\quad - C_z(\cos(\sigma)(\cos(\alpha)\sin(\eta) + \cos(\eta)\sin(\alpha)\sin(\beta)) \\
&\quad - \sin(\sigma)(\cos(\alpha)\cos(\eta) - \sin(\alpha)\sin(\beta)\sin(\eta)))\} \\
&\quad - 2v\omega_\oplus\{\sin(\lambda)\cos(\gamma) - \cos(\lambda)\sin(\gamma)\sin(\psi)\} \\
&\quad - \frac{v^2\cos^2(\gamma)}{r}\cos(\psi)\tan(\lambda) - r\omega_\oplus^2\cos(\lambda)\sin(\lambda)\cos(\psi)\Big)\frac{1}{v\cos(\gamma)} \\
\dot{\gamma} &= \Big(\frac{\rho v^2}{2}\frac{S}{m}\{(C_y(\cos(\sigma)\cos(\beta)\sin(\eta) - \cos(\beta)\sin(\sigma)\cos(\eta)) \\
&\quad - C_x(\cos(\sigma)(\sin(\alpha)\cos(\eta) + \cos(\alpha)\sin(\eta)\sin(\beta)) \\
&\quad + \sin(\sigma)(\sin(\eta)\sin(\alpha) - \cos(\alpha)\sin(\beta)\sin(\eta))) \\
&\quad + C_z(\cos(\sigma)(\cos(\alpha)\cos(\eta) - \sin(\eta)\sin(\alpha)\sin(\beta)) \\
&\quad - \sin(\sigma)(\cos(\alpha)\sin(\eta) + \sin(\alpha)\sin(\beta)\cos(\eta)))\} \\
&\quad - \frac{\mu_\oplus}{r^2}\cos(\gamma) + 2v\omega_\oplus\{\cos(\lambda)\cos(\psi)\} \\
&\quad + \frac{v^2\cos^2(\gamma)}{r} + r\omega_\oplus^2\{\cos(\lambda)\cos(\gamma) + \sin(\lambda)\sin(\gamma)\sin(\psi)\}\Big)\frac{1}{v} \\
\dot{p} &= \frac{1}{I_{xx}}\left\{L + I_{zx}(\dot{r} + pq) + (I_{yy} - I_{zz})qr + I_{yz}(q^2 - r^2) + I_{xy}(\dot{q} - rp)\right\} \\
\dot{q} &= \frac{1}{I_{yy}}\left\{M + I_{zx}(r^2 - p^2) + (I_{zz} - I_{xx})rp + I_{yz}(\dot{r} - pq) + I_{xy}(\dot{p} + qr)\right\} \\
\dot{r} &= \frac{1}{I_{zz}}\left\{N + I_{zx}(\dot{p} - qr) + (I_{xx} - I_{yy})pq + I_{yz}(\dot{q} + rp) + I_{xy}(p^2 - q^2)\right\} \\
\begin{matrix}\dot{\sigma}\\ \dot{\alpha}\\ \dot{\beta}\end{matrix} &= \mathbf{G^{-1}}\begin{bmatrix} p - p_F \\ q - q_F \\ r - r_F \end{bmatrix}_B
\end{aligned}
$$

*This font color indicates terms that are functions of control deflections.

Figure 3.4.1: Full 6-DoF Equations of Motion

3.5 Optimal Control Problem

3.5.1 Cost Functions

Two different cost functionals are investigated for this research. First, the control profile for minimum time solution from initial condition to final condition is investigated. Therefore the cost function is formulated as:

$$J_1 = \int_{t_0}^{t} dt \tag{3.5.1}$$

Secondly, a profile that minimizes the control cost at each time will be studied. The cost functional in this case would be:

$$J_2 = \int_{t_0}^{t} \mathbf{u^T u}\ \mathbf{dt} \tag{3.5.2}$$

where $\mathbf{u}$ is the control vector at each timestep. Such a cost function is intended to reward lesser deflection of each of the n control surfaces.

3.5.2 Path Constraints

The optimal control problem is subjected to the following constraints.

- The instantaneous heat rate at any point in the trajectory must be less than Q_{max}. The estimation of instantaneous heat rate based on state values is detailed in Section 2.8.

  $$Q(t) \leq Q_{max} \tag{3.5.3}$$

  where $Q(t)$ is given by Equation 2.8.1 in $\frac{W}{m^2}$.

- Maximum Dynamic Pressure: Dynamic pressure is required to be under the

prescribed maximum at all times.

$$q_\infty(t) \leq q_{\infty,max} \tag{3.5.4}$$

The instantaneous dynamic pressure is computed as follows

$$q_\infty = \frac{1}{2}\,\rho\,v^2$$

where ρ is the atmospheric density.

- Load Factor: The load factor a_{LF} is the total magnitude normal and lateral acceleration, with respect to BRF, that the airframe is subjected to and should not exceed the limits specified by the airframe designer.

$$a_{LF}(t) = \sqrt{a_y^2 + a_z^2} \tag{3.5.5}$$

$$a_{LF}(t) \leq a_{LF,max} \tag{3.5.6}$$

where the lateral and normal loads respectively are computed as

$$a_y = C_y\,q_\infty\,\frac{S}{m}$$
$$a_z = C_z\,q_\infty\,\frac{S}{m}$$

3.5.3 State and Control Bounds

The limits in state and control values are often imposed by mission specific consideration or physical limits.

- The inertial angular velocity about the body axes should be within the design

bounds

$$\begin{aligned} |p(t)| &\leq |p_{max}| \\ |q(t)| &\leq |q_{max}| \\ |r(t)| &\leq |r_{max}| \end{aligned} \tag{3.5.7}$$

- There are limits imposed on the angle-of-attack and side-slip angle from aerodynamic and heating considerations.

$$\alpha_{min} \leq \alpha(t) \leq \alpha_{max} \tag{3.5.8}$$

$$\beta_{min} \leq \beta(t) \leq \beta_{max} \tag{3.5.9}$$

- The control deflections must be contained within their prescribed bounds.

$$\delta n_{min} \leq \delta n(t) \leq \delta n_{max} \tag{3.5.10}$$

3.6 Notional Spaceplane

This study uses a notional spaceplane with a blunt nose and physical properties described in Table 3.6.1.

Table 3.6.1: Physical Properties of Notional Spaceplane

Notional Spaceplane Physical Properties [29] [5]	
Mass	3200 kg
Aerodynamic Ref. Area	7.29 m^2
Mean Chord Length	1.78 m
Wingspan	4.57 m
Nose Radius	1.2 m
I_{xx}	0.59×10^6 $kg.m^2$
I_{yy}	1.3×10^6 $kg.m^2$
I_{zz}	1.53×10^6 $kg.m^2$
I_{xz}	0.024×10^6 $kg.m^2$
I_{xy}	$0\,kg.m^2$
I_{yz}	$0\,kg.m^2$

The vehicle-specific limits on state parameters are shown in Table 3.6.2. The angle-of-attack and sideslip angle limits are determined by the data available from the aerodynamic database. From the angular rate limits, it can be inferred that the system is more "sporty" in the pitch plane than in the yaw or roll plane.

Table 3.6.2: State Limits - Notional Spaceplane

Notional Spaceplane State Limits [29]	
Minimum Angle of Attack (α_{min})	55°
Maximum Angle of Attack (α_{max})	-25°
Minimum Angle of Sideslip (β_{min})	18°
Maximum Angle of Sideslip (β_{min})	-18°
Maximum Roll Rate Magnitude	10 deg/s
Maximum Pitch Rate Magnitude	50 deg/s
Maximum Yaw Rate Magnitude	5 deg/s
Maximum Dynamic Pressure	14.4 kPa
Maximum Load Factor	10 g
Maximum Heat Flux	7.95×10^5 W/m^2

There are six control surfaces for the notional spaceplane. They are listed alongside their bounds as follows:

Table 3.6.3: Control Bounds for Notional Spaceplane

Control Bounds for Notional Spaceplane [29]			
Control Surface	Symbol	Lower Limit (°)	Upper Limit (°)
Right Flap	δ_1	-30	30
Left Flap	δ_2	-30	30
Right Tail	δ_3	-30	30
Left Tail	δ_4	-20	30
Body Flap	δ_5	-25	20
Speedbrake	δ_6	35	70

The aerodynamic database for the vehicle is a function of Mach(M), α, β, p, q, r,

$\delta_1, \delta_2, \delta_3, \delta_4.\delta_5$ and δ_6. The aerodynamic model used for this system is described in Equation 3.6.1. It was generated using Missile DATCOM by AFRL/RQQA [29].

$$
\begin{aligned}
C_x &= C_{x0}(M,\alpha) + \Delta C_{x_q}(M,\alpha,q) \\
&\quad + \Delta C_{x_{\delta 1}}(M,\alpha,\delta_1) + \Delta C_{x_{\delta 2}}(M,\alpha,\delta_2) + \Delta C_{x_{\delta 3}}(M,\alpha,\delta_3) \\
&\quad + \Delta C_{x_{\delta 4}}(M,\alpha,\delta_4) + \Delta C_{x_{\delta 5}}(M,\alpha,\delta_5) + \Delta C_{x_{\delta 6}}(M,\alpha,\delta_6) \\
C_y &= C_{y0}(M,\alpha) + \Delta C_{y_r}(M,\alpha,r) + \Delta C_{y_p}(M,\alpha,p) + \Delta C_{y_\beta}(M,\alpha,\beta) \\
&\quad + \Delta C_{y_{\delta 1}}(M,\alpha,\delta_1) + \Delta C_{y_{\delta 2}}(M,\alpha,\delta_2) + \Delta C_{y_{\delta 3}}(M,\alpha,\delta_3) \\
&\quad + \Delta C_{y_{\delta 4}}(M,\alpha,\delta_4) + \Delta C_{y_{\delta 5}}(M,\alpha,\delta_5) \\
C_z &= C_{z0}(M,\alpha) + \Delta C_{z_q}(M,\alpha,q) \\
&\quad + \Delta C_{z_{\delta 1}}(M,\alpha,\delta_1) + \Delta C_{z_{\delta 2}}(M,\alpha,\delta_2) + \Delta C_{z_{\delta 3}}(M,\alpha,\delta_3) \\
&\quad + \Delta C_{z_{\delta 4}}(M,\alpha,\delta_4) + \Delta C_{z_{\delta 5}}(M,\alpha,\delta_5) + \Delta C_{z_{\delta 6}}(M,\alpha,\delta_6) \\
C_l &= C_{l0}(M,\alpha) + \Delta C_{l_r}(M,\alpha,r) + \Delta C_{l_p}(M,\alpha,p) + \Delta C_{l_\beta}(M,\alpha,\beta) \\
&\quad + \Delta C_{l_{\delta 1}}(M,\alpha,\delta_1) + \Delta C_{l_{\delta 2}}(M,\alpha,\delta_2) + \Delta C_{l_{\delta 3}}(M,\alpha,\delta_3) \\
&\quad + \Delta C_{l_{\delta 4}}(M,\alpha,\delta_4) + \Delta C_{l_{\delta 5}}(M,\alpha,\delta_5) \\
C_m &= C_{m0}(M,\alpha) + \Delta C_{m_q}(M,\alpha,q) + \Delta C_{m_\beta}(M,\alpha,\beta) \\
&\quad + \Delta C_{m_{\delta 1}}(M,\alpha,\delta_1) + \Delta C_{m_{\delta 2}}(M,\alpha,\delta_2) + \Delta C_{m_{\delta 3}}(M,\alpha,\delta_3) \\
&\quad + \Delta C_{m_{\delta 4}}(M,\alpha,\delta_4) + \Delta C_{m_{\delta 5}}(M,\alpha,\delta_5) + \Delta C_{m_{\delta 6}}(M,\alpha,\delta_6) \\
C_n &= C_{n0}(M,\alpha) + \Delta C_{n_r}(M,\alpha,r) + \Delta C_{n_p}(M,\alpha,p) + \Delta C_{n_\beta}(M,\alpha,\beta) \\
&\quad + \Delta C_{n_{\delta 1}}(M,\alpha,\delta_1) + \Delta C_{n_{\delta 2}}(M,\alpha,\delta_2) + \Delta C_{n_{\delta 3}}(M,\alpha,\delta_3) \\
&\quad + \Delta C_{n_{\delta 4}}(M,\alpha,\delta_4) + \Delta C_{n_{\delta 5}}(M,\alpha,\delta_5)
\end{aligned}
\tag{3.6.1}
$$

3.6.1 Neural Network Approximation for Aerodynamic Database

As mentioned in Section 2.7, a series of neural networks is used to approximate each increment in Tables 2.6.1, 2.6.2 and 2.6.3. Table lookups are numerically expensive and does not guarantee a continuous derivative [17]. MATLAB's Deep Learning Toolbox is used to develop a Neural Network for each increments as a function of their independent variable. The neural network is then extracted as a function using the `genFunction` command and applied to Equation 3.6.1 to compute the coefficients. In effect, each coefficient is expressed as a function evaluation of the respective inputs instead of a tedious set of table-lookups.

Two hidden layers were chosen to approximate each increment. There were 6 nodes in each hidden layer. These choices were made based on trial and error by evaluating the fit quality and the time elapsed to build a neural network. The activation functions used at each layer is specified by the user. If the activation functions are chosen to be continuous and differentiable, the neural networks will be too [22]. Hence, the hyperbolic tangent sigmoid (`tansig`) function and radial basis (`radbas`) function are chosen to be the activation function for each of the hidden layers. Each of them are defined as follows:

$$\texttt{tansig}(n) = \frac{2}{(1+e^{-2n})} - 1 \tag{3.6.2}$$

$$\texttt{radbas}(n) = e^{-n^2} \tag{3.6.3}$$

The Deep Learning Toolbox uses the aerodynamic database to train the neural network based on a training algorithm also provided by the user. The training algorithm iterates on the values for weights and biases at nodes until the ANN is able to approximate the data to a specified certainty. MATLAB has availed a host of training algorithms. The Bayesian Regularization algorithm (`trainbr`) was chosen to estimate all the increments.

Comparison between the truth data and the estimated data using neural nets are shown in Section 4.2.

3.7 Test Scenario - The Pullup Maneuver

For this study, a pullup maneuver is considered. The maneuver is characterized by a gradual uplift of the flight path angle (γ) close to zero degrees from an initially negative angle, at a desired altitude. There is no out-of-plane motion desired at this phase.

3.7.1 Initial State

The initial conditions are defined as follows:

Table 3.7.1: Initial State for Test Case

Initial State for Test Case		
	Value	Units
Initial Altitude (h_0)	65	km
Initial Longitude (ϕ_0)	0	deg
Initial Latitude (λ_0)	0	deg
Initial Speed (v_0)	4.8	km/s
Initial Heading (ψ_0)	0	deg
Initial Flight-path Angle (γ_0)	-7	deg
Initial Roll-Rate (p_0)	0	deg/s
Initial Pitch-Rate (q_0)	0.5	deg/s
Initial Yaw-Rate (r_0)	0	deg/s
Initial Bank Angle (σ_0)	0	deg
Initial Angle of Attack(α_0)	2	deg
Initial Angle of Sideslip (β_0)	0	deg

3.7.2 Final State

The desired final state is tabulated in Table 3.7.2.

Table 3.7.2: Desired Final State for Test Case

Desired Final State for Test Case		
	Nominal Value	Units
Final Altitude (h_f)	45	km
Final Longitude (ϕ_f)	2	deg
Final Latitude (λ_f)	0	deg
Final Speed (v_f)	3.8	km/s
Final Heading (ψ_f)	$free$	deg
Final Flight-path Angle (γ_f)	0	deg
Final Roll-Rate (p_f)	$free$	deg/s
Final Pitch-Rate (q_f)	$free$	deg/s
Final Yaw-Rate (r_f)	$free$	deg/s
Final Bank Angle (σ_f)	$free$	deg
Final Angle of Attack (α_f)	$free$	deg
Final Angle of Sideslip (β_f)	$free$	deg

There are no mission-specific requirements imposed at the final value of states described "free" in Table 3.7.2. However, they are subject to the vehicle-specific state limits listed in Table 3.6.2.

3.7.3 Final State Error Tolerances

For the optimal control solver, targeting to the precise final state may be impractical. Therefore, an error criterion is established for each state's final value. They are tabulated as follows:

Table 3.7.3: Error Bounds for Final State

Error Bounds for Final State		
	Error Tolerance	Units
Final Altitude (h_f)	± 0.25	km
Final Longitude (ϕ_f)	$\pm 4 \times 10^{-5}$	deg
Final Latitude (λ_f)	$\pm 4 \times 10^{-5}$	deg
Final Speed (v_f)	$\pm$ 5	m/s
Final Heading (ψ_f)	$\pm$ 2	deg
Final Flight-path Angle (γ_f)	$\pm$ 2	deg

The tolerances for latitude and longitude were chosen to correspond to an error that matches the altitude tolerance in the local horizontal plane. The set tolerances ensure that the final position will be in a circle centered at the desired coordinates with radius of approximately 4 meters.

3.7.4 Optimal Control Solver Settings

The settings used for the optimal control solver GPOPS is listed below:

Table 3.7.4: Default GPOPS Solver Settings for Test Problem

Relevant Default GPOPS Solver Settings	
Mesh Method	hp-LiuRao-Legendre
Mesh Error Tolerance	1×10^{-4}
Non Linear Programming Solver	snopt
NLP Solver Tolerance	1×10^{-6}
Derivative Supplier	sparseFD
Derivative Order	second
Derivative Dependencies	sparseNaN
Scaling Method	automatic bounds

GPOPS feeds a multi-dimensional mesh generated from problem parameters into a non-linear programming solver such as SNOPT or IPOPT. GPOPS iterates on that mesh until the NLP Solver is able to find the optimal solution within the error specified. After each iteration, the mesh is refined based on specific methods [8]. The method "hp-LiuRao-Legendre" was chosen for this study. This method "employs orthogonal collocation at Legendre-Gauss-Radau points, and adjusts both the mesh size and the degree of the approximating polynomials in the refinement process [30]".

The Non-Linear Program (NLP) is based on a mesh generated by placing collocation points computed from roots of Legendre polynomials. The state derivatives are approximated using a sparse Forward Difference method. Lagrange polynomials are used to interpolate between the collocation points and a Gaussian quadrature is used to compute integrals [30]. The NLP is fed to a solver such as SNOPT or IPOPT (See Section 4.8.1).

IV. Results and Analysis

4.1 Preamble

The comparison between the aerodynamic database and the neural network approximation is explored in Section 4.2. Subsequently, the equations of motion derived in Section 3.3 are verified using results from Bollino's dissertation [5]. Later, a predefined control profile for the notional spaceplane is used to validate the simulation environment developed in Section 3.4. The stated control profile is used as an initial guess to seek an optimal control profile for the same initial and final conditions to evaluate cost functions discussion in Section 3.5.1. In the final section the sensitivity of the solution to varying solver settings and mesh tolerance is studied.

4.2 Approximation of Aerodynamic Data Using Neural Networks

As mentioned in Section 3.6.1, a collection of Artificial Neural Networks were developed to estimate the notional spaceplane's aerodynamic database. This section outlines the comparison between the interpolated values from the database and the neural network approximation for each increment with respect to a dozen independent variables.

An input vector $\hat{u}$ is defined that contains all the independent variables required to look-up the base coefficients and increments from the aerodynamic database.

$$\hat{u} = [M, \alpha, \beta, p, q, r, \delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6] \tag{4.2.1}$$

The minimum and maximum limits for each elements of the input vector are listed in Tables 3.6.2 and 3.6.3. The aerodynamic data spans from Mach 0.1 to Mach 20. Within the defined operating limits, each element of the input vector is randomly

populated 250 times. For each draw of the random input vector, each of the force and moment coefficients are computed from the aerodynamic database. For the same draw, the ANN approximation for the coefficients is also computed. The error between the "truth" data and the ANN approximation is compared in (Figure 4.2.1).

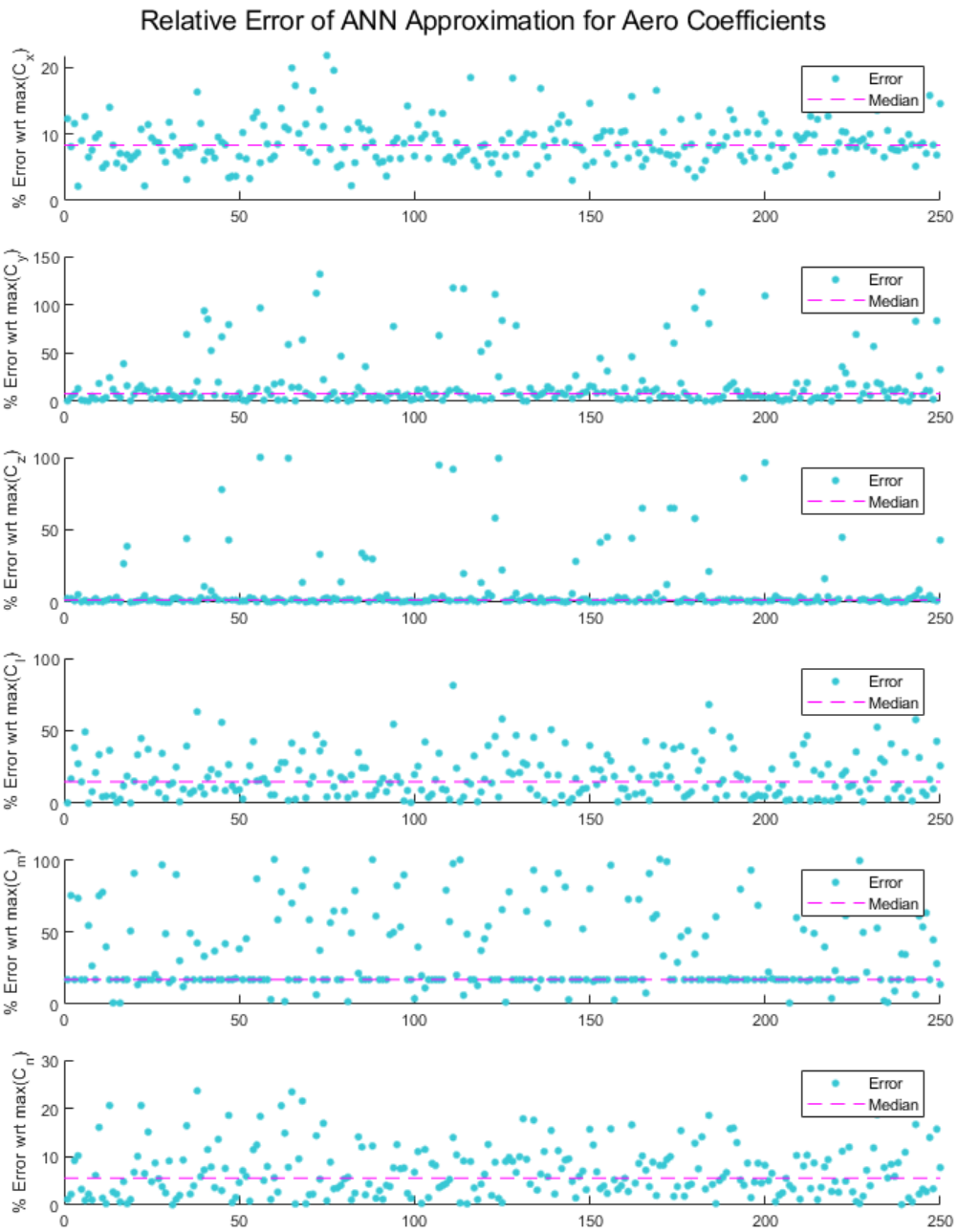

Figure 4.2.1: Error Analysis of Neural Net Approximations

In Figure 4.2.1, the absolute value of percent error, with respect to the highest

possible value for each coefficients is plotted for each 250 draws. The magenta line indicates the median error for each coefficients, in percent values as tabulated in the Table 4.2.1.

Table 4.2.1: Median Error From ANN Approximation of Aero Database

Aero Coefficient	Median Error (%)
C_x	8.3
C_y	8.0
C_z	1.6
C_l	14.8
C_m	17.0
C_n	5.6

The "truth data",i.e, the aerodynamic database which is based on numerous wind tunnel experiments and computational analysis, is subject to large uncertainty. The error bars are reported to be as high as 43% [5]. The maximum error deviation of the ANN approximation from the aerodynamic database is 17 %, as reported in Table 4.2.1.

4.3 Simulation Results with Sub-Optimal Control

A sub-optimal control is found by trial and error that accomplishes a pull-up maneuver described in Section 3.7. The control profile is shown in Figure 4.3.1.

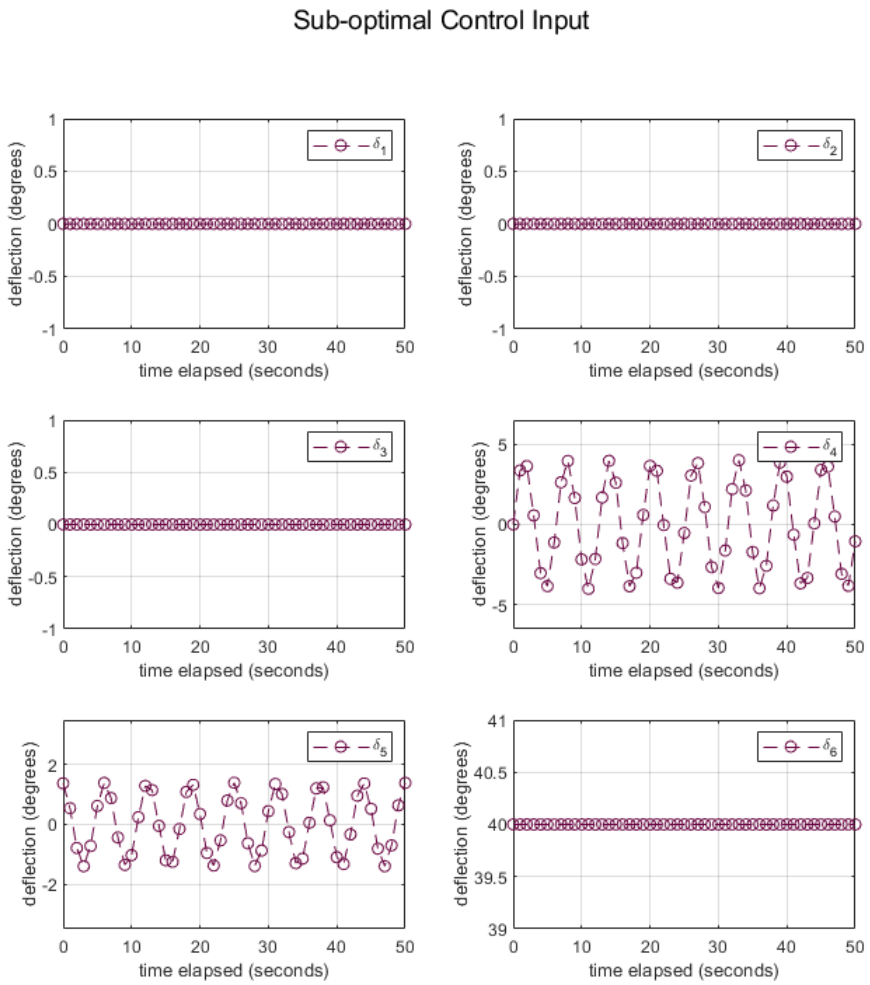

Figure 4.3.1: Sub-optimal Control Profile

The simulation output of the 50-second long control input is depicted in Figure 4.3.2.

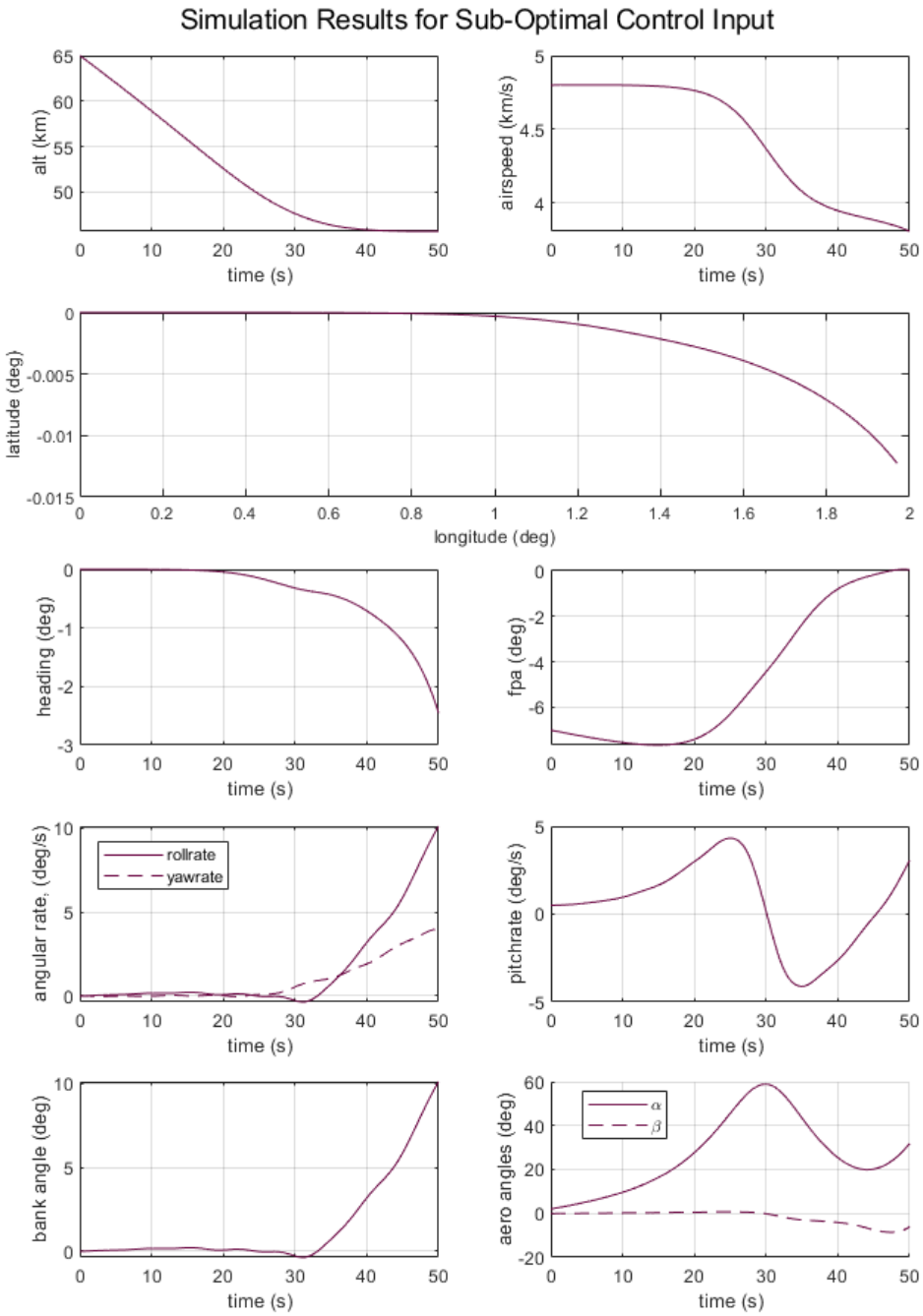

Figure 4.3.2: Simulation Results with sub-optimal Control Input

It can be verified from Figure 4.3.2 that the "pull-up" is executed and all the state constraints described in Section 3.6.2 are met. The maneuver takes exactly 50 seconds. In the following plot, the path constraints described in Section 3.5.2 are computed and compared against prescribed limits.

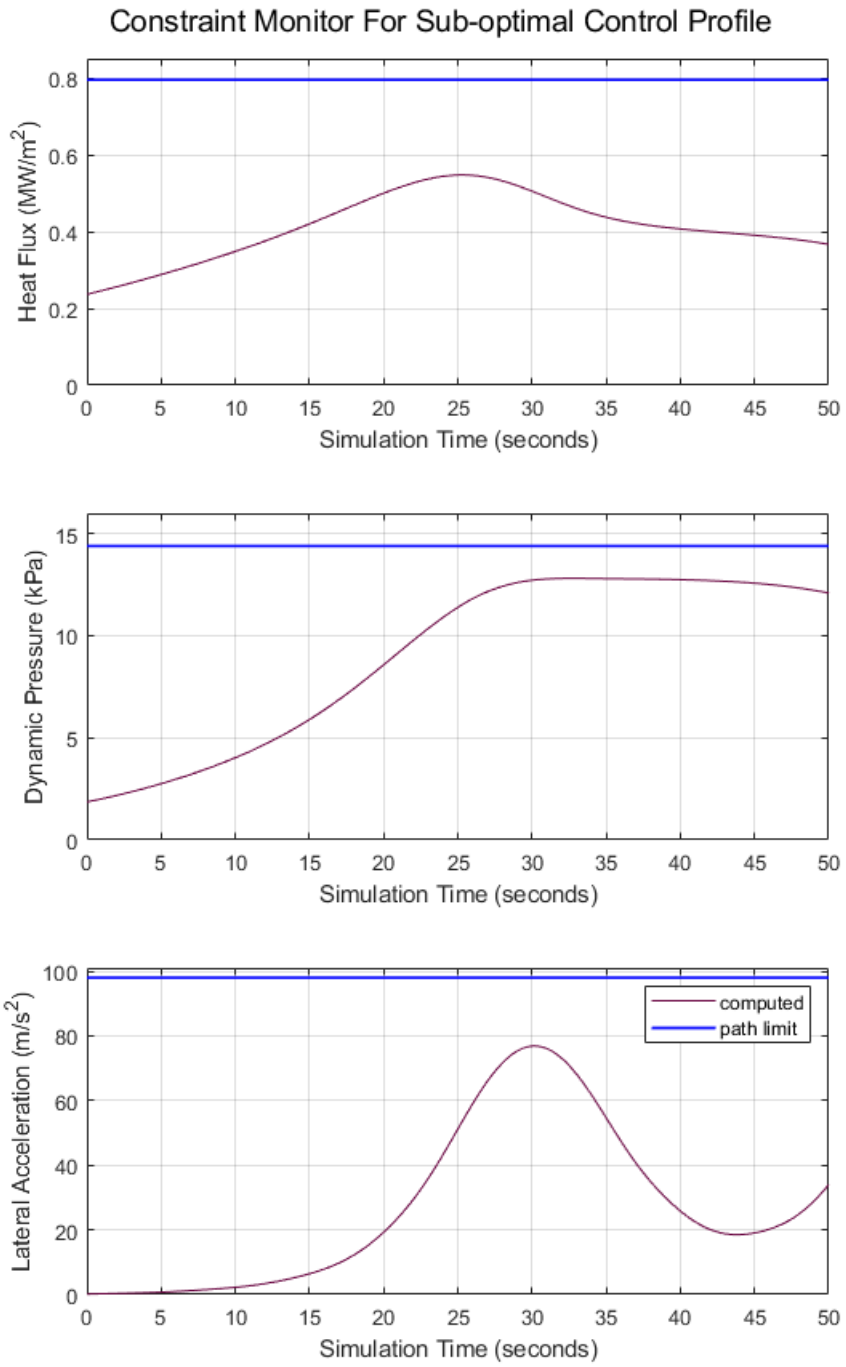

Figure 4.3.3: Path Constraints for sub-optimal Control Input

4.4 Verification of Derived Equations of Motion

The equations of motion that describe the reentry dynamics in a full 6-DoF sense were derived in Section 3.3. They are recapped in Figure 3.4.1. Equations 3.3.4, 3.3.10 and 3.3.12 can be verified based on Hicks [2], Bollino [5] and Etkin [18]. Although, equations that describe the rates of change of $\dot{\sigma}, \dot{\alpha}$, and $\dot{\beta}$ were derived for this work. The approximations from Bollino [5] were used to verify Equations 3.3.29 in Section 4.4.1.

4.4.1 Bollino's Equation of Motion for Aerodynamic Angle Rates

The 6-DoF equations derived by Bollino [5] assume a flat-Earth and ignore effects caused by Earth's rotation. Moreover, the formulation assumes small roll and bank angles.

$$\begin{aligned} \dot{\sigma} &\approx \frac{C_L \rho v S}{2} \frac{\sin(\sigma)}{m} \tan(\gamma) + p\cos(\alpha) + r\sin(\alpha) \\ \dot{\alpha} &\approx q - \dot{\gamma} \\ \dot{\beta} &\approx p\sin(\alpha) - r\cos(\alpha) \end{aligned} \tag{4.4.1}$$

The sub-optimal control profile used in Section 4.3 is used to simulate the trajectory. Equations for $\dot{\sigma}, \dot{\alpha}$ and $\dot{\beta}$ in Figure 3.4.1 is replaced with Equations 4.4.1. The resulting trajectory is plotted in Figure 4.4.1, overlaid with the trajectory from Figure 4.3.2. The error between the states for each simulation is plotted in Figure 4.4.1. There is little difference between the two trajectories simulated as evident from Figure 4.4.1. The error between the two suites of reentry dynamics is negligible for altitude, longitude, latitude, speed, flight path angle and heading. As Figure 4.4.2 illustrates, the error for the rest of the six states is negligible at the beginning of the simulation

and grows with time. The assumptions made by Bollino such as the premise of a non-rotating flat Earth may explain the difference. When the following assumptions are made:

$$\begin{aligned} \omega_{\oplus} &= 0 \\ \dot{\mu} &= 0 \\ \dot{\lambda} &= 0 \\ \dot{\psi} &= 0 \\ \dot{\lambda} &= 0 \\ \beta &= 0 \\ \eta &= 0 \\ \dot{\eta} &= 0 \end{aligned} \tag{4.4.2}$$

Equation 3.3.29 reduces to:

$$\begin{aligned} \dot{\sigma} &\approx \frac{C_x r - C_z p + (C_x \cos(\alpha) + C_z \sin(\alpha))\dot{\gamma} \sin(\sigma)}{C_z \cos(\alpha) - C_x \sin(\alpha)} \\ \dot{\alpha} &\approx q - \dot{\gamma} \cos(\sigma) \\ \dot{\beta} &\approx p \sin(\alpha) - r \cos(\alpha) - \dot{\gamma} \sin(\sigma) \end{aligned} \tag{4.4.3}$$

Assuming a negligible bank angle ($\sigma = 0$) in Equation 4.4.3, Equation 3.3.29 is simplified to

$$\begin{aligned} \dot{\sigma} &\approx \frac{C_x r - C_z p}{C_z \cos(\alpha) - C_x \sin(\alpha)} \\ \dot{\alpha} &\approx q - \dot{\gamma} \\ \dot{\beta} &\approx p \sin(\alpha) - r \cos(\alpha) \end{aligned} \tag{4.4.4}$$

The expressions for $\dot{\alpha}$ and $\dot{\beta}$ in Bollino's formulation are identical to reduced in 4.4.4. The force equations of motion in this study are formulated in terms of coefficients in the body frame (C_x, C_y, C_z). Bollino's equations express the coefficients in Flight-Path Frame (C_L, C_D).

When formulated in terms of lift and drag coefficients, the terms for $\dot{\sigma}$ must be equivalent. One maay also notice that the errors grow with time as Earth's rotation and other assumptions from Equation 4.4.2 takes effect.

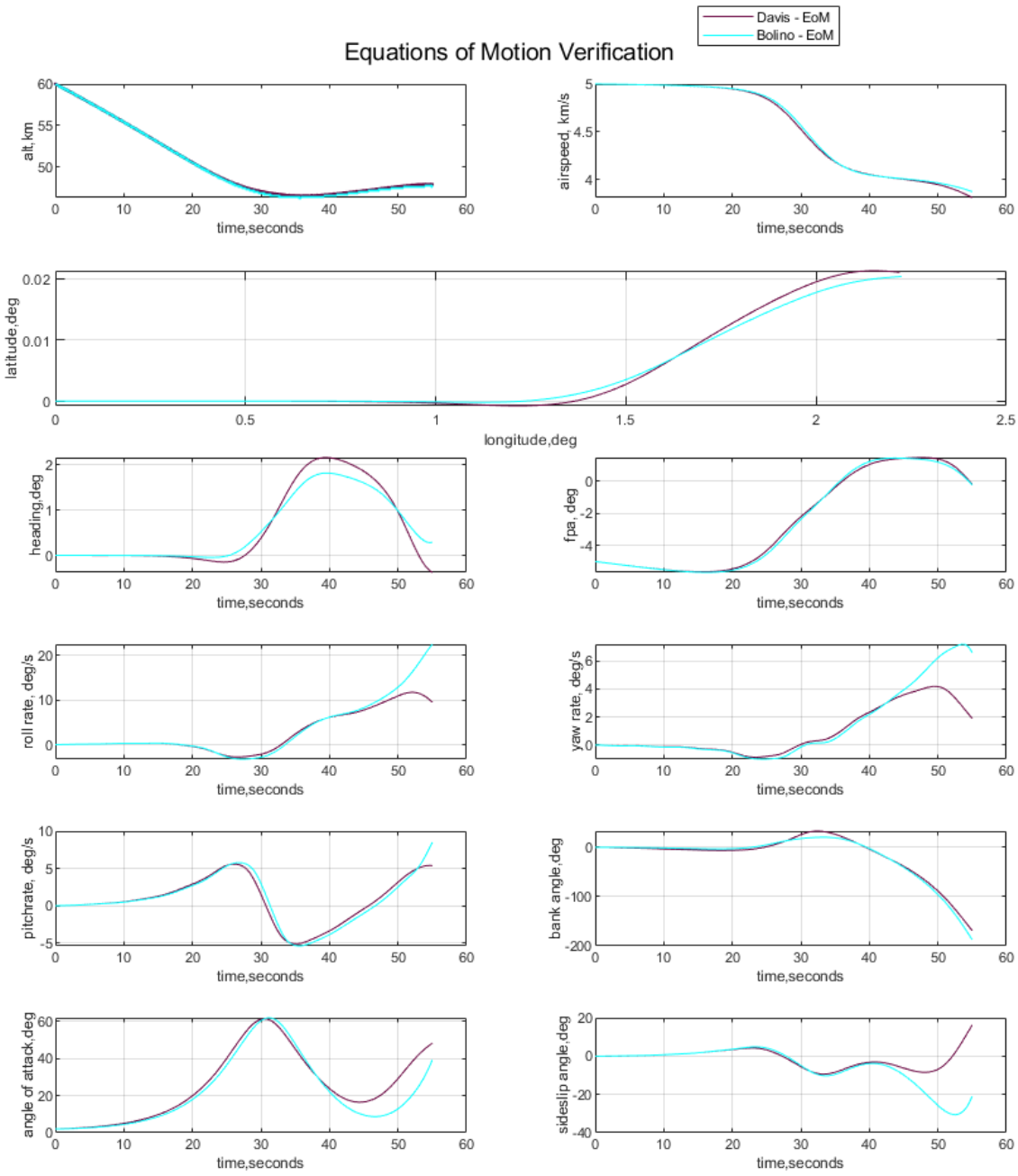

Figure 4.4.1: Output Trajectory using Bollino's 6-DoF Equations of Motion

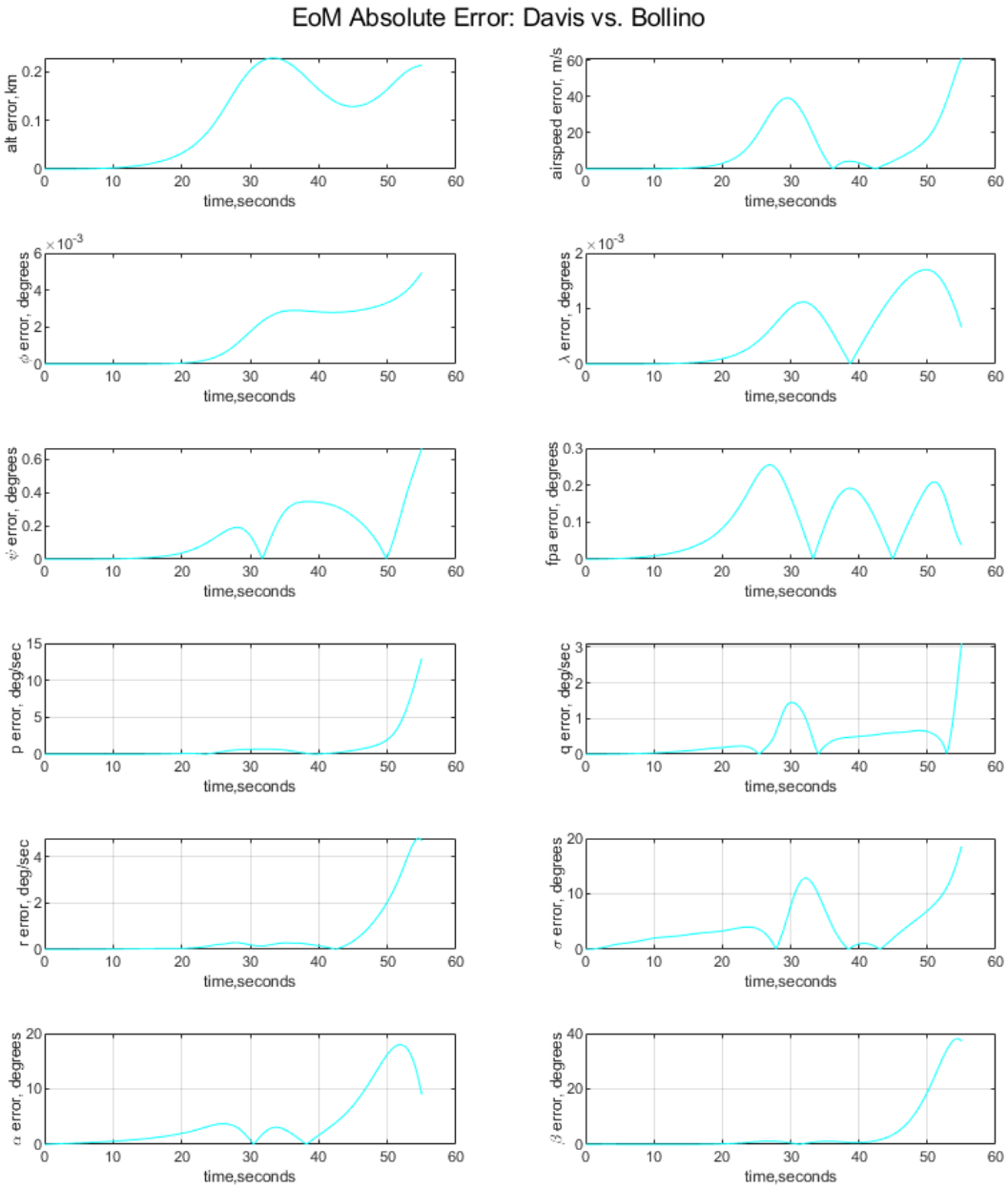

Figure 4.4.2: Error in States between 6-DoF Equations of Motion

4.5 Optimal Control Results for Minimum Time Trajectory

Using Equation 3.5.1 as the cost function, a minimum time control solution is investigated. The sub-optimal control profile and the resulting trajectory is ingested as an initial guess. The optimal control profile for the minimum time trajectory with specifications for initial state, final state and final state tolerances described in Section 3.7 is sought. GPOPS-specific inputs used are listed in Table 3.7.4. The computed optimal control profile is shown in Figure 4.5.1: the brute force control input is also shown for comparison. The optimal time is 49.6 seconds as opposed to 50 seconds in the sub-optimal simulation.

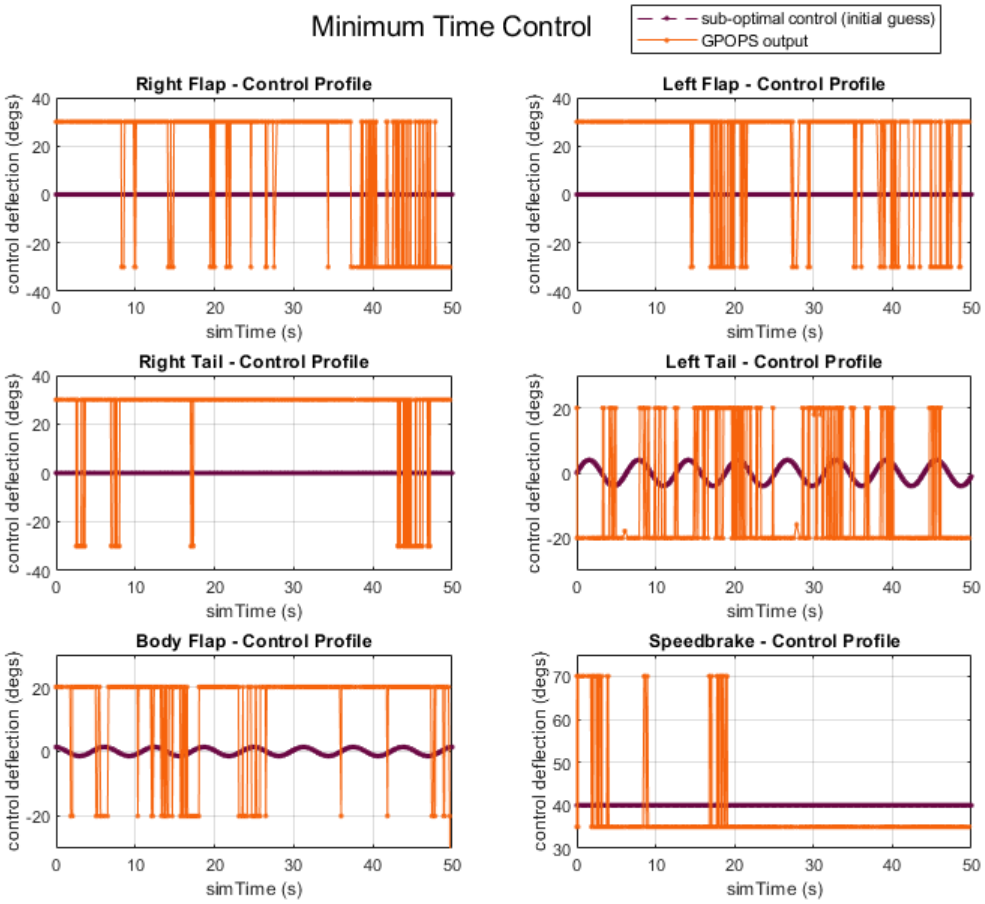

Figure 4.5.1: Computed Optimal Control for Base Scenario

It can be observed from the Figure 4.5.1 that the optimal control for each deflection resembles bang-bang behavior. For the most part, each deflection switches between the maximum value and minimum value. The simulation output for the computed control profile is depicted in Figure 4.5.2. It could be inferred that there is no appreciable difference in the state trajectories between the two different controls tested. It is easily verified from Figures 4.5.2 and 4.5.3 that the path and state constraints are not violated.

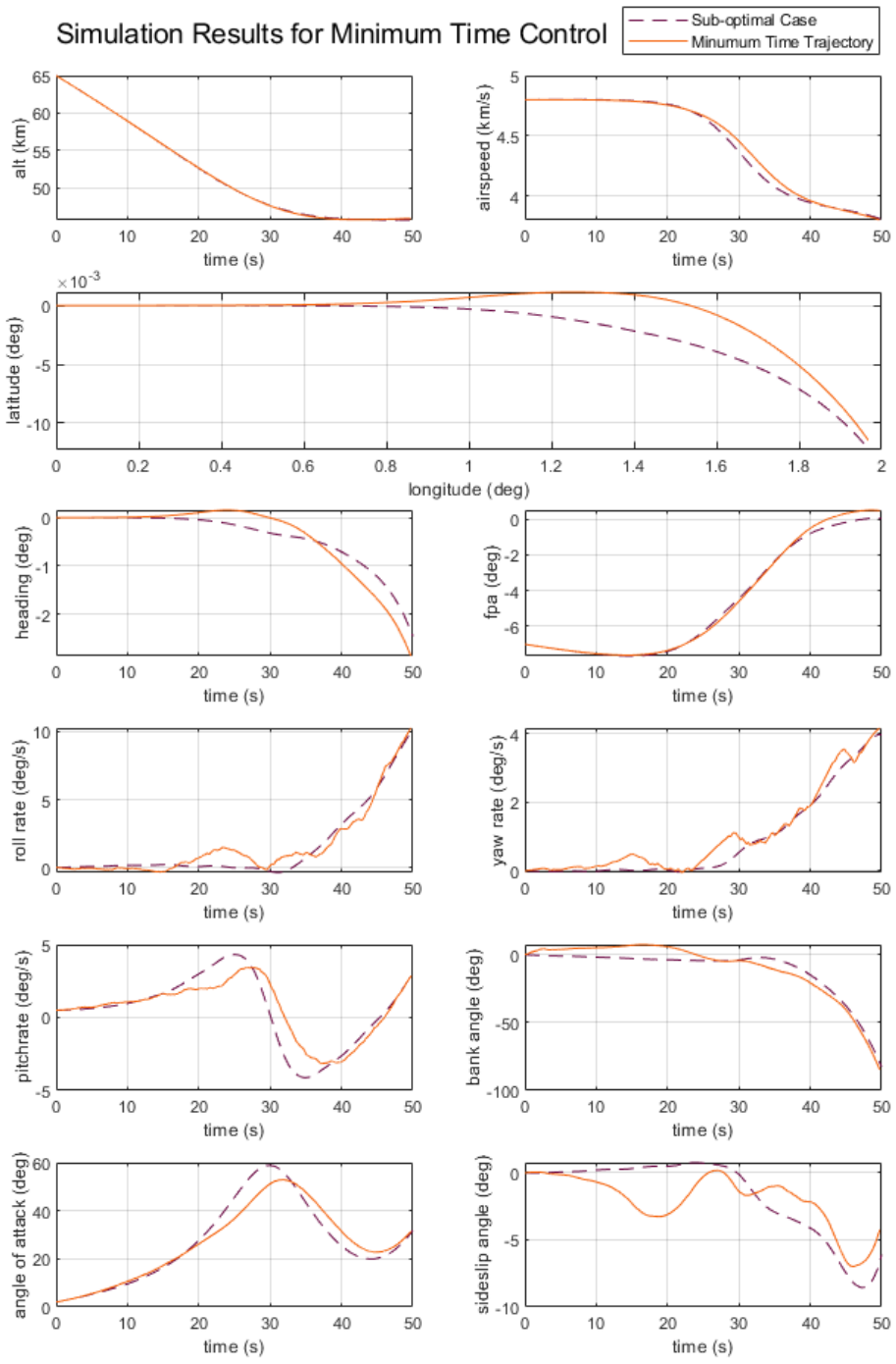

Figure 4.5.2: Simulation Results with Optimal Control - Base Scenario

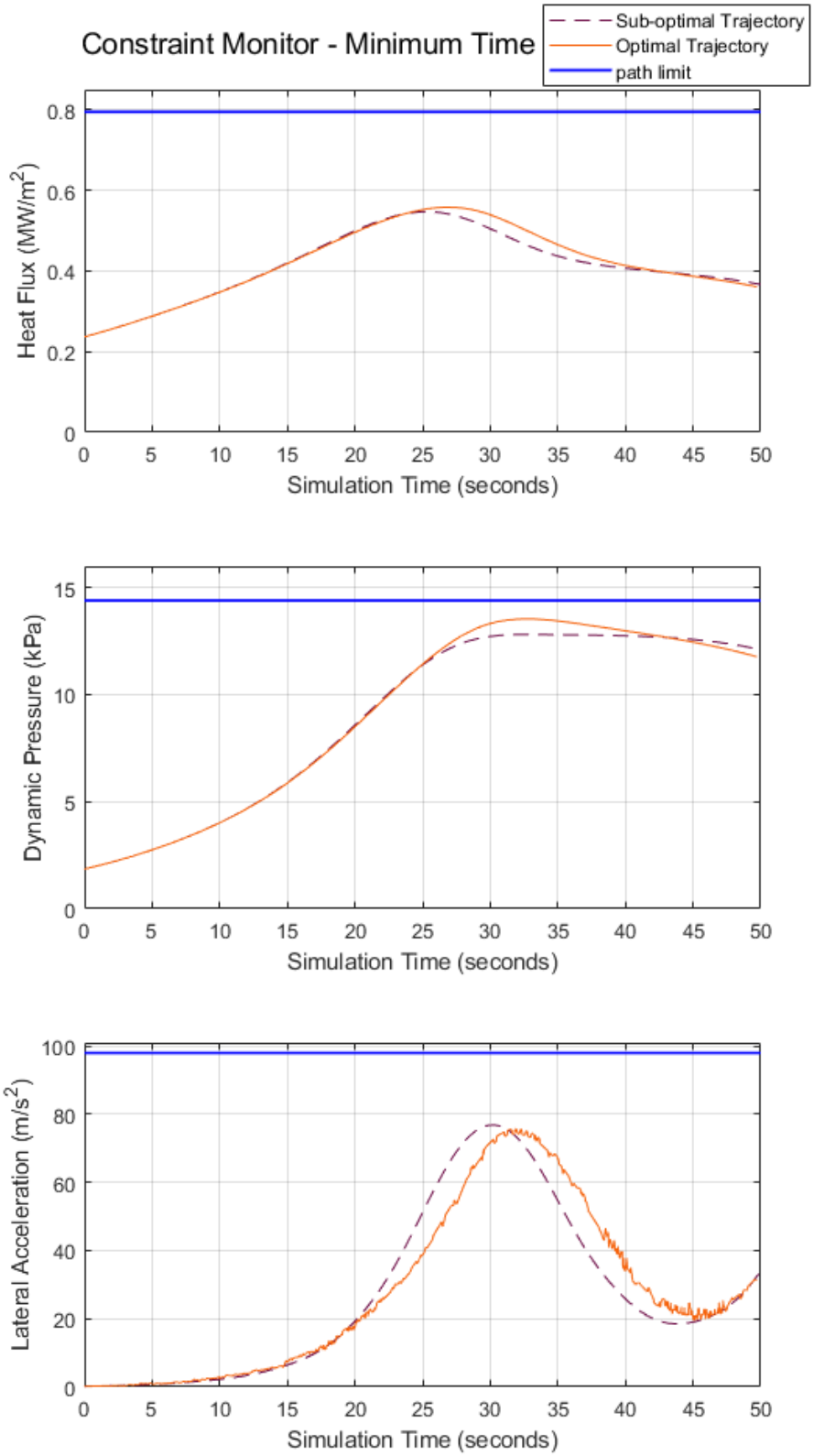

Figure 4.5.3: Path Constraint Monitor - Base Scenario

4.6 Optimal Control Results With Final State Stabilization

The desired final state of the last six state variables were set to free in the sub-optimal scenario outlined in Table 3.7.2. As shown in Figure 4.5.2 , this lead to high magnitudes of roll rate, yaw rate, pitch rate and bank angle towards the end of the simulation time. To stabilize the final state, an optimal control profile is sought with new final state boundary condition for the following states:

Table 4.6.1: Desired State Conditions for Final State Stabilization

Desired State Conditions for Final State Stabilization			
	Nominal Value	Error Tolerance	Units
Final rollrate (p'_f)	0	$\pm$ 2	deg/s
Final pitchrate (q'_f)	0	$\pm$ 2	deg/s
Final yawrate (r'_f)	0	$\pm$ 2	deg/s
Final Bank Angle (σ'_f)	0	$\pm$ 2	deg
Final Angle of Attack(α'_f)	0	$\pm$ 2	deg
Final Angle of Sideslip (β'_f)	0	$\pm$ 2	deg

The control input for minimum time trajectory is sought. Therefore, the cost function is the same as in Section 4.5. The tolerances on the newly bounded final state terms are also listed in 4.6.1. The optimal control profile is plotted in Figure 4.6.1. The optimal time computed, in this case, hasn't changed drastically. The simulation time is 49.7 seconds. Similar to Section 4.5, the optimal control policy emulates a bang-bang behavior.

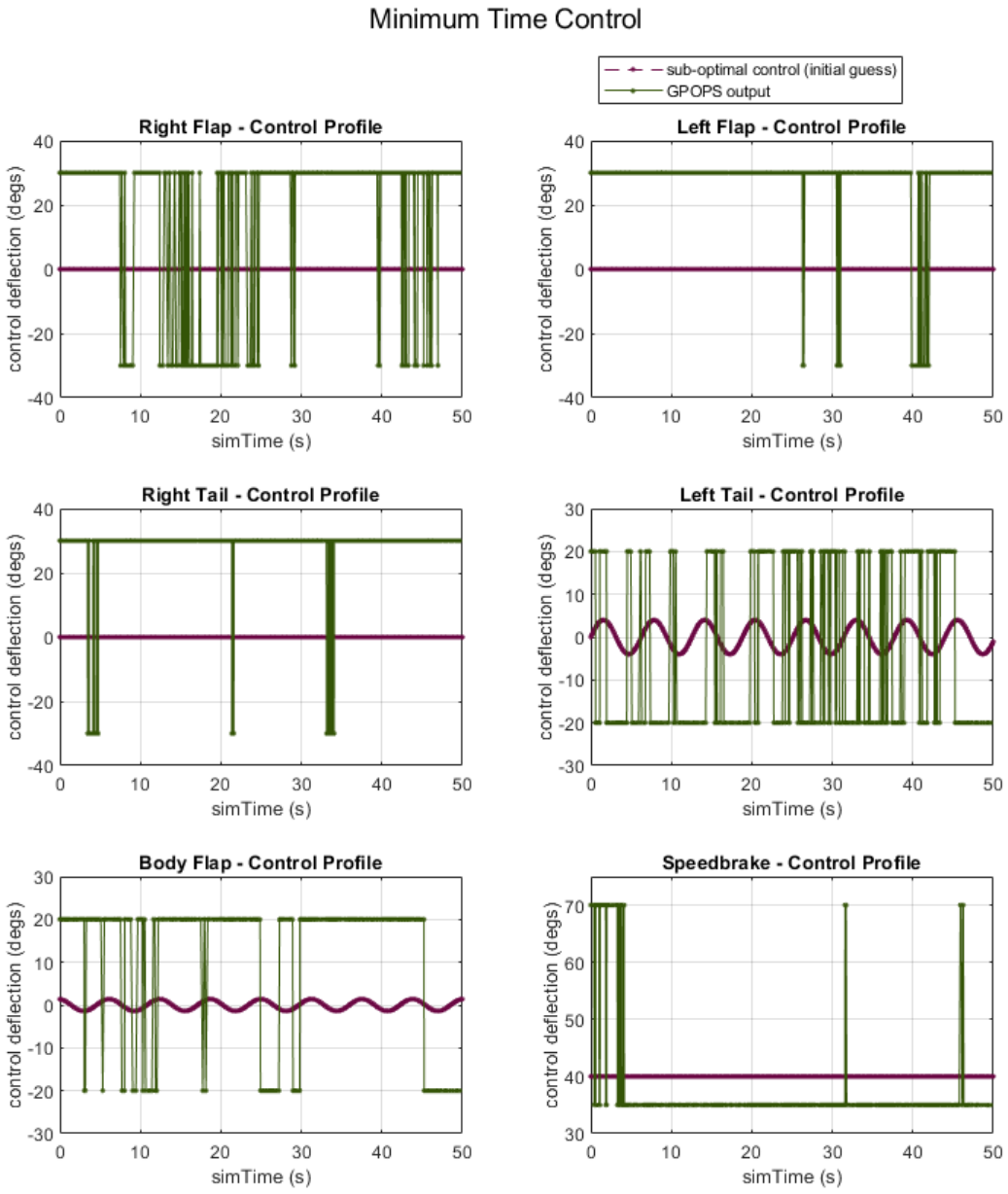

Figure 4.6.1: Computed Optimal Control: Base Scenario for Stabilized Final State

The simulation results for the optimal control is illustrated in Figure 4.6.2. It can be seen that the desired values for states listed in Table 4.6.1 are achieved within defined error tolerances. From Figure 4.6.3, it can be seen that the resultant trajectory

clearly avoided crossing over the path constraints.

It can be seen from Figure 4.6.3 that the optimal control profile drives the dynamic pressure to its maximum allowable limit while seeking to satisfy the new constraints.

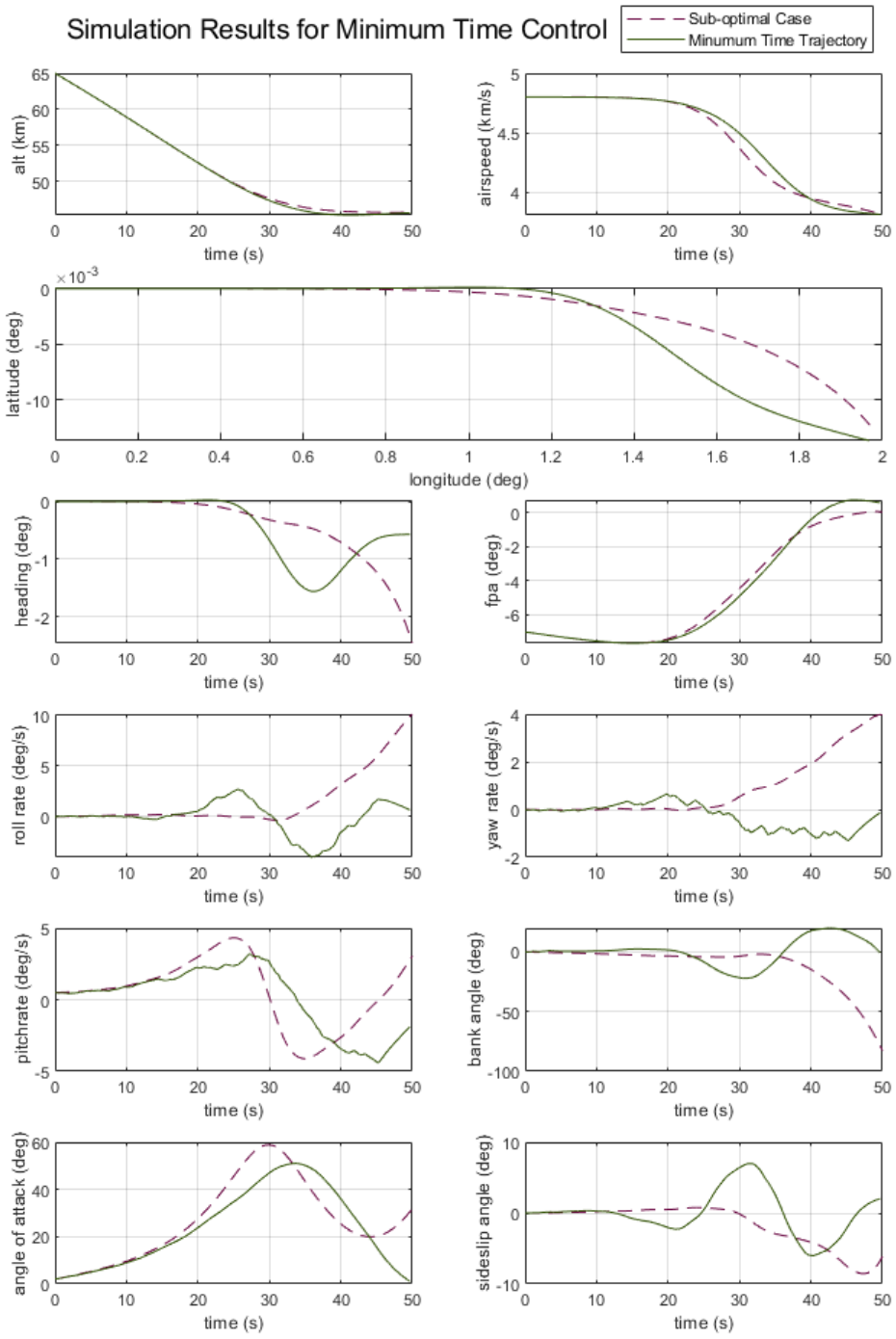

Figure 4.6.2: Simulation Results: Base Scenario for Stabilized Final State

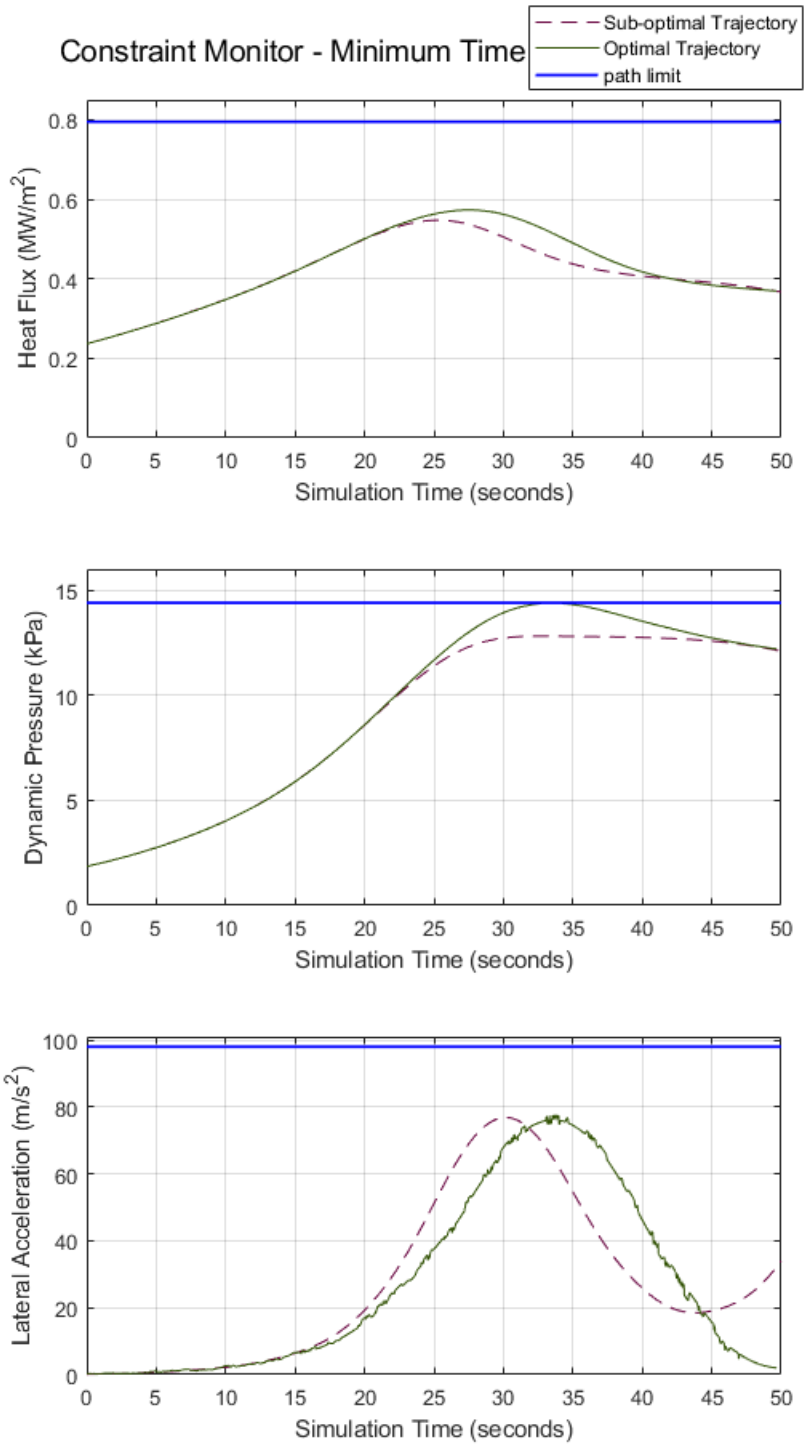

Figure 4.6.3: Path Constraint Monitor: Base Scenario for Stabilized Final State

4.7 Optimal Control For Minimum Control Trajectory

For the minimum control trajectory, the following cost function is evaluated:

$$J_2 = \int_{t_0}^{t} \mathbf{u^T u}\ \mathbf{dt}$$

where $\mathbf{u}$ is the control vector at each time-step. The optimal control profile computed in Sections 4.5 and 4.6 can be described as "rugged". It is unrealistic for a control surface to toggle between two extremes instantly as prescribed by Figures 4.5.1 and 4.6.1. By imposing a minimum control cost function, the goal is to smoothen out the control profile. The tolerances on the final state is not varied from Table 3.7.3. The optimal control profile is plotted in Figure 4.6.1. The control profile computed by GPOPS for this case is illustrated in Figure 4.7.1.

GPOPS converged to a solution within the specified tolerances. However, the control profile did not smooth out as expected.

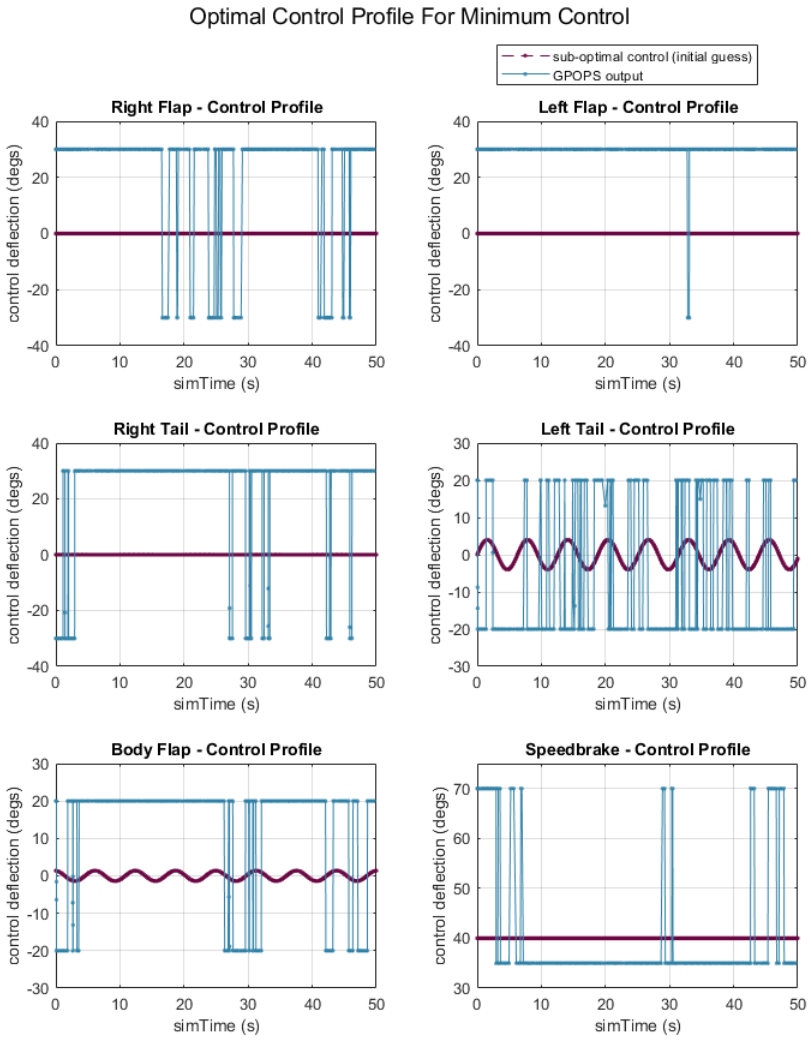

Figure 4.7.1: Computed Optimal Control: Minimum Control Case

The history of states and the path constraints compared to the sub-optimal scenario are shown in Figures 4.7.2 and 4.7.3 respectively.

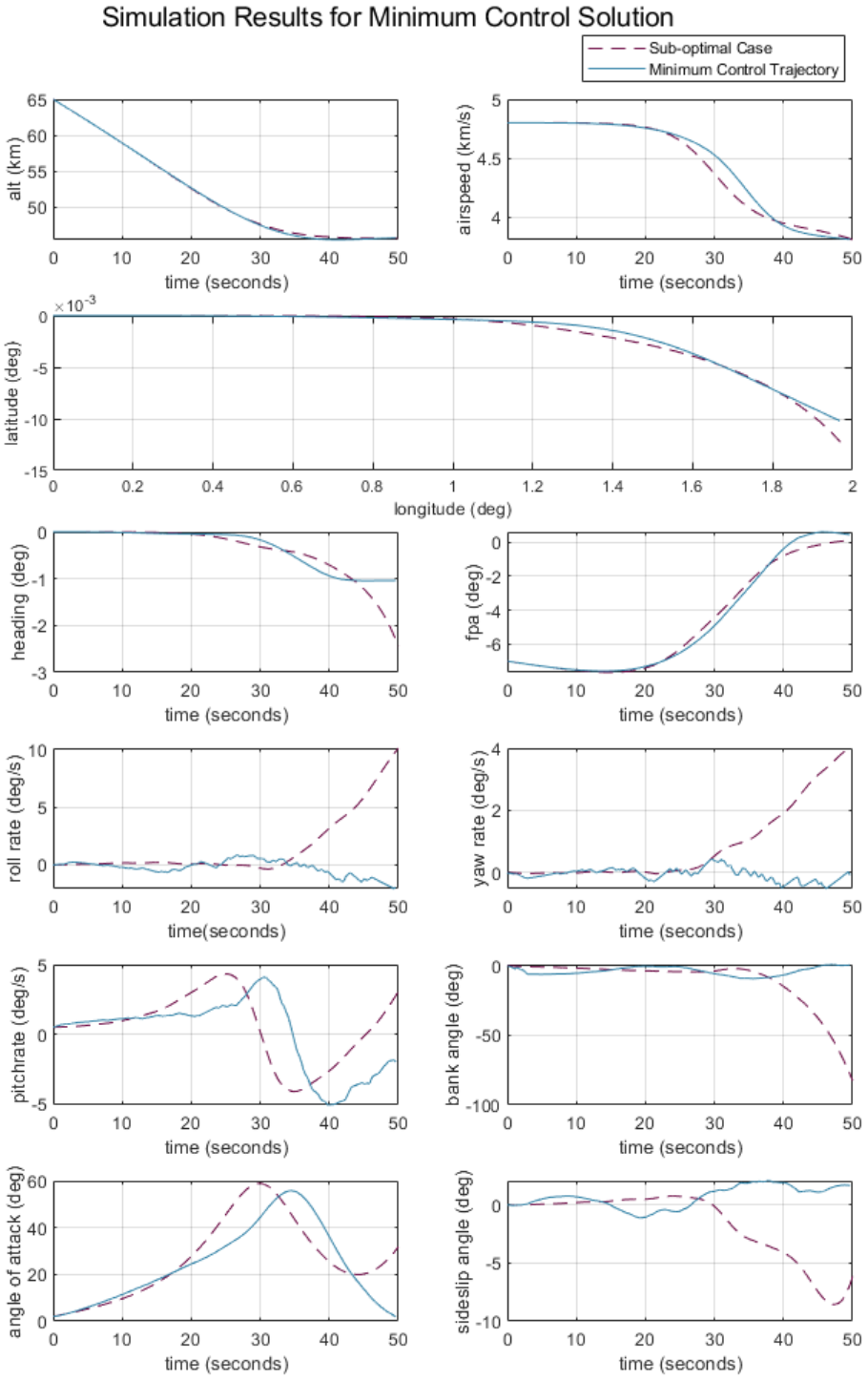

Figure 4.7.2: Simulation Results: Minimum Control

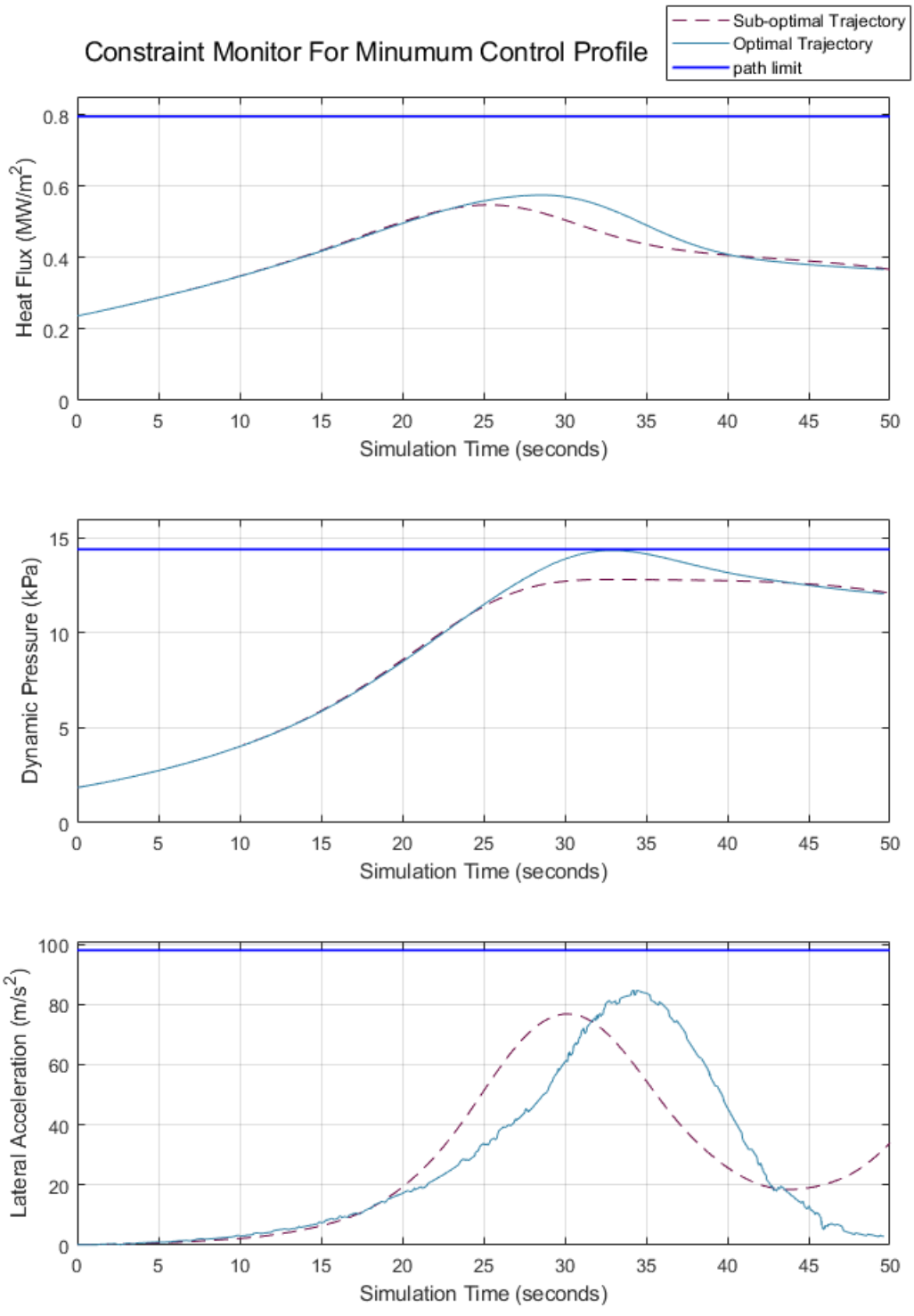

Figure 4.7.3: Path Constraint Monitor: Minimum Control

4.8 Sensitivity Study on GPOPS Parameters

For each problem, GPOPS requires a set of parameters as inputs. The default values used for each of them in this study are outlined in Table 3.7.4. A sensitivity study was undertaken to evaluate the sensitivity of the output solution to the choice of NLP solver and mesh error tolerance. This sensitivity study used the same parameters from Section 4.5.

4.8.1 Solver: snopt vs. ipopt

GPOPS has two different options for Non-Linear Programming (NLP) solvers: SNOPT and IPOPT. SNOPT(Sparse Nonlinear OPTimizer) uses a Quasi-Newton Sequential Quadratic Programming (SQP) method to converge on a solution. It is developed and licensed by Stanford University. IPOPT (Interior Point Optimizer) uses an interior point method to optimize a given problem. SNOPT was used to solve problems in Sections 4.5, 4.6 and 4.7. However, GPOPS did not converge to a solution when IPOPT was applied to the problem within the given number of iterations. A possible reason could be that IPOPT does not take advantage of an initial guess, as opposed to SNOPT which converges quickly with a good initial guess [31].

4.8.2 Mesh Error Tolerance

The base case in Section 4.5 is solved with a default mesh error tolerance of 1×10^{-4}. A solution was unobtainable with more restrictive tolerance of 1×10^{-5}. The comparison between the control profiles for a tolerance of 1×10^{-3} is compared against the baseline in Figure 4.8.1. The resulting trajectories for each mesh tolerances, are plotted in Figure 4.8.2.

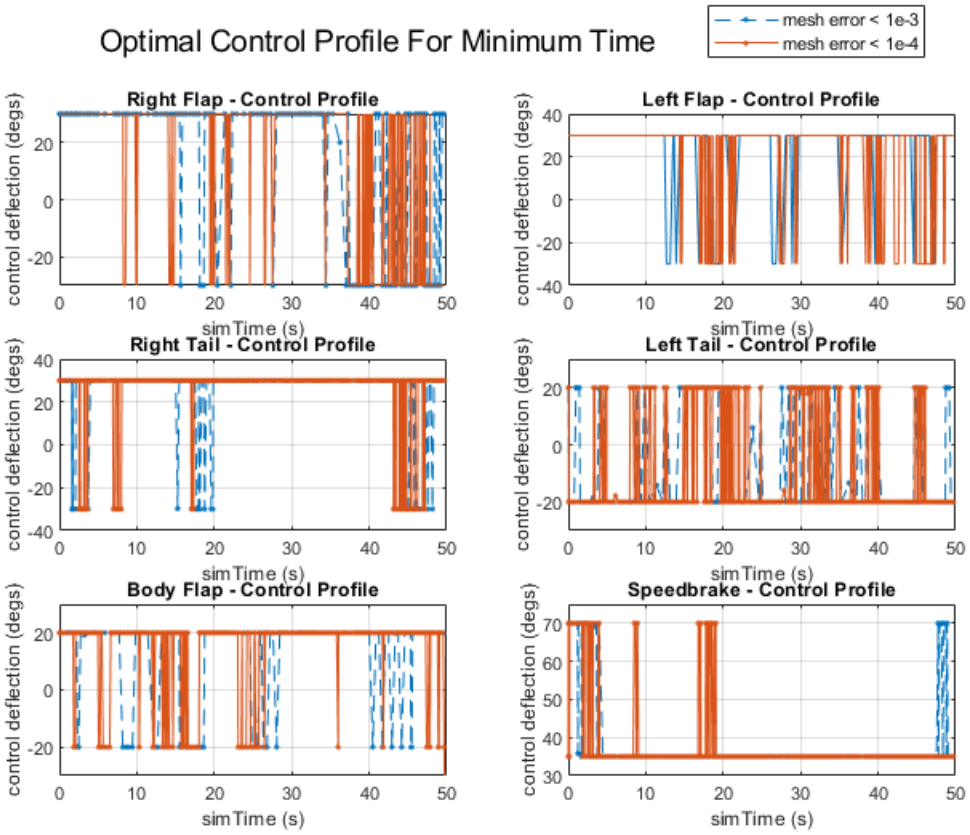

Figure 4.8.1: Mesh Tolerance Analysis: Control Profile

The differences between the state history for these trajectories and computed control profiles are minor, as plotted in Figures 4.8.1 and 4.8.2.

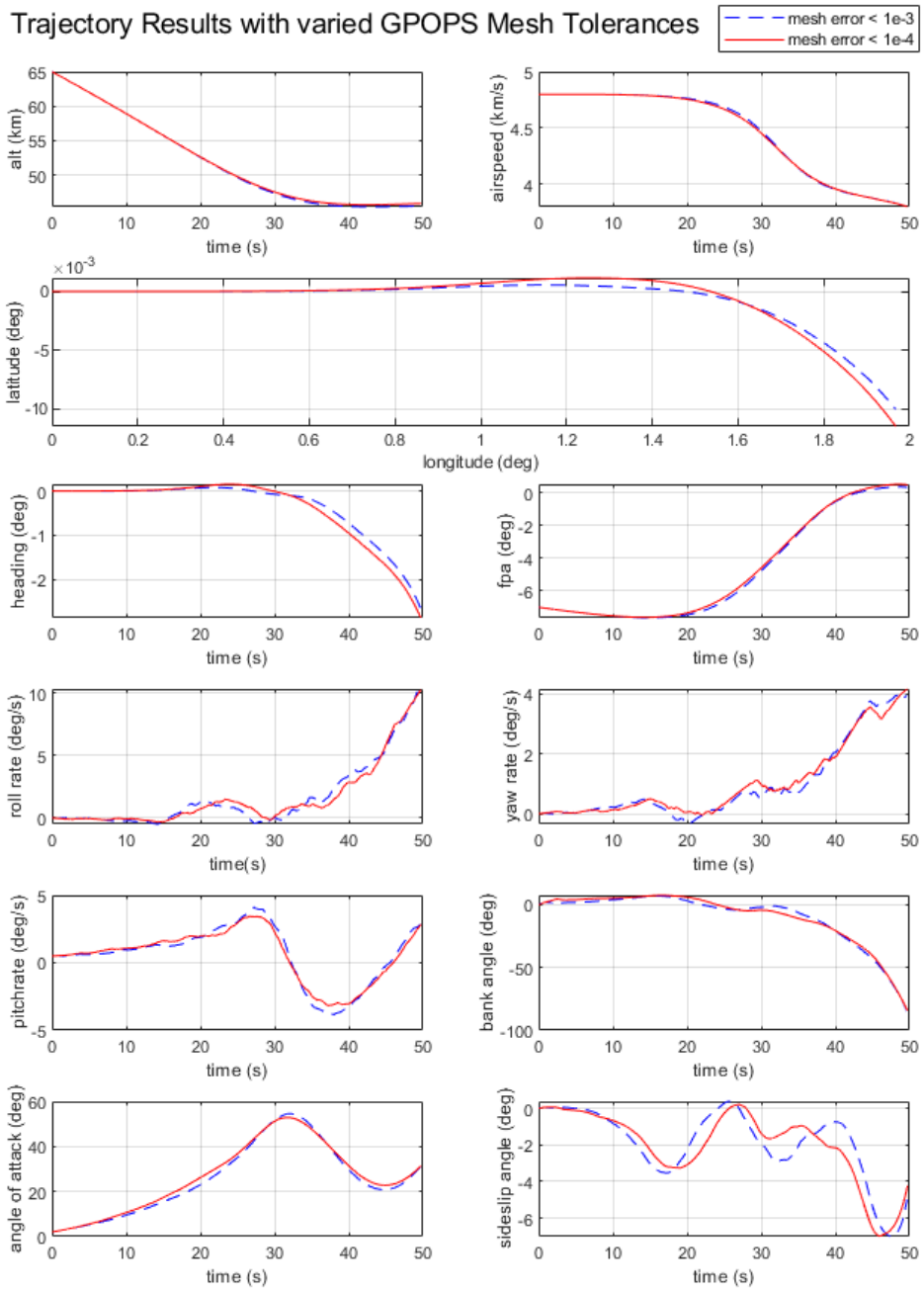

Figure 4.8.2: Mesh Tolerance Analysis: Trajectory

V. Conclusions and Recommendations

The results above validates the equations of motion derived in Section 3.3. The results of the 6-DoF simulation for all three control inputs are realistic and reasonable. Given a high-fidelity aerodynamic database, requisite vehicle properties and a control input the simulation could be adapted to emulate the reentry profile of a given RV. The control deflections of the vehicle is integrated into the reentry equations of the motion so that their direct effect on the resulting trajectory is easily observed.

A novel method of using neural networks to approximate aerodynamic data proved to be successful. Neural networks allowed to approximate multi-dimensional lookup tables for aerodynamic coefficients as continuous analytical functions in terms of independent variables. The use of ANN approximation has alleviated expensive multi-dimensional table-lookups and the problems caused by discontinuos derivatives computed from discrete data.

The investigation to seek optimal control policy for a given scenario was less of a success. Although a solution was found within the defined mesh tolerance, there is no assurance that GPOPS has found the global minimum. Additionally, the flight time was not significantly reduced in either Section 4.5 or 4.6. It is found that a strong initial guess is required for a proper convergence to a solution. So far, the only way to generate an initial guess is by trial and error. Alternative ways to generate a strong initial guess is outside the scope of this study and is discussed in Section 5.2. As demonstrated in Section 4.6, with a strong initial guess, GPOPS could be employed to make corrections to control profile to achieve the desired end state.

5.1 Limitations and Caveats

Among many things that affect the effectiveness of the research, the aerodynamic database is the most prominent. Even with the current state-of-the-art technology, the error on predicted aerodynamic coefficients is sizeable [5]. This simulation assumes perfect knowledge of aerodynamic data, which is impractical. However, with improvements in our understanding of complex aerodynamic effects, more accurate prediction of aerodynamic data is imminent. The goal of this research is to demonstrate that a high-fidelity full 6-DoF simulation could be done with appropriate aerodynamic data.

The optimal control solution fails to converge for long flight-times, such as: reentry to impact/landing. It is also difficult to generate a guess control profile for such a trajectory. It is common to divide the problem into different phases to establish waypoints in such a case [4].

The current formulation is also subject to the singularities listed below:

- At the geographic poles: When the latitude is approaches 90 degrees, i.e when $\lambda \ \pm 90^0$, the denominator in Equation 3.3.3 approaches zero. This will cause issues for intercontinental missions which may require to fly over the poles. The remedy to this is to establish a reference frame based on the mission's launch and target locations and re-derive the equations of motion with respect to that frame [12].

- $\gamma \ \pm 90^0$: Such a case will cause an singularity in the Force Equations of Motion (Equation 3.3.10).Fortunately, the cases where flight path angle is close to ninety degrees is virtually non-existent.

- $\beta \ \pm 90^0$: As discussed in Section 3.3.5.1, Equations 3.3.29 fail when sideslip angle approaches 90 degrees. Again, the magnitude of β does not exceed 45^0 for the overwhelming majority of reentry problems.

5.2 Future Work

Recommendations for follow-on research to this project are listed as follows:

- GPOPS-II could be used to investigate optimal trajectories subject to mission-specific cost function. Betts [32] has explored computing a control profile that optimized the cross range of a 3-DoF trajectory. The system reenters at "null-island" with the initial velocity pointing easterwards along the equator. The cost function was formulated as:

$$J_3 = -\lambda(t_f) \tag{5.2.1}$$

- More realistic planetary model: Switching from a geocentric to geodetic planetary model with modeling of wind effects. The gravity model could also be updated to inculcate spherical harmonic effects.

- Design of an autopilot to predict control behavior for a desired trajectory. As mentioned earlier, the current way to predict a control profile for a desired trajectory is by trial and error. An autopilot algorithm can be designed to deterministic ally solve for a non-optimal control history. Such a feat is very complex and is its own thesis topic. A few studies exist that investigates building autopilot for less complex points. Ito et al [33] have used the concept of Dynamic Inversion techniques to build such a system. For a highly complex suite of equations of motion such as this case, more novel methods may be of help. Lee [11] has explored the possibility using reinforcement learning algorithms to compute the minimum time control profile for a 3-DoF reentry system. That problem, which is far less complex than he focus of this study, took days of computer time to arrive at a solution. Therefore, doing so in a truly 6-DoF simulation environment could only be accomplished in a high-performance

computing environment.

- Dynamic Programming: This methodology reduces large optimization problems in to a number of smaller discrete sub-problems. The subproblems are then solved based on Bellman's Principle of Optimality which states the optimal policy from to the final state is independent of any states or control decision that preceded an intermediate state. Grant et al. [3] has used a disrete dynamic programming to produce accurate initial guesses for a 3-DoF reentry problem. It was reported that this approach greatly reduced the number of required computations. Applying their methods to a 6-DoF reentry scenario is a plausible avenue of future research.

Appendix A. Transformation Matrix From BRF to FPF

$\boldsymbol{T_B^F}$ is a direction cosine matrix that defines the transformation from Body Reference Frame (BRF) to Fligh-Path Reference Frame(FPF). From Equation 3.2.17, $\boldsymbol{T_B^F}$ is expanded as:

$$
\begin{aligned}
T_B^F &= T_A^F T_B^A = \left(T_F^A\right)^T T_B^A \\
&= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \\
a_{11} &= \cos(\alpha)\cos(\beta) \\
a_{12} &= \sin(\beta) \\
a_{13} &= \cos(\beta)\sin(\alpha) \\
a_{21} &= \cos(\sigma)(\sin(\alpha)\sin(\eta) - \cos(\alpha)\cos(\eta)\sin(\beta)) \\
&\quad - \sin(\sigma)(\cos(\eta)\sin(\alpha) + \cos(\alpha)\sin(\beta)\sin(\eta)) \\
a_{22} &= \cos(\sigma)\cos(\beta)\cos(\eta) + \cos(\beta)\sin(\sigma)\sin(\eta) \\
a_{23} &= \sin(\sigma)(\cos(\alpha)\cos(\eta) - \sin(\alpha)\sin(\beta)\sin(\eta)) \\
&\quad - \cos(\sigma)(\cos(\alpha)\sin(\eta) + \cos(\eta)\sin(\alpha)\sin(\beta)) \\
a_{31} &= -\cos(\sigma)(\cos(\eta)\sin(\alpha) + \cos(\alpha)\sin(\beta)\sin(\eta)) \\
&\quad - \sin(\sigma)(\sin(\alpha)\sin(\eta) - \cos(\alpha)\cos(\eta)\sin(\beta)) \\
a_{32} &= \cos(\sigma)\cos(\beta)\sin(\eta) - \cos(\beta)\cos(\eta)\sin(\sigma) \\
a_{33} &= \cos(\sigma)(\cos(\alpha)\cos(\eta) - \sin(\alpha)\sin(\beta)\sin(\eta)) \\
&\quad + \sin(\sigma)(\cos(\alpha)sin(\eta) + \cos(\eta)\sin(\alpha)sin(\beta))
\end{aligned}
\tag{1.0.1}
$$

Appendix B. MATLAB Scripts

Most of the MATLAB scripts are omitted for the purposes of brevity. However the function that embodies equations in Figure 3.4.1 is listed below.

```
function phaseout = reentryDynamics(input)
earthRate_rad_s = 7.29115e-5;
earthRadius_m   = 6371203.92;
refDensity_kg_m3 = 1.225;
gravConstant_m3_s2 = 3.9860064e14;

vehicle = input.auxdata.vehicleProperties;
vehicleMass_kg = vehicle.vehicleMass_kg;
%% Define Inputs and Controls, Get Time
radius_m   = input.phase.state(:,1);
%longitude_rad     = input.phase.state(:,2);
latitude_rad = input.phase.state(:,3);
speed_m_s    = input.phase.state(:,4);
psi_rad    = input.phase.state(:,5);
fpa_rad = input.phase.state(:,6);
rollRate_rad_s = input.phase.state(:,7);
pitchRate_rad_s = input.phase.state(:,8);
yawRate_rad_s = input.phase.state(:,9);
bank_rad = input.phase.state(:,10);
alpha_rad = input.phase.state(:,11);
beta_rad = input.phase.state(:,12);

d1_rad    = input.phase.control(:,1);
```

```
d2_rad    = input.phase.control(:,2);
d3_rad    = input.phase.control(:,3);
d4_rad    = input.phase.control(:,4);
d5_rad    = input.phase.control(:,5);
d6_rad   =  input.phase.control(:,6);

%simTime_s  = input.phase.time;
%% Kinematic Equations
rDot_m_s = speed_m_s.*sin(fpa_rad);
lonRate_rad_s = speed_m_s.*cos(fpa_rad).*cos(psi_rad)./
   radius_m./cos(latitude_rad);
latRate_rad_s = speed_m_s.*cos(fpa_rad).*sin(psi_rad)./
   radius_m;
%% atmospheric Calculations
alt_m = radius_m - earthRadius_m;
% density estimation
legacyRho_kg_m3 = refDensity_kg_m3*exp(-alt_m/7250);
deltaRho1_kg_m3 = 0.05*exp(-((alt_m-4200)/3000).^2) ...
    + 0.11*exp(-((alt_m-8700)/6400).^2);
deltaRho2_kg_m3 = 0.007*exp(-((alt_m-19000)/2800).^2) ...
    + 0.002*exp(-((alt_m-23000)/1800).^2);
deltaRho3_kg_m3 = -0.0014*exp(-((alt_m-37200)/11400).^2)
   ...
    + 0.00027*exp(-((alt_m-48700)/10300).^2);
deltaRho4_kg_m3 = -1.2e-4*exp(-((alt_m-53800)/6400).^2)
   ...
```

```
    + -1.4e-05*exp(-((alt_m-60200)/6400).^2);
%DensityModel
density_kg_m3 = legacyRho_kg_m3 + deltaRho1_kg_m3 +
   deltaRho2_kg_m3 ...
                                  + deltaRho3_kg_m3 +
                                     deltaRho4_kg_m3;
dynamicPressure_Pa = 0.5*density_kg_m3.*speed_m_s.*
   speed_m_s;
dynamicForce_N = (vehicle.aeroRefArea_m2)*
   dynamicPressure_Pa;%q*RefArea

speedOfSound_m_s = 3.791e-22*alt_m.^5 + -5.793e-17*alt_m
   .^4 +  1.111e-12*alt_m.^3 ...
        +1.435e-07*alt_m.^2 - 0.00536*alt_m + 341.9;
mach_nd = speed_m_s./speedOfSound_m_s;
%% Force and Moment Coefficients

[Cx_nd] = computeCx_nd(mach_nd,alpha_rad,...
                    d1_rad,d2_rad,d3_rad,d4_rad,d5_rad,
                       d6_rad,...
                    speed_m_s,pitchRate_rad_s,vehicle);
[Cy_nd] = computeCy_nd(mach_nd,alpha_rad,beta_rad,...
                                  d1_rad,d2_rad,d3_rad,
                                     d4_rad,...
                                  speed_m_s,rollRate_rad_s,
                                     yawRate_rad_s,vehicle);
```

```
[Cz_nd] = computeCz_nd(mach_nd,alpha_rad,...
                            d1_rad,d2_rad,d3_rad,d4_rad,
                               d5_rad,d6_rad,...
                            speed_m_s,pitchRate_rad_s,
                               vehicle);
[Cl_nd] = computeCl_nd(mach_nd,alpha_rad,beta_rad,...
                            d1_rad,d2_rad,d3_rad,d4_rad,
                               d5_rad,...
                            speed_m_s,rollRate_rad_s,
                               yawRate_rad_s,vehicle);
[Cm_nd] = computeCm_nd(mach_nd,alpha_rad,beta_rad,...
                            d1_rad,d2_rad,d3_rad,d4_rad,
                               d5_rad,d6_rad,...
                            speed_m_s,pitchRate_rad_s,
                               vehicle);
[Cn_nd] = computeCn_nd(mach_nd,alpha_rad,beta_rad,...
                            d1_rad,d2_rad,d3_rad,d4_rad
                               ,...
                            speed_m_s,rollRate_rad_s,
                               yawRate_rad_s,vehicle);
%% Attitude Rates
torqueL_Nm = (vehicle.wingspan_m)*dynamicForce_N.*Cl_nd;
torqueM_Nm = (vehicle.meanChord_m)*dynamicForce_N.*Cm_nd;
torqueN_Nm = (vehicle.wingspan_m)*dynamicForce_N.*Cn_nd;

Ixx_kg_m2 = vehicle.inertiaTensor_kgm2(1,1);
```

```
Iyy_kg_m2 = vehicle.inertiaTensor_kgm2(2,2);
Izz_kg_m2 = vehicle.inertiaTensor_kgm2(3,3);
Izx_kg_m2 = vehicle.inertiaTensor_kgm2(1,3);

pDot_rad_s2 = Izz_kg_m2*(torqueL_Nm+Izx_kg_m2*torqueN_Nm/
   Izz_kg_m2...
              + yawRate_rad_s.*pitchRate_rad_s*(Iyy_kg_m2-
                 Izz_kg_m2-(Izx_kg_m2^2)/Izz_kg_m2)...
              + rollRate_rad_s.*pitchRate_rad_s*Izx_kg_m2
                 *...
              (1+(Ixx_kg_m2-Iyy_kg_m2)/Izz_kg_m2))/(
                 Izz_kg_m2*Ixx_kg_m2 - Izx_kg_m2^2);
qDot_rad_s2 = (torqueM_Nm + Izx_kg_m2*(yawRate_rad_s.^2 -
   rollRate_rad_s.^2) + ...
               (Izz_kg_m2-Ixx_kg_m2)*yawRate_rad_s.*
                  rollRate_rad_s)/Iyy_kg_m2;
rDot_rad_s2 = (torqueN_Nm + Izx_kg_m2*(pDot_rad_s2-
   pitchRate_rad_s.*yawRate_rad_s) + ...
               (Ixx_kg_m2-Iyy_kg_m2)*pitchRate_rad_s.*
                  rollRate_rad_s)/Izz_kg_m2;
%% Force Equations
doublePrimeCy_nd =  Cy_nd.*cos(beta_rad) - Cx_nd.*cos(
   alpha_rad).*sin(beta_rad) ...
    - Cz_nd.*sin(alpha_rad).*sin(beta_rad);
doublePrimeCz_nd = Cz_nd.*cos(alpha_rad) - Cx_nd.*sin(
   alpha_rad);
```

```
eta_rad = (doublePrimeCy_nd./hypot(doublePrimeCy_nd,1e-12)
   )...
     .*acos(doublePrimeCz_nd./hypot(doublePrimeCz_nd,
        doublePrimeCy_nd));

speedRate_m_s = (dynamicForce_N.*(Cy_nd.*sin(beta_rad) ...
                + Cx_nd.*cos(alpha_rad).*cos(beta_rad) ...
                + Cz_nd.*cos(beta_rad).*sin(alpha_rad)))./
                   vehicleMass_kg ...
                - (gravConstant_m3_s2.*sin(fpa_rad))./
                   radius_m.^2 ...
                + earthRate_rad_s.^2.*radius_m.*cos(
                   latitude_rad).*(cos(latitude_rad).*sin(
                   fpa_rad) ...
                - cos(fpa_rad).*sin(latitude_rad).*sin(
                   psi_rad));

psiRate_rad_s  = -(2.*earthRate_rad_s.*speed_m_s.*(cos(
   fpa_rad).*sin(latitude_rad) ...
                - cos(latitude_rad).*sin(fpa_rad).*sin(
                   psi_rad)) ...
                - (Cx_nd.*dynamicForce_N.*(cos(bank_rad)
                   .*(sin(alpha_rad).*sin(eta_rad) ...
                - cos(alpha_rad).*cos(eta_rad).*sin(
                   beta_rad)) ...
```

```
                    - sin(bank_rad).*(cos(eta_rad).*sin(
                       alpha_rad) ...
                    + cos(alpha_rad).*sin(beta_rad).*sin(
                       eta_rad))) ...
                    - Cz_nd.*dynamicForce_N.*(cos(bank_rad).*(
                       cos(alpha_rad).*sin(eta_rad) ...
                    + cos(eta_rad).*sin(alpha_rad).*sin(
                       beta_rad)) ...
                    - sin(bank_rad).*(cos(alpha_rad).*cos(
                       eta_rad) ...
                    - sin(alpha_rad).*sin(beta_rad).*sin(
                       eta_rad))) ...
                    + Cy_nd.*dynamicForce_N.*(cos(bank_rad).*
                       cos(beta_rad).*cos(eta_rad) ...
                    + cos(beta_rad).*sin(bank_rad).*sin(
                       eta_rad))).*vehicleMass_kg ...
                    + earthRate_rad_s.^2.*radius_m.*cos(
                       latitude_rad).*cos(psi_rad).*sin(
                       latitude_rad) ...
                    + (speed_m_s.^2.*cos(fpa_rad).^2.*cos(
                       psi_rad).*tan(latitude_rad))./radius_m)
                       ...
                    ./(speed_m_s.*cos(fpa_rad));

fpaRate_rad_s  = ((speed_m_s.^2.*cos(fpa_rad).^2)./
   radius_m ...
```

```
- (Cx_nd.*dynamicForce_N.*(cos(bank_rad)
   .*(cos(eta_rad).*sin(alpha_rad) ...
+ cos(alpha_rad).*sin(beta_rad).*sin(
   eta_rad)) ...
+ sin(bank_rad).*(sin(alpha_rad).*sin(
   eta_rad) ...
- cos(alpha_rad).*cos(eta_rad).*sin(
   beta_rad))) ...
- Cz_nd.*dynamicForce_N.*(cos(bank_rad).*(
   cos(alpha_rad).*cos(eta_rad) ...
- sin(alpha_rad).*sin(beta_rad).*sin(
   eta_rad)) ...
+ sin(bank_rad).*(cos(alpha_rad).*sin(
   eta_rad) ...
+ cos(eta_rad).*sin(alpha_rad).*sin(
   beta_rad))) ...
+ Cy_nd.*dynamicForce_N.*cos(beta_rad).*
   sin(bank_rad - eta_rad))./
   vehicleMass_kg ...
- (gravConstant_m3_s2.*cos(fpa_rad))./
   radius_m.^2 ...
+ earthRate_rad_s.^2.*radius_m.*cos(
   latitude_rad).*(cos(fpa_rad).*cos(
   latitude_rad) ...
+ sin(fpa_rad).*sin(latitude_rad).*sin(
   psi_rad)) ...
```

```
                + 2.*earthRate_rad_s.*speed_m_s.*cos(
                   latitude_rad).*cos(psi_rad))...
                ./speed_m_s;
%% Aerodyn. Angle Rate Equations
bankRate_rad_s = ((Cx_nd.^2.*sin(alpha_rad).^3 + Cz_nd
   .^2.*(sin(alpha_rad) ...
                  - sin(alpha_rad).^3) - 2.*Cx_nd.*Cz_nd
                     .*(cos(alpha_rad) ...
                  - cos(alpha_rad).^3) - Cx_nd.^2.*cos(
                     eta_rad).^2.*sin(alpha_rad).*sin(
                     beta_rad).^2 ...
                  - Cz_nd.^2.*cos(eta_rad).^2.*sin(
                     alpha_rad).*sin(beta_rad).^2 ...
                  + Cx_nd.^2.*cos(alpha_rad).^2.*cos(
                     beta_rad).^2.*cos(eta_rad).^2.*sin(
                     alpha_rad) ...
                  - Cz_nd.^2.*cos(alpha_rad).^2.*cos(
                     beta_rad).^2.*cos(eta_rad).^2.*sin(
                     alpha_rad) ...
                  - Cx_nd.*Cz_nd.*cos(alpha_rad).^3.*cos(
                     beta_rad).^2.*cos(eta_rad).^2 ...
                  + Cx_nd.*Cz_nd.*cos(alpha_rad).*cos(
                     beta_rad).^2.*cos(eta_rad).^2.*sin(
                     alpha_rad).^2 ...
                  - Cy_nd.*Cz_nd.*cos(alpha_rad).^2.*cos(
                     beta_rad).*cos(eta_rad).^2.*sin(
```

```
    beta_rad) ...
  + Cy_nd.*Cz_nd.*cos(beta_rad).*cos(
    eta_rad).^2.*sin(alpha_rad).^2.*sin(
    beta_rad) ...
  + 2.*Cx_nd.*Cy_nd.*cos(alpha_rad).*cos(
    beta_rad).*cos(eta_rad).^2.*sin(
    alpha_rad).*sin(beta_rad)).*(
    fpaRate_rad_s.*cos(alpha_rad).*cos(
    bank_rad).*sin(eta_rad) ...
  - yawRate_rad_s - fpaRate_rad_s.*cos(
    alpha_rad).*cos(eta_rad).*sin(
    bank_rad) ...
  + psiRate_rad_s.*cos(beta_rad).*sin(
    alpha_rad).*sin(fpa_rad) ...
  - latRate_rad_s.*cos(beta_rad).*cos(
    fpa_rad).*cos(psi_rad).*sin(alpha_rad
    ) ...
  + psiRate_rad_s.*cos(alpha_rad).*cos(
    fpa_rad).*sin(bank_rad).*sin(eta_rad)
      ...
  - latRate_rad_s.*cos(alpha_rad).*cos(
    bank_rad).*sin(eta_rad).*sin(psi_rad)
      ...
  + latRate_rad_s.*cos(alpha_rad).*cos(
    eta_rad).*sin(bank_rad).*sin(psi_rad)
      ...
```

```
+ earthRate_rad_s.*cos(beta_rad).*sin(
   alpha_rad).*sin(fpa_rad).*sin(
   latitude_rad) ...
+ fpaRate_rad_s.*sin(alpha_rad).*sin(
   bank_rad).*sin(beta_rad).*sin(eta_rad
   ) ...
+ lonRate_rad_s.*cos(beta_rad).*sin(
   alpha_rad).*sin(fpa_rad).*sin(
   latitude_rad) ...
+ psiRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(eta_rad).*cos(fpa_rad)
    ...
+ fpaRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*sin(alpha_rad).*sin(
   beta_rad) ...
- latRate_rad_s.*sin(alpha_rad).*sin(
   bank_rad).*sin(beta_rad).*sin(eta_rad
   ).*sin(psi_rad) ...
+ earthRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(eta_rad).*cos(fpa_rad)
   .*sin(latitude_rad) ...
+ lonRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(eta_rad).*cos(fpa_rad)
   .*sin(latitude_rad) ...
- earthRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(latitude_rad).*cos(
```

```
    psi_rad).*sin(eta_rad) ...
+ earthRate_rad_s.*cos(alpha_rad).*cos(
    eta_rad).*cos(latitude_rad).*cos(
    psi_rad).*sin(bank_rad) ...
+ latRate_rad_s.*cos(alpha_rad).*cos(
    bank_rad).*cos(eta_rad).*cos(psi_rad)
    .*sin(fpa_rad) ...
- lonRate_rad_s.*cos(alpha_rad).*cos(
    bank_rad).*cos(latitude_rad).*cos(
    psi_rad).*sin(eta_rad) ...
+ lonRate_rad_s.*cos(alpha_rad).*cos(
    eta_rad).*cos(latitude_rad).*cos(
    psi_rad).*sin(bank_rad) ...
+ earthRate_rad_s.*cos(beta_rad).*cos(
    fpa_rad).*cos(latitude_rad).*sin(
    alpha_rad).*sin(psi_rad) ...
+ earthRate_rad_s.*cos(alpha_rad).*cos(
    fpa_rad).*sin(bank_rad).*sin(eta_rad)
    .*sin(latitude_rad) ...
- psiRate_rad_s.*cos(bank_rad).*cos(
    fpa_rad).*sin(alpha_rad).*sin(
    beta_rad).*sin(eta_rad) ...
+ psiRate_rad_s.*cos(eta_rad).*cos(
    fpa_rad).*sin(alpha_rad).*sin(
    bank_rad).*sin(beta_rad) ...
```

```
+ lonRate_rad_s.*cos(beta_rad).*cos(
   fpa_rad).*cos(latitude_rad).*sin(
   alpha_rad).*sin(psi_rad) ...
- latRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*sin(alpha_rad).*sin(
   beta_rad).*sin(psi_rad) ...
+ lonRate_rad_s.*cos(alpha_rad).*cos(
   fpa_rad).*sin(bank_rad).*sin(eta_rad)
   .*sin(latitude_rad) ...
+ latRate_rad_s.*cos(alpha_rad).*cos(
   psi_rad).*sin(bank_rad).*sin(eta_rad)
   .*sin(fpa_rad) ...
- earthRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*cos(latitude_rad).*cos(
   psi_rad).*sin(alpha_rad).*sin(
   beta_rad) ...
- earthRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(eta_rad).*cos(
   latitude_rad).*sin(fpa_rad).*sin(
   psi_rad) ...
- lonRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*cos(latitude_rad).*cos(
   psi_rad).*sin(alpha_rad).*sin(
   beta_rad) ...
- lonRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(eta_rad).*cos(
```

```
    latitude_rad).*sin(fpa_rad).*sin(
    psi_rad) ...
-   earthRate_rad_s.*cos(bank_rad).*cos(
    fpa_rad).*sin(alpha_rad).*sin(
    beta_rad).*sin(eta_rad).*sin(
    latitude_rad) ...
+   earthRate_rad_s.*cos(eta_rad).*cos(
    fpa_rad).*sin(alpha_rad).*sin(
    bank_rad).*sin(beta_rad).*sin(
    latitude_rad) ...
-   lonRate_rad_s.*cos(bank_rad).*cos(
    fpa_rad).*sin(alpha_rad).*sin(
    beta_rad).*sin(eta_rad).*sin(
    latitude_rad) ...
+   lonRate_rad_s.*cos(eta_rad).*cos(
    fpa_rad).*sin(alpha_rad).*sin(
    bank_rad).*sin(beta_rad).*sin(
    latitude_rad) ...
-   earthRate_rad_s.*cos(latitude_rad).*
    cos(psi_rad).*sin(alpha_rad).*sin(
    bank_rad).*sin(beta_rad).*sin(eta_rad
    ) ...
-   latRate_rad_s.*cos(bank_rad).*cos(
    psi_rad).*sin(alpha_rad).*sin(
    beta_rad).*sin(eta_rad).*sin(fpa_rad)
     ...
```

```
+ latRate_rad_s.*cos(eta_rad).*cos(
   psi_rad).*sin(alpha_rad).*sin(
   bank_rad).*sin(beta_rad).*sin(fpa_rad
   ) ...
- earthRate_rad_s.*cos(alpha_rad).*cos(
   latitude_rad).*sin(bank_rad).*sin(
   eta_rad).*sin(fpa_rad).*sin(psi_rad)
   ...
- lonRate_rad_s.*cos(latitude_rad).*cos(
   psi_rad).*sin(alpha_rad).*sin(
   bank_rad).*sin(beta_rad).*sin(eta_rad
   ) ...
- lonRate_rad_s.*cos(alpha_rad).*cos(
   latitude_rad).*sin(bank_rad).*sin(
   eta_rad).*sin(fpa_rad).*sin(psi_rad)
   ...
+ earthRate_rad_s.*cos(bank_rad).*cos(
   latitude_rad).*sin(alpha_rad).*sin(
   beta_rad).*sin(eta_rad).*sin(fpa_rad)
   .*sin(psi_rad) ...
- earthRate_rad_s.*cos(eta_rad).*cos(
   latitude_rad).*sin(alpha_rad).*sin(
   bank_rad).*sin(beta_rad).*sin(fpa_rad
   ).*sin(psi_rad) ...
+ lonRate_rad_s.*cos(bank_rad).*cos(
   latitude_rad).*sin(alpha_rad).*sin(
```

```
    beta_rad).*sin(eta_rad).*sin(fpa_rad)
    .*sin(psi_rad) ...
- lonRate_rad_s.*cos(eta_rad).*cos(
    latitude_rad).*sin(alpha_rad).*sin(
    bank_rad).*sin(beta_rad).*sin(fpa_rad
    ).*sin(psi_rad))) ...
./(cos(beta_rad).*(Cz_nd.*cos(alpha_rad)
     - Cx_nd.*sin(alpha_rad)).^2) ...
- ((Cz_nd.^2.*cos(alpha_rad).^3 ...
+ Cx_nd.^2.*(cos(alpha_rad) - cos(
    alpha_rad).^3) ...
- 2.*Cx_nd.*Cz_nd.*(sin(alpha_rad) - sin
    (alpha_rad).^3) ...
- Cx_nd.^2.*cos(alpha_rad).*cos(eta_rad)
    .^2.*sin(beta_rad).^2 ...
- Cz_nd.^2.*cos(alpha_rad).*cos(eta_rad)
    .^2.*sin(beta_rad).^2 ...
- Cx_nd.^2.*cos(alpha_rad).*cos(beta_rad
    ).^2.*cos(eta_rad).^2.*sin(alpha_rad)
    .^2 ...
+ Cz_nd.^2.*cos(alpha_rad).*cos(beta_rad
    ).^2.*cos(eta_rad).^2.*sin(alpha_rad)
    .^2 ...
- Cx_nd.*Cz_nd.*cos(beta_rad).^2.*cos(
    eta_rad).^2.*sin(alpha_rad).^3 ...
```

```
+ Cx_nd.*Cz_nd.*cos(alpha_rad).^2.*cos(
   beta_rad).^2.*cos(eta_rad).^2.*sin(
   alpha_rad) ...
+ Cx_nd.*Cy_nd.*cos(alpha_rad).^2.*cos(
   beta_rad).*cos(eta_rad).^2.*sin(
   beta_rad) ...
- Cx_nd.*Cy_nd.*cos(beta_rad).*cos(
   eta_rad).^2.*sin(alpha_rad).^2.*sin(
   beta_rad) ...
+ 2.*Cy_nd.*Cz_nd.*cos(alpha_rad).*cos(
   beta_rad).*cos(eta_rad).^2.*sin(
   alpha_rad).*sin(beta_rad))...
.*(rollRate_rad_s - psiRate_rad_s.*cos(
   alpha_rad).*cos(beta_rad).*sin(
   fpa_rad) ...
+ fpaRate_rad_s.*cos(bank_rad).*sin(
   alpha_rad).*sin(eta_rad) ...
- fpaRate_rad_s.*cos(eta_rad).*sin(
   alpha_rad).*sin(bank_rad) ...
- earthRate_rad_s.*cos(alpha_rad).*cos(
   beta_rad).*sin(fpa_rad).*sin(
   latitude_rad) ...
- fpaRate_rad_s.*cos(alpha_rad).*sin(
   bank_rad).*sin(beta_rad).*sin(eta_rad
   ) ...
```

```
- lonRate_rad_s.*cos(alpha_rad).*cos(
   beta_rad).*sin(fpa_rad).*sin(
   latitude_rad) ...
+ psiRate_rad_s.*cos(fpa_rad).*sin(
   alpha_rad).*sin(bank_rad).*sin(
   eta_rad) ...
- latRate_rad_s.*cos(bank_rad).*sin(
   alpha_rad).*sin(eta_rad).*sin(psi_rad
   ) ...
+ latRate_rad_s.*cos(eta_rad).*sin(
   alpha_rad).*sin(bank_rad).*sin(
   psi_rad) ...
- fpaRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(eta_rad).*sin(beta_rad
   ) ...
+ latRate_rad_s.*cos(alpha_rad).*cos(
   beta_rad).*cos(fpa_rad).*cos(psi_rad)
    ...
+ psiRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*cos(fpa_rad).*sin(alpha_rad
   ) ...
- earthRate_rad_s.*cos(alpha_rad).*cos(
   beta_rad).*cos(fpa_rad).*cos(
   latitude_rad).*sin(psi_rad) ...
+ earthRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*cos(fpa_rad).*sin(alpha_rad
```

```
   ).*sin(latitude_rad) ...
+ psiRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(fpa_rad).*sin(beta_rad
   ).*sin(eta_rad) ...
- psiRate_rad_s.*cos(alpha_rad).*cos(
   eta_rad).*cos(fpa_rad).*sin(bank_rad)
   .*sin(beta_rad) ...
- lonRate_rad_s.*cos(alpha_rad).*cos(
   beta_rad).*cos(fpa_rad).*cos(
   latitude_rad).*sin(psi_rad) ...
+ latRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(eta_rad).*sin(beta_rad
   ).*sin(psi_rad) ...
+ lonRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*cos(fpa_rad).*sin(alpha_rad
   ).*sin(latitude_rad) ...
- earthRate_rad_s.*cos(bank_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
   alpha_rad).*sin(eta_rad) ...
+ earthRate_rad_s.*cos(eta_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
   alpha_rad).*sin(bank_rad) ...
+ latRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*cos(psi_rad).*sin(alpha_rad
   ).*sin(fpa_rad) ...
```

```
- lonRate_rad_s.*cos(bank_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
   alpha_rad).*sin(eta_rad) ...
+ lonRate_rad_s.*cos(eta_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
   alpha_rad).*sin(bank_rad) ...
+ earthRate_rad_s.*cos(fpa_rad).*sin(
   alpha_rad).*sin(bank_rad).*sin(
   eta_rad).*sin(latitude_rad) ...
+ latRate_rad_s.*cos(alpha_rad).*sin(
   bank_rad).*sin(beta_rad).*sin(eta_rad
   ).*sin(psi_rad) ...
+ lonRate_rad_s.*cos(fpa_rad).*sin(
   alpha_rad).*sin(bank_rad).*sin(
   eta_rad).*sin(latitude_rad) ...
+ latRate_rad_s.*cos(psi_rad).*sin(
   alpha_rad).*sin(bank_rad).*sin(
   eta_rad).*sin(fpa_rad) ...
+ earthRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(eta_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
   beta_rad) ...
+ lonRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(eta_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
   beta_rad) ...
```

```
+ earthRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(fpa_rad).*sin(beta_rad
   ).*sin(eta_rad).*sin(latitude_rad)
   ...
- earthRate_rad_s.*cos(alpha_rad).*cos(
   eta_rad).*cos(fpa_rad).*sin(bank_rad)
   .*sin(beta_rad).*sin(latitude_rad)
   ...
+ lonRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(fpa_rad).*sin(beta_rad
   ).*sin(eta_rad).*sin(latitude_rad)
   ...
- lonRate_rad_s.*cos(alpha_rad).*cos(
   eta_rad).*cos(fpa_rad).*sin(bank_rad)
   .*sin(beta_rad).*sin(latitude_rad)
   ...
+ earthRate_rad_s.*cos(alpha_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
   bank_rad).*sin(beta_rad).*sin(eta_rad
   ) ...
+ latRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(psi_rad).*sin(beta_rad
   ).*sin(eta_rad).*sin(fpa_rad) ...
- latRate_rad_s.*cos(alpha_rad).*cos(
   eta_rad).*cos(psi_rad).*sin(bank_rad)
   .*sin(beta_rad).*sin(fpa_rad) ...
```

```
- earthRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*cos(latitude_rad).*sin(
   alpha_rad).*sin(fpa_rad).*sin(psi_rad
   ) ...
+ lonRate_rad_s.*cos(alpha_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
   bank_rad).*sin(beta_rad).*sin(eta_rad
   ) ...
- lonRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*cos(latitude_rad).*sin(
   alpha_rad).*sin(fpa_rad).*sin(psi_rad
   ) ...
- earthRate_rad_s.*cos(latitude_rad).*
   sin(alpha_rad).*sin(bank_rad).*sin(
   eta_rad).*sin(fpa_rad).*sin(psi_rad)
   ...
- lonRate_rad_s.*cos(latitude_rad).*sin(
   alpha_rad).*sin(bank_rad).*sin(
   eta_rad).*sin(fpa_rad).*sin(psi_rad)
   ...
- earthRate_rad_s.*cos(alpha_rad).*cos(
   bank_rad).*cos(latitude_rad).*sin(
   beta_rad).*sin(eta_rad).*sin(fpa_rad)
   .*sin(psi_rad) ...
+ earthRate_rad_s.*cos(alpha_rad).*cos(
   eta_rad).*cos(latitude_rad).*sin(
```

```
    bank_rad).*sin(beta_rad).*sin(fpa_rad
    ).*sin(psi_rad) ...
  - lonRate_rad_s.*cos(alpha_rad).*cos(
    bank_rad).*cos(latitude_rad).*sin(
    beta_rad).*sin(eta_rad).*sin(fpa_rad)
    .*sin(psi_rad) ...
  + lonRate_rad_s.*cos(alpha_rad).*cos(
    eta_rad).*cos(latitude_rad).*sin(
    bank_rad).*sin(beta_rad).*sin(fpa_rad
    ).*sin(psi_rad)))...
./(cos(beta_rad).*(Cz_nd.*cos(alpha_rad)
     - Cx_nd.*sin(alpha_rad)).^2) ...
  + (cos(eta_rad).^2.*(sin(beta_rad).*
    Cx_nd.^2 ...
  - Cy_nd.*cos(alpha_rad).*cos(beta_rad).*
    Cx_nd + sin(beta_rad).*Cz_nd.^2 ...
  - Cy_nd.*cos(beta_rad).*sin(alpha_rad).*
    Cz_nd).*(psiRate_rad_s.*sin(beta_rad)
    .*sin(fpa_rad) ...
  - pitchRate_rad_s - fpaRate_rad_s.*cos(
    bank_rad).*cos(beta_rad).*cos(eta_rad
    ) ...
  - fpaRate_rad_s.*cos(beta_rad).*sin(
    bank_rad).*sin(eta_rad) ...
  - latRate_rad_s.*cos(fpa_rad).*cos(
    psi_rad).*sin(beta_rad) ...
```

```
+ earthRate_rad_s.*sin(beta_rad).*sin(
   fpa_rad).*sin(latitude_rad) ...
+ lonRate_rad_s.*sin(beta_rad).*sin(
   fpa_rad).*sin(latitude_rad) ...
+ latRate_rad_s.*cos(bank_rad).*cos(
   beta_rad).*cos(eta_rad).*sin(psi_rad)
    ...
+ earthRate_rad_s.*cos(fpa_rad).*cos(
   latitude_rad).*sin(beta_rad).*sin(
   psi_rad) ...
+ lonRate_rad_s.*cos(fpa_rad).*cos(
   latitude_rad).*sin(beta_rad).*sin(
   psi_rad) ...
+ latRate_rad_s.*cos(beta_rad).*sin(
   bank_rad).*sin(eta_rad).*sin(psi_rad)
    ...
+ psiRate_rad_s.*cos(bank_rad).*cos(
   beta_rad).*cos(fpa_rad).*sin(eta_rad)
    ...
- psiRate_rad_s.*cos(beta_rad).*cos(
   eta_rad).*cos(fpa_rad).*sin(bank_rad)
    ...
+ earthRate_rad_s.*cos(bank_rad).*cos(
   beta_rad).*cos(eta_rad).*cos(
   latitude_rad).*cos(psi_rad) ...
```

```
+ lonRate_rad_s.*cos(bank_rad).*cos(
   beta_rad).*cos(eta_rad).*cos(
   latitude_rad).*cos(psi_rad) ...
+ earthRate_rad_s.*cos(bank_rad).*cos(
   beta_rad).*cos(fpa_rad).*sin(eta_rad)
   .*sin(latitude_rad) ...
- earthRate_rad_s.*cos(beta_rad).*cos(
   eta_rad).*cos(fpa_rad).*sin(bank_rad)
   .*sin(latitude_rad) ...
+ lonRate_rad_s.*cos(bank_rad).*cos(
   beta_rad).*cos(fpa_rad).*sin(eta_rad)
   .*sin(latitude_rad) ...
- lonRate_rad_s.*cos(beta_rad).*cos(
   eta_rad).*cos(fpa_rad).*sin(bank_rad)
   .*sin(latitude_rad) ...
+ earthRate_rad_s.*cos(beta_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
   bank_rad).*sin(eta_rad) ...
+ latRate_rad_s.*cos(bank_rad).*cos(
   beta_rad).*cos(psi_rad).*sin(eta_rad)
   .*sin(fpa_rad) ...
- latRate_rad_s.*cos(beta_rad).*cos(
   eta_rad).*cos(psi_rad).*sin(bank_rad)
   .*sin(fpa_rad) ...
+ lonRate_rad_s.*cos(beta_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
```

```
                    bank_rad).*sin(eta_rad) ...
                - earthRate_rad_s.*cos(bank_rad).*cos(
                    beta_rad).*cos(latitude_rad).*sin(
                    eta_rad).*sin(fpa_rad).*sin(psi_rad)
                    ...
                + earthRate_rad_s.*cos(beta_rad).*cos(
                    eta_rad).*cos(latitude_rad).*sin(
                    bank_rad).*sin(fpa_rad).*sin(psi_rad)
                     ...
                - lonRate_rad_s.*cos(bank_rad).*cos(
                    beta_rad).*cos(latitude_rad).*sin(
                    eta_rad).*sin(fpa_rad).*sin(psi_rad)
                    ...
                + lonRate_rad_s.*cos(beta_rad).*cos(
                    eta_rad).*cos(latitude_rad).*sin(
                    bank_rad).*sin(fpa_rad).*sin(psi_rad)
                    ))...
                ./(Cz_nd.*cos(alpha_rad) - Cx_nd.*sin(
                    alpha_rad)).^2;

%bankRate_rad_s = wrapToPi(bankRate_rad_s);

alphaRate_rad_s = -(rollRate_rad_s.*cos(alpha_rad).*sin(
   beta_rad) ...
                  - fpaRate_rad_s.*cos(bank_rad).*cos(
                     eta_rad) ...
```

```
- pitchRate_rad_s.*cos(beta_rad) ...
- fpaRate_rad_s.*sin(bank_rad).*sin(
   eta_rad) ...
+ yawRate_rad_s.*sin(alpha_rad).*sin(
   beta_rad) ...
+ psiRate_rad_s.*cos(bank_rad).*cos(
   fpa_rad).*sin(eta_rad) ...
- psiRate_rad_s.*cos(eta_rad).*cos(
   fpa_rad).*sin(bank_rad) ...
+ latRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*sin(psi_rad) ...
+ latRate_rad_s.*sin(bank_rad).*sin(
   eta_rad).*sin(psi_rad) ...
+ earthRate_rad_s.*cos(bank_rad).*cos(
   fpa_rad).*sin(eta_rad).*sin(
   latitude_rad) ...
- earthRate_rad_s.*cos(eta_rad).*cos(
   fpa_rad).*sin(bank_rad).*sin(
   latitude_rad) ...
+ lonRate_rad_s.*cos(bank_rad).*cos(
   fpa_rad).*sin(eta_rad).*sin(
   latitude_rad) ...
- lonRate_rad_s.*cos(eta_rad).*cos(
   fpa_rad).*sin(bank_rad).*sin(
   latitude_rad) ...
```

```
+ earthRate_rad_s.*cos(latitude_rad).*
   cos(psi_rad).*sin(bank_rad).*sin(
   eta_rad) ...
+ latRate_rad_s.*cos(bank_rad).*cos(
   psi_rad).*sin(eta_rad).*sin(fpa_rad
   ) ...
- latRate_rad_s.*cos(eta_rad).*cos(
   psi_rad).*sin(bank_rad).*sin(
   fpa_rad) ...
+ lonRate_rad_s.*cos(latitude_rad).*
   cos(psi_rad).*sin(bank_rad).*sin(
   eta_rad) ...
+ earthRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*cos(latitude_rad).*cos(
   psi_rad) ...
+ lonRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*cos(latitude_rad).*cos(
   psi_rad) ...
- earthRate_rad_s.*cos(bank_rad).*cos(
   latitude_rad).*sin(eta_rad).*sin(
   fpa_rad).*sin(psi_rad) ...
+ earthRate_rad_s.*cos(eta_rad).*cos(
   latitude_rad).*sin(bank_rad).*sin(
   fpa_rad).*sin(psi_rad) ...
- lonRate_rad_s.*cos(bank_rad).*cos(
   latitude_rad).*sin(eta_rad).*sin(
```

```
                    fpa_rad).*sin(psi_rad) ...
                  + lonRate_rad_s.*cos(eta_rad).*cos(
                    latitude_rad).*sin(bank_rad).*sin(
                    fpa_rad).*sin(psi_rad))...
                  ./cos(beta_rad);

betaRate_rad_s = rollRate_rad_s.*sin(alpha_rad) -
   yawRate_rad_s.*cos(alpha_rad) ...
                + fpaRate_rad_s.*cos(bank_rad).*sin(
                  eta_rad) ...
                - fpaRate_rad_s.*cos(eta_rad).*sin(
                  bank_rad) ...
                + psiRate_rad_s.*cos(bank_rad).*cos(
                  eta_rad).*cos(fpa_rad) ...
                + psiRate_rad_s.*cos(fpa_rad).*sin(
                  bank_rad).*sin(eta_rad) ...
                - latRate_rad_s.*cos(bank_rad).*sin(
                  eta_rad).*sin(psi_rad) ...
                + latRate_rad_s.*cos(eta_rad).*sin(
                  bank_rad).*sin(psi_rad) ...
                + lonRate_rad_s.*cos(bank_rad).*cos(
                  eta_rad).*cos(fpa_rad).*sin(
                  latitude_rad) ...
                - earthRate_rad_s.*cos(bank_rad).*cos(
                  latitude_rad).*cos(psi_rad).*sin(
                  eta_rad) ...
```

```
+ earthRate_rad_s.*cos(eta_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
   bank_rad) ...
+ latRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*cos(psi_rad).*sin(fpa_rad)
   ...
- lonRate_rad_s.*cos(bank_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
   eta_rad) ...
+ lonRate_rad_s.*cos(eta_rad).*cos(
   latitude_rad).*cos(psi_rad).*sin(
   bank_rad) ...
+ earthRate_rad_s.*cos(fpa_rad).*sin(
   bank_rad).*sin(eta_rad).*sin(
   latitude_rad) ...
+ lonRate_rad_s.*cos(fpa_rad).*sin(
   bank_rad).*sin(eta_rad).*sin(
   latitude_rad) ...
+ latRate_rad_s.*cos(psi_rad).*sin(
   bank_rad).*sin(eta_rad).*sin(fpa_rad)
   ...
+ earthRate_rad_s.*cos(bank_rad).*cos(
   eta_rad).*cos(fpa_rad).*sin(
   latitude_rad) ...
- earthRate_rad_s.*cos(latitude_rad).*sin(
   bank_rad).*sin(eta_rad).*sin(fpa_rad).*
```

```
                    sin(psi_rad) ...
                - lonRate_rad_s.*cos(latitude_rad).*sin(
                    bank_rad).*sin(eta_rad).*sin(fpa_rad).*
                    sin(psi_rad) ...
                - earthRate_rad_s.*cos(bank_rad).*cos(
                    eta_rad).*cos(latitude_rad).*sin(
                    fpa_rad).*sin(psi_rad) ....
                - lonRate_rad_s.*cos(bank_rad).*cos(
                    eta_rad).*cos(latitude_rad).*sin(
                    fpa_rad).*sin(psi_rad);
%% Output
phaseout.dynamics  = [rDot_m_s, lonRate_rad_s,
   latRate_rad_s, ...
                    speedRate_m_s, psiRate_rad_s,
                       fpaRate_rad_s,...
                    pDot_rad_s2 qDot_rad_s2 rDot_rad_s2
                       ,...
                    bankRate_rad_s,alphaRate_rad_s,
                       betaRate_rad_s];

%% constraints [heatRate,dynamicPressure,lateralAccel]
fluxConstant = 5.75e-5;
heatFlux_W_m2 = (fluxConstant*sqrt(density_kg_m3/
   refDensity_kg_m3)...
    .*(speed_m_s).^(3.15))/sqrt(vehicle.noseRadius_m);
```

```
lateralAcceleration_m_s2 = hypot(Cz_nd,Cy_nd).*
   dynamicForce_N/(vehicle.vehicleMass_kg);

phaseout.path = [heatFlux_W_m2 dynamicPressure_Pa
   lateralAcceleration_m_s2];
%%


end
```

Bibliography

1. Luis Guerreiro. *Development of a Guidance and Control Design Tool for Entry Space Vehicles with Different Lift-over-Drag Ratios.* Master's thesis, Instituto Superior Técnico, University of Lisbon, Portugal, 2011.

2. Kerry D. Hicks. *Introduction to Astrodynamic Re-Entry (Second Edition).* CreateSpace Publishing, Lexington, KY, 2014.

3. Michael Grant, Ian Clark, and Robert Braun. *Rapid Entry Corridor Trajectory Optimization for Conceptual Design.* In *AIAA Atmospheric Flight Mechanics Conference*, August 2010.

4. Timothy R. Jorris. Common Aero Vehicle Reentry Trajectory Optimization Satisfying Waypoint and No-fly Constraints. PhD thesis, Air Force Institute of Technology, September 2007.

5. Kevin Bollino. Optimal Guidance Command Generation and Tracking for Reusable Launch Vehicle Reentry. PhD thesis, Naval Postgraduate School, Monterey, CA, 2006.

6. Claus Weiland. Aerodynamic Data of Space Vehicles. Springer-Verlag, Berlin Heidelberg, Germany, 2014.

7. Marie Albisser. Identification of aerodynamic coefficients from free flight data. PhD thesis, École Polytechnique de l'Université de Lorraine, Lorraine, France, 2015.

8. Michael A. Patterson and Anil V. Rao. *GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using Hp-Adaptive Gaus-*

sian Quadrature Collocation Methods and Sparse Nonlinear Programming. ACM Trans. Math. Softw., 41(1), October 2014.

9. H.E. Wang and S.T. Chu. Variable-lift re-entry at superorbital and orbital speeds. *AIAA Journal*, 1(5):1047–1055, May 1963.

10. Adolf Busemann Robert D. Culp, Nguyen X. Vinh. *Optimum Three Dimensional Atmospheric Entry From the Analytical Solution of Chapman's Exact Equations.* Technical Report NASA CR-132571, National Air and Space Administration, 1976.

11. Corey J. Lee. *Hypersonic Vehicle Control and Trajectory Determination Through the Application of Artificial Intelligence.* Master's thesis, Air Force Institute of Technology, 2020.

12. Matthew J. Abrahamson. *Boost Through Reentry Trajectory Planning For Maneuvering Reentry Vehicles.* Master's thesis, Massachusetts Institute of Technology, 2008.

13. Derrick G. Tetzman. *Simulation and Optimization of Spacecraft Re-entry Trajectories.* Master's thesis, University of Minnesota, May 2010.

14. F. Landis Markley and John Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control.* Springer, New York, NY, 2014.

15. Dominick Andrisani. *The Standard Atmosphere.* Retrieved on 27 September 2020, from `https://engineering.purdue.edu/~andrisan/Courses/AAE490A_S2002/Atmosphere.pdf`.

16. Luisa D Fairfax Luke S Strohm, JD Vasile and Frank E Fresconi. *Trajectory Shaping for Quasi-Equilibrium Glide in Guided Munitions*. Technical Report ARL-TR-8749, Army Research Laboratory, 2019.

17. Tadeusz J. Masternak. Multi-Objective Trajectory Optimization of a Hypersonic Reconnaissance Vehicle with Temperature Constraints. PhD thesis, Air Force Institute of Technology, December 2014.

18. Bernard Etkin. *Dynamics of Atmospheric Flight*. John Wiley & Sons, Inc., Toronto, ON, 1972.

19. G.L. Winchenbach. *Aerodynamic Testing in a Free Flight Spark Range*. Technical Report WL-TR-1997-7006, Wright Laboratory Armament Directorate, 1997.

20. Angelo Miele. *Flight Mechanics : Theory of Flight Paths*. Dover Publications, Mineola, NY, 2015.

21. Sargur Srihari. *Lectures on Deep Learning*. Tutorial Slides, Spring 2020. Retrieved on 10 January 2021, from `https://cedar.buffalo.edu/~srihari/CSE676/`.

22. Jose Principe. *Lecture Notes on Function Approximation With MLPs, Radial Basis and Support Vector*. Course Website. Retrieved on 11 October 2020, from `http://www.cnel.ufl.edu/courses/EEL6814/chapter5.pdf`.

23. John D. Anderson. *Introduction to Flight (Fifth Edition)*. McGraw-Hill, New York, NY, 2005.

24. M.E. Tuaber and K Sutton. Stagnation point radiative heating relations for earth and mars entries. *Journal of Spacecraft and Rockets*, pages 40–42, Jan-Feb 1991.

25. Reinald G. Finke. *Calculation of Reentry Vehicle Temperature History*. Technical Report IDA PAPER P-2395, Institute for Defense Analyses, 1990.

26. Gilbert Strang. Introduction to Linear Algebra. Wellesley-Cambridge Press, Wellesley, MA, 2016.

27. Joseph A. Paradiso. *Application of Linear Programming to Coordinated Management of Jets and Aerosurfaces For Aerospace Vehicle Control*. Technical Report CSDL-R-2065, The Charles Stark Draper Laboratory, 1988.

28. N.N. *Aerodynamic Design Data Book - Orbiter Vehicle STS-1*. Technical Report SD-72-SH-0060-1, Rockwell International, 1980.

29. Michael W. Oppenheimer. email communications, December 2020.

30. Fengjin Liu, William W. Hager, and Anil V. Rao. Adaptive mesh refinement method for optimal control using nonsmoothness detection and mesh size reduction. *Journal of the Franklin Institute*, 352(10):4081–4106, 2015.

31. Anil V. Rao. *Introduction to the Optimal Control Software, GPOPS-II*. Tutorial Slides, July 2018. Retrieved on 18 October 2020, from `https://www.siue.edu/~juliu/cbms18/img/Rao2.pdf`.

32. J.T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. SIAM Press, Philadelphia, PA, 2009.

33. Daigoro Ito, Jennifer Georgie, John Valasek, and Donald T. Ward. *Reentry Vehicles Flight Controls Design Guidelines - Dynamic Inversion*. Technical Report NASA-TP-2002-210771, Flight Simulation Laboratory, Texas A&M University, 1997.