RICE UNIVERSITY

Comparison of Guidance Methods for Autonomous Planetary Lander

by

Wyatt Harris, 2Lt USAF

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

APPROVED, THESIS COMMITTEE:

Pol Spanos
L.B. Ryon Chair in Engineering

Andrew Meade
Mechanical Engineering & Materials Science

Andrew Dick
Mechanical Engineering & Materials Science

Nazareth Bedrossian
Mechanical Engineering & Materials Science

HOUSTON, TEXAS
APRIL 2011

Abstract

Comparison of Guidance Methods for Autonomous Planetary Lander

by

Wyatt Harris

Guidance methods for a powered descent planetary lander are examined. Several guidance laws are developed and analyzed including Apollo derived, modified Apollo, time efficient and fuel efficient methods. These laws are implemented using two different methods: Apollo approach, and real time. The equations of motion for an arbitrary planetary lander with sloshing fuel and a gimbaled nozzle are derived using a Newtonian approach. A six degree of freedom (6DOF) simulation of the vehicle system dynamics is programmed in Simulink to test the various guidance methods. The guidance methods are assessed using several different trajectories.

Acknowledgements

I would like to thank Dr. Pol Spanos and Rice University for giving me the opportunity to obtain a masters degree at Rice, and specifically Dr. Spanos for his steadfast research/scholastic guidance. Special thanks to Dr. Nazareth Bedrossian for the opportunity to do my thesis work at Draper Labs, aka the 'Naz Academy.' The Naz Academy, while a little different than my first academy experience, taught me some great life lessons (all while not leaving me too burnt out...). I learned a lot about the world (that is, according to Naz), and even a little about engineering. I am very grateful for Naz's mentorship and assistance with completing my thesis.

I am very grateful to Christine Appleby of Draper Labs for her help with modeling the mechanical system. I would also like to thank Lee Yang of Draper Labs for helping me learn about guidance methods.

I have the greatest parents and brother, Frank, that I could ever ask for. It is because of them that I am where I am today, and I love and thank them for everything they have done for me.

I am lucky to have such a wonderful girlfriend, Hilary. Her love and support has meant more to me than she knows, and I am truly grateful for being with her. She has been an immense source of motivation for me—I thank her for being so amazing.

I would also like to say thanks to all my friends for always being there for me. In particular, Tim- we spent far more time together than should be spent with anyone: living in the same place, carpooling, and working together. Somehow we managed pretty well though; I like to think it is because Tim took his anger at me out on other drivers.

Lastly, thanks to the United States Air Force for allowing me to go to graduate school for the first two years of my career.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

Contents

List of Figures

List of Tables

1 Introduction

Future unmanned missions to other planetary bodies such as the moon or Mars will likely require vehicles to be capable of operating over their entire flight regimes without human input. For vehicles with rocket engines, where planetary descent is primarily 'powered,' the rocket engine is used to slow the descent of the vehicle to a soft landing. For applications where a specific landing site is required or hazard avoidance may be necessary, a system is needed to autonomously maneuver the vehicle to a desired location. This system is referred to as Guidance, Navigation, and Controls (GNC), and is responsible for getting the vehicle from a current position (initial state) to a desired final position (target state) [1].

A common configuration for a powered descent planetary lander involves using a gimbaled rocket engine. For this configuration, the engine can swivel around its gimbal point, thereby changing the direction of the thrust vector acting on the vehicle body. Controlling the gimbal (swivel) of the rocket engine allows the vehicle to change the direction of its flight path. The engine is throttle-able, meaning the magnitude of thrust provided by the engine can be prescribed between the minimum and maximum limits of the engine. Controlling the throttle of the engine allows the vehicle to change its velocity. A gimbaled engine of this sort is capable of simultaneously slowing the descent of the vehicle and maneuvering it to its target state.

Powered descent vehicles using gimbaled engines require storage tanks containing liquid rocket propellant. As the vehicle descends and maneuvers, this liquid propellant has a propensity to slosh back and forth, causing a disturbance force on the

vehicle that can affect its flight path or stability [2]. It is important to take this fact into account when simulating a vehicle and analyzing GNC methods.

The methods, simulation, and analysis detailed herein assume a powered descent vehicle on this configuration, with a gimbaled rocket engine and sloshing propellant.

1.1 GNC Overview

GNC refers to the complete framework of software and hardware responsible for maneuvering a vehicle from an initial state to a desired target state. The framework consists of several subsystems that together read feedback information in the form of sensor data, develop a feasible trajectory, and convert this information into control inputs that command actuators to get the vehicle to the desired target state [3].

The subsystems of this framework are guidance, navigation, and control. They are described in the following sections.

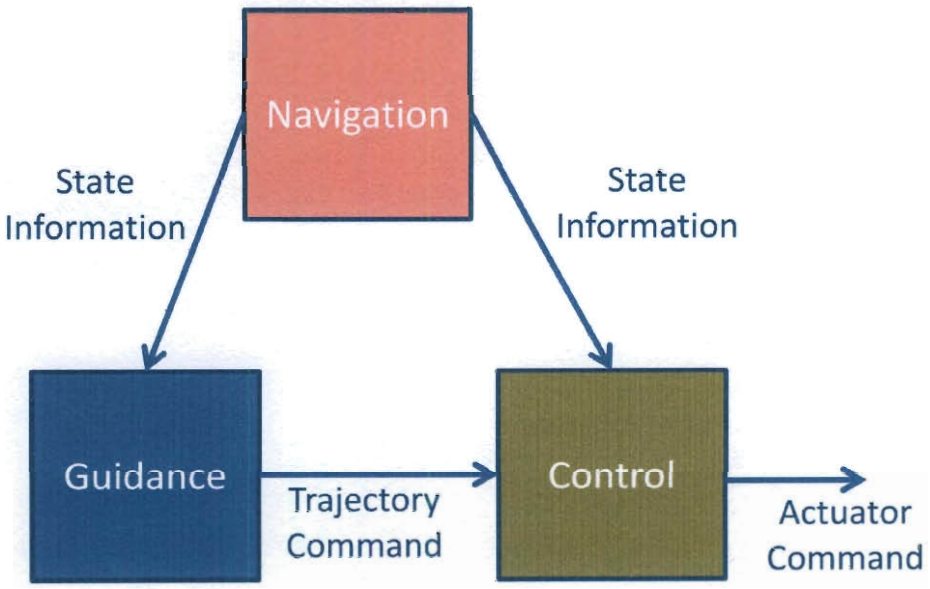

Figure 1-1: Block Diagram Showing Subsystems of Typical GNC Architecture

The specific commands generated by each subsystem are dependent on how the architecture is implemented. The architecture implemented for the Apollo space program is considered the legacy method, and has been widely used since the Apollo program [4]. The commands generated by each subsystem for this method are shown in the following sections. Alternative methods of implementation are possible; however their differences lie primarily within the control subsystem. The analysis and simulation detailed herein assumes the legacy implementation.

1.1.1 Navigation

The navigation subsystem of GNC is responsible for providing information on the current state of the vehicle to other GNC subsystems. Depending of the receiving subsystem, this is inertial position ($\hat{\boldsymbol{r}}_{\boldsymbol{I}}$), velocity ($\hat{\boldsymbol{v}}_{\boldsymbol{I}}$), acceleration ($\hat{\boldsymbol{a}}_{\boldsymbol{I}}$), rotation ($\hat{\boldsymbol{q}}_{\boldsymbol{I}\rightarrow\boldsymbol{b}}$), and/or rotation rate ($\hat{\boldsymbol{\omega}}_{\boldsymbol{b}}$) state information, and is an estimate of the actual state information based on data from sensors such as inertial measurement units (IMU), Radar, Lidar, etc [5] [3].

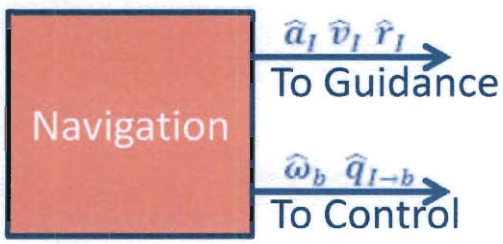

Figure 1-2: Navigation Subsystem of GNC Architecture

For simulation purposes, no algorithms or software are necessary for navigation, as 'sensor' data is simply fed back from plant dynamic. However, sensor error can be taken into account by modeling it using stochastic techniques in the simulation program.

1.1.2 Guidance

The guidance subsystem of GNC is responsible for developing a trajectory command to get the vehicle from where it is (initial state) to where it wants to be (target state) [5]. Using the legacy implementation, this trajectory command comes in the form of an acceleration command ($\boldsymbol{\lambda}$ for direction and $|\boldsymbol{a_T}|$ for magnitude), a jerk command ($\dot{\boldsymbol{\lambda}}$), and the time value associated with these commands (t_λ).

Figure 1-3: Guidance Subsystem of GNC Architecture

Guidance uses current state information from navigation and target state information that is pre-loaded a priori or determined using a separate subroutine (such as hazard avoidance software). With this information, it solves for a trajectory to get from the current state to the target state. The methods by which this is done are the subject of this work.

1.1.3 Control

The control subsystem is responsible for taking the attitude command from steering and referencing it with the current state information from navigation to create a command for the actuator (engine) in the form of a gimbal command ($\boldsymbol{\beta}$), and thrust command ($\boldsymbol{T}$).

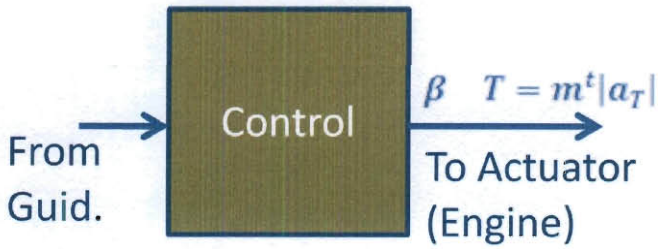

Figure 1-4: Control Subsystem of GNC Architecture

For the legacy implementation method, thrust is the acceleration command from guidance fed forward and scaled by the mass of the vehicle ($\boldsymbol{m^t|a_T|}$). A basic proportional derivative (PD) controller is used on the attitude and rate errors to generate a gimbal command. Command shaping, also known as a steering subroutine, converts the trajectory command from guidance into a command useable by control. The steering subroutine generates a body rotation quaternion command ($\boldsymbol{q_{I\to b}^{cmd}}$) and rotation rate command ($\boldsymbol{\omega_b^{cmd}}$), which are in turn used by the PD controller [6] [7].

For the simulation and analysis of this work, a legacy controller using stable gains is used and held constant throughout.

1.2 Guidance Heritage

The goal of guidance is to find a trajectory to get a vehicle from an initial state to a desired target state. This problem is further complicated by the fact that the vehicle trajectory cannot violate physical constraints such as fuel, thrust, or flight path limits. There are many ways to formulate trajectories, but using the legacy GNC architecture, all methods involve developing an acceleration profile for the vehicle. If this profile is followed, the vehicle will follow the desired trajectory to get to its target state.

Modern GNC methods, and specifically guidance methods, are largely based on techniques developed during the Apollo lunar program in the 1960's. Since the Apollo program, powered descent vehicles have generally used some modified version of Apollo guidance. However, Apollo or modified Apollo methods are not fuel or time optimal. That is, they do not guarantee the vehicle will reach its destination using as little fuel as possible or in as little time as possible. For these reason, optimal guidance methods are of significant interest in recent years as they potentially offer fuel and/or time of flight savings for powered descent vehicles [8].

1.2.1 Apollo

The Apollo Lunar Lander was the first vehicle of its type, using only rocket power to descend to a soft landing [9]. As such, there were no previously developed methods for powered descent guidance, so the Apollo designers had to create their own. These Apollo guidance methods are discussed in [4] and [5].

The basic approach to guidance for the terminal descent phase of the Apollo lander is to solve a two point boundary value problem using a quadratic polynomial acceleration profile (see Chapter 3). To have a closed form solution for this problem though, acceleration in the vertical direction is constrained to be linear [8]. For the specific case of the moon landings, this approach proved sufficient, and was about as much the Apollo era flight computers could handle.

However, this approach does not consider any physical constraints on the vehicle, and is not fuel or time optimal. Since the Apollo program, this method has been modified so that constraints can be applied, and fuel and time efficiency considered.

1.2.2 Modified Apollo

The biggest downside of the Apollo method is that it is closed form and does not provide any free parameters. While this is computationally efficient, constraints cannot be applied, nor can fuel or time performance be improved. By taking away the linear vertical acceleration stipulation applied to Apollo guidance, a free parameter is formed in the acceleration two point boundary value problem, as there is one more variable than equation (see Chapter 3 for development). This free parameter is generally taken to be time of flight. This is the basis of modified Apollo guidance methods. Furthermore, the acceleration profile is not necessarily a quadratic polynomial, but is generally taken to be a linear, quadratic, or cubic polynomial function.

Vehicle physical constraints can now be applied by constraining the time of flight parameter. Fuel or time performance can be improved by using a search method to find an ideal time of flight.

1.2.3 Optimal Guidance

Though performance can be improved with the modified Apollo guidance, the efficiency of the method is still restricted by the shape of the acceleration profile (shape of polynomial). Optimal guidance refers to any method used to generate a trajectory that optimizes some flight parameter such as fuel use or time of flight.

In [10] and [11] it is shown that there is a closed form solution for the specific case of one dimensional powered descent (see Section 3.1.4 for further discussion and development). For three dimensional powered descent, however, a closed form solution does not exist [8]. To find an optimal guidance solution for a three dimensional case,

optimization techniques must be used. [8], [12], [13], [14], [15], [16], [17], and [18] discuss and develop various methods for solving the optimal guidance problem. But these techniques are computationally intensive, and difficult to implement in real time. The optimal acceleration profile is shown to have a bang-coast-bang structure where the engine is either on (maximum thrust) or off (minimum thrust) [18]. This on/off style of acceleration is the basis for the method described in Section 3.1.4.

1.3 Contributions

This thesis provides a complete derivation of the equations of motion for a lander vehicle with a gimbaled nozzle and sloshing propellant. These equations of motion are derived for six degrees of freedom, and are arranged in mass matrix format for simultaneous calculation of state variables (necessary for computer simulation). The equations of motion are developed to be used with prescribed engine torque (for gimbal control), but they are also simplified so gimbal angle can be directly prescribed. Methods for planet centered gravity and applying aerodynamic forces are included.

A max-min acceleration profile used for providing trajectories to get from an initial state to a target state is developed for vertical and horizontal maneuvers and is shown to be fuel efficient. The merits (time of flight and fuel use) of this acceleration profile relative to other acceleration profiles are examined for a vertical maneuver and horizontal maneuver.

A method for applying vehicle physical constraints to guidance trajectories while calculating a time efficient trajectory is described. Two methods for implementing guidance are discussed. The first is an Apollo approach implementation similar to that

used for the Apollo moon lander. The second is a new real time method, where guidance trajectories are calculated in real time. The relative merits of each method are compared using three different flight path trajectories.

Finally, a method for finding a vehicle's landing zone 'footprint' is developed. This method utilizes a guidance based approach to determine the potential landing area within which a vehicle can land. It is found that parameters such as time of flight, propellant mass and initial velocity have a large affect on a vehicle's footprint.

2 Planetary Lander Dynamics

The algorithms for autonomous vehicles, such as those used for Guidance and Control (G&C) processes depend on the dynamics of the system. Therefore, it is logical to test flight software in an environment representative of the actual flight system before real life flight occurs. To validate flight software using computational techniques, a realistic model of the system (plant) in question should be used for simulation.

To create a simulation, a mathematical model of the system is first generated, using acceptable simplifying assumptions and restrictions if necessary. This model describes the dynamics of the system through a set of equations of motion (EOM's). The EOM's are then coded in computer software to create a simulation. The simulation model accepts inputs from and provides relevant outputs for G&C processes. The simulation model should also provide relevant outputs that give the user a complete understanding of the system response.

The goal of this chapter is to develop a complete set of nonlinear EOM's that describe the dynamics of an arbitrary rigid spacecraft with sloshing propellant and a gimbaled nozzle. Furthermore, the EOM's are rearranged in a manner conducive to computer coding. These equations are derived with six degree of freedom (6DOF) motion taken into consideration: translation in three dimensions (e.g. X, Y, and Z), and rotation in three dimensions (e.g. pitch, roll, and yaw).

The system examined for this derivation is the moon lander shown in Figure 2-1. The parts of this system include: the vehicle airframe (dry mass of vehicle minus mass of engine), the engine, and propellant 'slosh' masses. The vehicle airframe and engine are

considered rigid bodies, and the propellant slosh masses are considered point masses (and therefore have no inertia).

Typically, for systems with cylindrical propellant tanks, each propellant mass is split into two pieces: a sloshing component, and a stationary 'non-sloshing' component. But for a system with spherical propellant tanks modeled using a spring-mass-damper all of the propellant in a given tank is considered to be sloshing, and is therefore part of the slosh mass. The particular vehicle examined has four separate slosh masses, but equations of this derivation are derived for any n number of slosh masses.

There are three reference frames considered within the scope of this derivation: an inertial frame, a body frame, and an engine frame. The origin of the inertial frame can be anywhere, and is typically chosen to be a convenient location on the ground. The inertial frame $i\hat{X}$ vector points in the local vertical; opposite the direction of gravity, assuming a flat Earth. A planet centered model with non-constant gravity can easily be applied, if necessary (see Section 2.8 below). The origin of the body frame is chosen to be the gimbal point of the rocket engine, because it simplifies aspects of the derivation. The body frame $b\hat{x}$ vector points towards the fore of the vehicle, along the centerline. The origin of the engine frame is chosen to be the engine's center of mass, and the engine frame $e\hat{x}$ vector points towards the gimbal point.

This derivation is quasi-static with regards to changing mass parameters such as propellant mass and location. This means that for the derivation of EOM's herein, all mass related parameters are considered constant, so their derivatives are zero. Changing mass is taken into account in the simulation of the system: mass parameters are

recomputed every time step separately and are fed into the EOM's as constants for that time step.

For this derivation, a Newtonian approach is taken. This is not the only approach that could be used; other approaches using Lagrange's equations or Kane's method have been examined, but it is evident that a Newtonian approach is most convenient for this system, as it greatly simplifies how external forces such as thrust or gravity are applied. The relevant equations of motion are derived from the relationship force equals mass multiplied by acceleration,

$$\vec{F} = m\vec{a}. \tag{2.1}$$

In order to solve for angular equations of motion, the following relationship is used:

$$\vec{M} = \underline{I}\vec{\alpha} + \vec{\omega} \times (\underline{I} \cdot \vec{\omega}), \tag{2.2}$$

where $\vec{M}$ is a moment, $\underline{I}$ is an inertia, $\vec{\alpha}$ is an angular acceleration, and $\vec{\omega}$ is an angular velocity.

Multibody EOM's are necessary to develop a complete understanding of the dynamics of the spacecraft. These EOM's can be considered from any location with respect to any frame, but this derivation chooses the most relevant locations and frames. The EOM's can be used to find the relevant parameters of the system:

I. Translational acceleration of the gimbal, with respect to the inertial frame.

II. Rotational acceleration of the body frame, with respect to the inertial frame.

III. Rotational acceleration of the engine frame, with respect to body frame.

IV. Translational acceleration of an arbitrary n^{th} slosh mass.

The complete description and derivation of these EOM's can be found in the following sections. The goal of the derivation—to develop EOM's for use in computer simulation—is kept in mind throughout: the final EOM's are placed in a format conducive to placement in matrix form, which is useful for coding. This form consists of state accelerations and their coefficients on one side of the equation, and all other terms on the other side of the equation.

For this derivation, all EOM's are solved in the body frame. This is done because it is a much simpler task to apply forces on the body in the body frame, and examine slosh displacement with respect to the body frame, as opposed to the inertial frame.

2.1 Notation

Vector notation is used to simplify the process of generating six degree of freedom (6DOF) equations of motion. Vectors are denoted by an arrow, ex. $\vec{r}$. Dyadics are denoted with an underscore, ex. $\underline{I}$. Scalars have no accent.

A brief overview of common variables used in this derivation:

I. The variables $\vec{a}$, $\vec{v}$, and $\vec{r}$ are used to denote translational acceleration, velocity, and position, respectively.

II. For propellant slosh, the slosh center of mass (CM) position vector is broken up into two components: an unchanging vector to the undisplaced CM, denoted by l, and a component that changes, denoted by d.

III. The variables α and ω denote angular acceleration and angular velocity, respectively.

IV. The variable φ is used to describe angle of attack of the vehicle, and β is used to describe angular rotation of the engine.

There are three frames of reference for this problem. There is an inertial frame (denoted by *i*), a body frame whose origin is at the engine gimbal point (denoted by *b*), and an engine frame whose origin is the engine CM (denoted by *e*). These frames are shown in Figure 2-1.

For acceleration and velocity vectors (translational or angular), two superscripts to right and left of the variable indicate the frame of interest and what it is translating/rotating about. For example, the variable ${}^{i}\vec{\alpha}^{b}$ represents the angular acceleration of the body frame with respect to the inertial frame.

For position vectors, a single superscript with two characters separated by a fraction symbol is used to indicate the point which the vector begins at and terminates at. For example: $\vec{r}^{\,\frac{Acm}{G}}$, this variable describes the vector *from* the engine gimbal, *to* the airframe CM.

The following relationship is used for vector differentiation. It describes how the derivative of a vector can be taken in another frame,

$$\frac{{}^{i}d}{dt}\left(\vec{r}^{\,\frac{Acm}{G}}\right) = \frac{{}^{b}d}{dt}\left(\vec{r}^{\,\frac{Acm}{G}}\right) + {}^{i}\vec{\omega}^{b} \times \vec{r}^{\,\frac{Acm}{G}}. \qquad (2.3)$$

When rearranging EOM's for placement in matrix notation, cross product notation is used. It is necessary to use cross product matrix notation in order to split up cross products that contain both a state and coefficient. Using cross product matrix notation, the cross product of two vectors $\vec{v}$ and $\vec{u}$ becomes the skew symmetric matrix of $\vec{v}$ times $\vec{u}$,

$$\vec{v} \times \vec{u} \rightarrow \vec{v}^x \vec{u}, \tag{2.4}$$

where the skew symmetric matrix of the vector $\vec{v}$ is

$$\vec{v}^x = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}. \tag{2.5}$$

2.2 Problem Setup

Figure 2-1 shows an arbitrary vehicle with sloshing fuel and a gimbaled nozzle, as well as each of the reference frames and their origins. Below the figure is a list of terms used throughout the derivation.

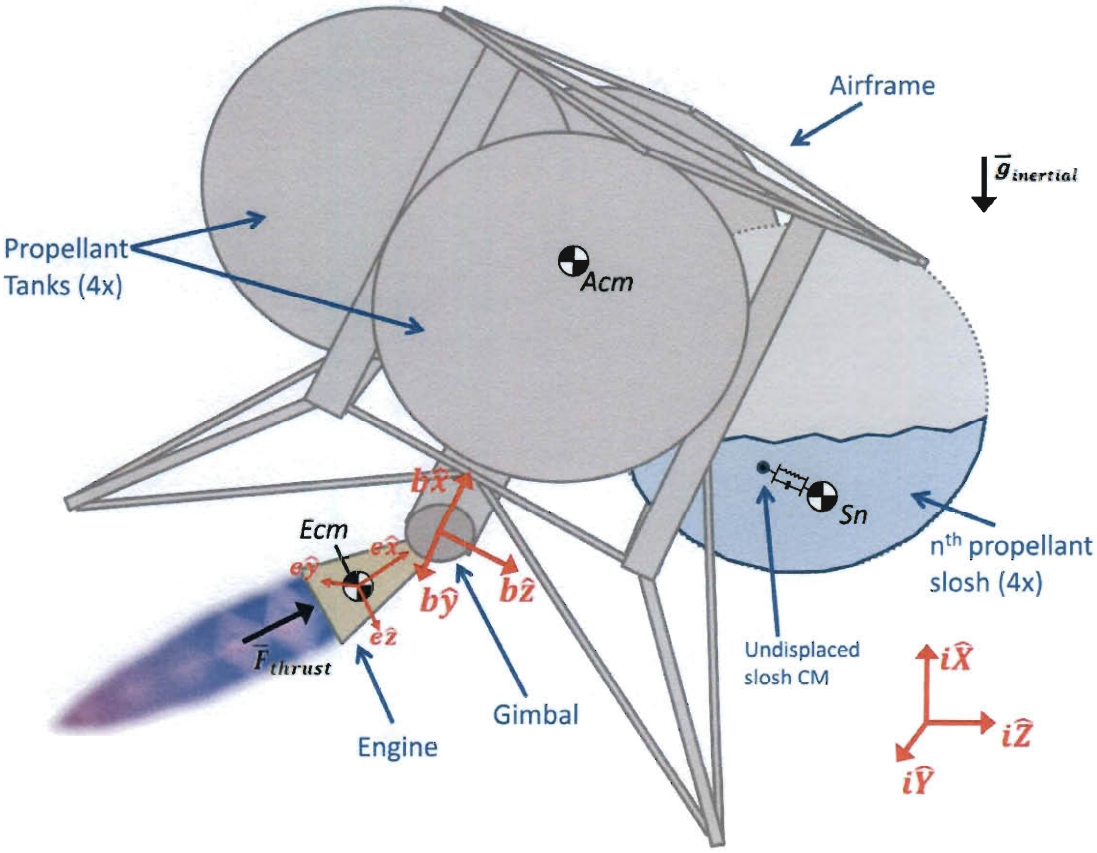

Figure 2-1: Problem Geometry and Reference Frames

List of Terms:

$i\hat{X}, i\hat{Y}, i\hat{Z}$	- Unit vectors of inertial reference frame.
$b\hat{x}, b\hat{y}, b\hat{z}$	- Unit vectors of body reference frame. $b\hat{x}$ points to the fore of the vehicle, along the centerline.
$e\hat{x}, e\hat{y}, e\hat{z}$	- Unit vectors of inertial reference frame. $e\hat{x}$ points to the gimbal.
Acm, Ecm, Sn	- Airframe center of mass, engine center of mass, and n^{th} slosh center of mass, respectively.

$\vec{r}^{\,\frac{Acm}{G}}, \vec{r}^{\,\frac{Ecm}{G}}$ - Position vector from body frame origin (rocket engine gimbal) to center of mass of airframe and engine respectively.

$\vec{l}_{sn}$ - Position vector from the gimbal to the undisplaced/equilibrium center of mass of n-th slosh mass.

$\vec{d}_{sn}$ - Displacement vector of n-th slosh mass from equilibrium position of associated slosh mass.

$\vec{r}^{\,\frac{Sn}{G}}$ - Position vector from body frame origin (rocket engine gimbal) to center of mass of n-th slosh mass:

$$\vec{r}^{\,\frac{Sn}{G}} = \vec{l}_{S_n} + \vec{d}_{S_n}$$

$\vec{r}^{\,\frac{Ccm}{G}}$ - Position vector from body frame origin (rocket engine gimbal) to the composite center of mass of the entire system (including fuel):

$$\vec{r}^{\,\frac{Ccm}{G}} = \frac{m^a \vec{r}^{\,\frac{Acm}{G}} + m^e \vec{r}^{\,\frac{Ecm}{G}} + \sum_1^n m^{sn} \vec{r}^{\,\frac{Sn}{G}}}{m^t}$$

${}^i\vec{a}^g$ - Translational acceleration of the body frame origin (engine gimbal).

${}^i\vec{\alpha}^b$ - Angular acceleration of body frame about inertial frame.

${}^i\vec{\alpha}^e$ - Angular acceleration of engine frame about inertial frame.

${}^b\vec{\alpha}^e$ - Angular acceleration of engine frame about body frame:

$${}^b\vec{\alpha}^e = \begin{bmatrix} -\ddot{\beta}_y \sin(\beta_z) - \dot{\beta}_y \dot{\beta}_z \cos(\beta_z) \\ \ddot{\beta}_y \cos(\beta_z) - \dot{\beta}_y \dot{\beta}_z \sin(\beta_z) \\ \ddot{\beta}_z \end{bmatrix}$$

${}^i\vec{\omega}^b$ - Angular velocity of body frame about inertial frame.

${}^{i}\vec{\omega}^{e}$ - Angular velocity of engine frame about inertial frame.

${}^{b}\vec{\omega}^{e}$ - Angular velocity of engine frame about body frame:

$$ {}^{b}\vec{\omega}^{e} = \begin{bmatrix} -\dot{\beta}_y \sin(\beta_z) \\ \dot{\beta}_y \cos(\beta_z) \\ \dot{\beta}_z \end{bmatrix} $$

${}^{i}\vec{\varphi}^{b}$ - XYZ angle of rotation body frame to inertial frame.

${}^{b}\vec{\beta}^{e}$ - XYZ angle of rotation from engine frame to body frame:

$$ {}^{b}\vec{\beta}^{e} = \begin{bmatrix} 0 \\ \beta_y \\ \beta_z \end{bmatrix} $$

m^a - Mass of airframe.

m^e - Mass of engine.

m^{sn} - Equivalent mass of *n*-th slosh mass. For a spherical tank, the entire propellant mass is considered slosh.

m^t - Total mass of vehicle, including propellant (wet mass):

$$ m^t = m^a + m^e + \sum_{1}^{n} m^{sn} $$

$\underline{I}^{\frac{A}{Acm}}$ - Inertia dyadic of airframe about center of mass of airframe.

$\underline{I}^{\frac{E}{Ecm}}$ - Inertia dyadic of engine about center of mass of engine.

$\vec{g}_{body}$ - Acceleration due to gravity, in the body frame. This is computed by:

$$ \vec{g}_{body} = DCM_{i \to b}(\vec{g}_{inertial}) $$

Where $DCM_{i\to b}$ is the Direction Cosine Matrix from the inertial frame to the body frame.

$\vec{F}_{thrust}$ - Thrust of rocket engine in the body frame.

$\vec{F}_{drag}$ - Aerodynamic drag (including wind force) acting on the vehicle.

$\vec{F}^{sys}$ - Sum of external forces on system in the body frame.

$\vec{M}^{\frac{sys}{Ccm}}$ - Sum of external moments on system, about composite center of mass of the system.

$\vec{M}^{\frac{E}{G}}$ - Sum of external moments on engine, about gimbal.

$\vec{F}^{sn}$ - Sum of external forces on n-th slosh mass.

$\vec{\tau}_{\beta}$ - Torque applied by gimbal motor.

ω_{sn} - Equivalent frequency of n-th slosh mass.

ζ_{sn} - Equivalent damping ratio of n-th slosh mass.

2.2.1 Engine Relative Parameters

This section provides a more detailed explanation of the engine relative parameters $\vec{F}_{thrust}$, ${}^{b}\vec{\alpha}^{e}$, ${}^{i}\vec{\alpha}^{e}$, ${}^{b}\vec{\omega}^{e}$, and ${}^{i}\vec{\omega}^{e}$.

To express $\vec{F}_{thrust}$as a vector acting on the body frame, first consider that $\vec{F}_{thrust}$ expressed in the engine frame is

$$\vec{F}^{e}_{thrust} = (F_{thrust})e\hat{x}. \tag{2.6}$$

That is, engine thrust always points in the local vertical, toward the gimbal point. To convert this to the body frame, $\vec{F}^{E}_{thrust}$is multiplied by the engine to body rotation matrix,

$$\vec{F}_{thrust} = {}^{b}R^{e}\,\vec{F}^{e}_{thrust}, \tag{2.7}$$

$${}^{b}R^{e} = \begin{bmatrix} \cos(\beta_z)\cos(\beta_y) & -\sin(\beta_z) & \cos(\beta_z)\sin(\beta_y) \\ \sin(\beta_z)\cos(\beta_y) & \cos(\beta_z) & \sin(\beta_z)\sin(\beta_y) \\ -\sin(\beta_y) & 0 & \cos(\beta_y) \end{bmatrix}. \tag{2.8}$$

This is a body fixed rotation about the $e\hat{z}$ axis first, followed by a rotation about the $e\hat{y}$ axis.

The angular acceleration of the engine with respect to the body frame, ${}^{b}\vec{\alpha}^{e}$, is found using the body to engine rotation matrix from above, and is developed in [19], as

$${}^{b}\vec{\alpha}^{e} = \begin{bmatrix} -\ddot{\beta}_y \sin(\beta_z) \\ \ddot{\beta}_y \cos(\beta_z) \\ \ddot{\beta}_z \end{bmatrix} + \begin{bmatrix} -\dot{\beta}_y\dot{\beta}_z \cos(\beta_z) \\ -\dot{\beta}_y\dot{\beta}_z \sin(\beta_z) \\ 0 \end{bmatrix}. \tag{2.9}$$

Note that this expression is only valid in the body frame.

The angular acceleration of the engine with respect to the inertial frame, ${}^{i}\vec{\alpha}^{e}$, is

$$\begin{aligned} {}^{i}\vec{\alpha}^{e} &= {}^{i}\vec{\alpha}^{b} + {}^{b}\vec{\alpha}^{e} + {}^{i}\vec{\omega}^{b} \times {}^{b}\vec{\omega}^{e} \\ &= {}^{i}\vec{\alpha}^{b} + \begin{bmatrix} -\ddot{\beta}_y \sin(\beta_z) \\ \ddot{\beta}_y \cos(\beta_z) \\ \ddot{\beta}_z \end{bmatrix} + \begin{bmatrix} -\dot{\beta}_y\dot{\beta}_z \cos(\beta_z) \\ -\dot{\beta}_y\dot{\beta}_z \sin(\beta_z) \\ 0 \end{bmatrix} + {}^{i}\vec{\omega}^{b} \times {}^{b}\vec{\omega}^{e}. \end{aligned} \tag{2.10}$$

The angular velocity of the engine with respect to the inertial frame, ${}^{b}\vec{\omega}^{e}$, is also found with the body to engine rotation matrix,

$${}^{b}\vec{\omega}^{e} = \begin{bmatrix} -\dot{\beta}_y \sin(\beta_z) \\ \dot{\beta}_y \cos(\beta_z) \\ \dot{\beta}_z \end{bmatrix}. \tag{2.11}$$

This expansion is only valid in the body frame.

The angular velocity of the engine with respect to the inertial frame, ${}^{i}\vec{\omega}^{e}$, is

$$ {}^{i}\vec{\omega}^{e} = {}^{i}\vec{\omega}^{b} + {}^{b}\vec{\omega}^{e} = {}^{i}\vec{\omega}^{b} + \begin{bmatrix} -\dot{\beta}_y \sin(\beta_z) \\ \dot{\beta}_y \cos(\beta_z) \\ \dot{\beta}_z \end{bmatrix}. \tag{2.12} $$

2.3 Translational Acceleration

This section details the derivation for the of the translational acceleration equation of motion of the gimbal. For the purpose of simulating the dynamics of the vehicle, the gimbal point is a convenient location to track as it is a known point on the vehicle that does not change over time with respect to the body frame or the engine frame.

To begin, Newton's equation for force on the system, $\vec{F}^{sys}$, is considered,

$$ \vec{F}^{sys} = m^{t}\, {}^{i}\vec{a}^{Ccm}. \tag{2.13} $$

The sum of the forces on the system are equal to the total mass of the system times the acceleration of the composite center of mass of the system. The acceleration of the composite center of mass of the system, ${}^{i}\vec{a}^{Ccm}$, is the second derivative taken in the inertial frame of the position vector from the inertial frame to the composite center of mass. That is,

$$ {}^{i}\vec{a}^{Ccm} = \frac{{}^{i}d^2}{dt^2}\left(\vec{r}^{\,\frac{Ccm}{i_o}}\right). \tag{2.14} $$

Because the gimbal is chosen as the point of interest for this derivation, the vector $\vec{r}^{\,\frac{Ccm}{i_o}}$ can be split up to include the gimbal point,

$$ \vec{r}^{\,\frac{Ccm}{i_o}} = \vec{r}^{\,\frac{G}{i_o}} + \vec{r}^{\,\frac{Ccm}{G}}. \tag{2.15} $$

The vector from the gimbal to the composite center of mass, $\vec{r}^{\frac{Ccm}{G}}$, can be split up into the sum of its parts, using a conventional center of mass relationship,

$$\vec{r}^{\frac{Ccm}{G}} = \frac{1}{m^t}\left(m^a\vec{r}^{\frac{Acm}{G}} + m^e\vec{r}^{\frac{Ecm}{G}} + \sum_1^n m^{Sn}\vec{r}^{\frac{Sn}{G}}\right). \tag{2.16}$$

Using this relationship, the acceleration of the composite center of mass can be expressed as

$$\begin{aligned} {}^i\vec{a}^{Ccm} &= \frac{{}^id^2}{dt^2}\left(\vec{r}^{\frac{G}{i_o}} + \frac{1}{m^t}\left(m^a\vec{r}^{\frac{Acm}{G}} + m^e\vec{r}^{\frac{Ecm}{G}} + \sum_1^n m^{Sn}\vec{r}^{\frac{Sn}{G}}\right)\right) \\ &= \frac{{}^id^2}{dt^2}\left(\vec{r}^{\frac{G}{i_o}}\right) + \frac{1}{m^t}\frac{{}^id^2}{dt^2}\left(m^a\vec{r}^{\frac{Acm}{G}} + m^e\vec{r}^{\frac{Ecm}{G}} + \sum_1^n m^{Sn}\vec{r}^{\frac{Sn}{G}}\right). \end{aligned} \tag{2.17}$$

Multiplying both sides of the above equation by m^t and a new expression for $\vec{F}^{sys}$ is returned,

$$\vec{F}^{sys} = m^t{}^i\vec{a}^{Ccm} = m^t\frac{{}^id^2}{dt^2}\left(\vec{r}^{\frac{G}{i_o}}\right) + \frac{{}^id^2}{dt^2}\left(m^a\vec{r}^{\frac{Acm}{G}} + m^e\vec{r}^{\frac{Ecm}{G}} + \sum_1^n m^{Sn}\vec{r}^{\frac{Sn}{G}}\right). \tag{2.18}$$

The total mass relationship found in Section 0 above can be used to split up the $m^t\frac{{}^id^2}{dt^2}\left(\vec{r}^{\frac{G}{i_o}}\right)$ term into

$$m^t\frac{{}^id^2}{dt^2}\left(\vec{r}^{\frac{G}{i_o}}\right) = m^a\frac{{}^id^2}{dt^2}\left(\vec{r}^{\frac{G}{i_o}}\right) + m^e\frac{{}^id^2}{dt^2}\left(\vec{r}^{\frac{G}{i_o}}\right) + \sum_1^n m^{Sn}\frac{{}^id^2}{dt^2}\left(\vec{r}^{\frac{G}{i_o}}\right). \tag{2.19}$$

This makes it possible to combine the second derivative terms of Equation (2.18), which simplifies the expression because then each part of the system can be examined individually,

$$m^t {}^i\vec{a}^{Ccm} = m^a \frac{{}^i d^2}{dt^2}\left(\vec{r}^{\frac{G}{i_o}} + \vec{r}^{\frac{Acm}{G}}\right) + m^e \frac{{}^i d^2}{dt^2}\left(\vec{r}^{\frac{G}{i_o}} + \vec{r}^{\frac{Ecm}{G}}\right)$$

$$+ \sum_1^n m^{S_n} \frac{{}^i d^2}{dt^2}\left(\vec{r}^{\frac{G}{i_o}} + \vec{r}^{\frac{Sn}{G}}\right) \tag{2.20}$$

$$= m^a {}^i\vec{a}^{Acm} + m^e {}^i\vec{a}^{Ecm} + \sum_1^n m^{S_n} {}^i\vec{a}^{S_n}.$$

The next step is to derive expressions for the translational acceleration of each vehicle part: ${}^i\vec{a}^{Acm}$, ${}^i\vec{a}^{Ecm}$, and ${}^i\vec{a}^{S_n}$.

2.3.1 Translational Acceleration of Airframe Center of Mass

This section develops the translational acceleration of the airframe center of mass, with respect to the inertial frame, ${}^i\vec{a}^{Acm}$, as

$${}^i\vec{a}^{Acm} = \frac{{}^i d^2}{dt^2}\left(\vec{r}^{\frac{Acm}{i_o}}\right). \tag{2.21}$$

The location from the inertial origin to the airframe center of mass (dry mass, without propellant), $\vec{r}^{\frac{Acm}{i_o}}$, can be expressed as a sum of the vector from the inertial origin to the gimbal, and the vector from the gimbal to the airframe center of mass. That is,

$$\vec{r}^{\frac{Acm}{i_o}} = \vec{r}^{\frac{G}{i_o}} + \vec{r}^{\frac{Acm}{G}}. \tag{2.22}$$

This relationship is used because it simplifies the derivation, as the vector from the gimbal to the airframe center of mass is fixed in the body frame, so its derivative in the body frame with respect to time will be zero. The gimbal point is chosen to be the origin of the body frame because it should be a simple task for designers to reference other parts of the vehicle to the location of the gimbal.

Next, the derivative of Equation (2.22) is taken with respect to the inertial frame to find the velocity of the airframe with respect to the inertial frame,

$$\begin{aligned} {}^{i}\vec{v}^{Acm} &= \frac{{}^{i}d}{dt}\left(\vec{r}^{\,\frac{Acm}{i_0}}\right) \\ &= {}^{i}\vec{v}^{G} + \frac{{}^{i}d}{dt}\left(\vec{r}^{\,\frac{Acm}{G}}\right). \end{aligned} \tag{2.23}$$

This expression can be further simplified using the vector differentiation relationship of Equation (2.3). This relationship is used to simplify the expression, as the the derivative of $\vec{r}^{\,\frac{Acm}{G}}$ in the body frame is zero. Specifically,

$$\begin{aligned} {}^{i}\vec{v}^{Acm} &= {}^{i}\vec{v}^{G} + \frac{{}^{b}d}{dt}\left(\vec{r}^{\,\frac{Acm}{G}}\right) + {}^{i}\vec{\omega}^{b} \times \vec{r}^{\,\frac{Acm}{G}} \\ &= {}^{i}\vec{v}^{G} + {}^{i}\vec{\omega}^{b} \times \vec{r}^{\,\frac{Acm}{G}}. \end{aligned} \tag{2.24}$$

Another derivative is taken using the chain rule to find the acceleration of the airframe center of mass,

$$\begin{aligned} {}^{i}\vec{a}^{Acm} &= \frac{{}^{i}d}{dt}\left({}^{i}\vec{v}^{Acm}\right) \\ &= {}^{i}\vec{a}^{G} + \frac{{}^{i}d}{dt}\left({}^{i}\vec{\omega}^{b}\right) \times \vec{r}^{\,\frac{Acm}{G}} + {}^{i}\vec{\omega}^{b} \times \frac{{}^{i}d}{dt}\left(\vec{r}^{\,\frac{Acm}{G}}\right) \\ &= {}^{i}\vec{a}^{G} + {}^{i}\vec{\alpha}^{b} \times \vec{r}^{\,\frac{Acm}{G}} + {}^{i}\vec{\omega}^{b} \times \left({}^{i}\vec{\omega}^{b} \times \vec{r}^{\,\frac{Acm}{G}}\right). \end{aligned} \tag{2.25}$$

This is the translational acceleration of the airframe center of mass, in the inertial frame, expressed in terms of the translational acceleration of the gimbal in the inertial frame and the angular rotations of the body frame with respect to the inertial frame.

2.3.2 Translational Acceleration of Engine Center of Mass

This section develops the translational acceleration of the engine center of mass, with respect to the inertial frame, ${}^{i}\vec{a}^{Ecm}$. To do this, the same procedure is used as that for the airframe center of mass. That is,

$$ {}^{i}\vec{a}^{Ecm} = \frac{{}^{i}d^2}{dt^2}\left(\vec{r}^{\,\frac{Ecm}{i_o}}\right). \tag{2.26} $$

The location from the inertial origin to the engine center of mass is expressed as

$$ \vec{r}^{\,\frac{Ecm}{i_o}} = \vec{r}^{\,\frac{G}{i_o}} + \vec{r}^{\,\frac{Ecm}{G}}, \tag{2.27} $$

the sum of the vector from the inertial origin to the gimbal, and the vector from the gimbal to the engine center of mass. As in Section 2.3.1, the gimbal point is used so that the translational acceleration of the engine equation can be expressed in terms of the translational acceleration of the gimbal. As before, this is technically not necessary, but is done for convenience.

The derivative of $\vec{r}^{\,\frac{Ecm}{i_o}}$ with respect to the inertial frame is taken to find the velocity of the engine center of mass,

$$ \begin{aligned} {}^{i}\vec{v}^{Ecm} &= \frac{{}^{i}d}{dt}\left(\vec{r}^{\,\frac{Ecm}{i_o}}\right) \\ &= {}^{i}\vec{v}^{G} + \frac{{}^{i}d}{dt}\left(\vec{r}^{\,\frac{Ecm}{G}}\right) \end{aligned} \tag{2.28} $$

The vector differentiation relationship of Equation (2.3) is again used so that the derivative of $\vec{r}^{\,\frac{Ecm}{G}}$ can be taken in the engine frame. This simplifies the equation because $\vec{r}^{\,\frac{Ecm}{G}}$ is constant in the engine frame and therefore its derivative in the engine frame with respect to time is zero. That is,

$$
\begin{aligned}
{}^{i}\vec{v}^{Ecm} &= {}^{i}\vec{v}^{G} + \frac{{}^{e}d}{dt}\left(\vec{r}^{\frac{Ecm}{G}}\right) + {}^{i}\vec{\omega}^{e} \times \vec{r}^{\frac{Ecm}{G}} \\
&= {}^{i}\vec{v}^{G} + {}^{i}\vec{\omega}^{e} \times \vec{r}^{\frac{Ecm}{G}}.
\end{aligned}
\tag{2.29}
$$

Another derivative is taken to find the acceleration of the engine center of mass with respect to the inertial frame,

$$
\begin{aligned}
{}^{i}\vec{a}^{Ecm} &= \frac{{}^{i}d}{dt}\left({}^{i}\vec{v}^{Ecm}\right) \\
&= {}^{i}\vec{a}^{G} + \frac{{}^{i}d}{dt}\left({}^{i}\vec{\omega}^{e}\right) \times \vec{r}^{\frac{Ecm}{G}} + {}^{i}\vec{\omega}^{e} \times \frac{{}^{i}d}{dt}\left(\vec{r}^{\frac{Ecm}{G}}\right) \\
&= {}^{i}\vec{a}^{G} + {}^{i}\vec{\alpha}^{e} \times \vec{r}^{\frac{Ecm}{G}} + {}^{i}\vec{\omega}^{e} \times \left({}^{i}\vec{\omega}^{e} \times \vec{r}^{\frac{Ecm}{G}}\right).
\end{aligned}
\tag{2.30}
$$

2.3.3 Translational Acceleration of nth Slosh Mass Center of Mass

The approach of Sections 2.3.1 and 2.3.2 is used to find the translational acceleration of each slosh mass, ${}^{i}\vec{a}^{Sn}$, but first it is necessary to determine how propellant slosh is mathematically modeled within this derivation.

Because the propellant tanks of this system are spherical, it is reasonable to assume that all of a given propellant mass is sloshing. To model sloshing, the propellant mass is considered a point mass affixed to a spring mass damper. This is a common method for representing propellant slosh [20]. The spring and damper act as sloshing propellant would, using parameters from look-up tables or experimentation. Another method used to model slosh involves modeling the slosh center of mass as the mass of a pendulum [21]. It is noted that for this method, the slosh mass is not necessarily the entire propellant mass.

The location of a slosh CM from the gimbal point is split up into two vectors. These vectors are shown in Figure 2-2,

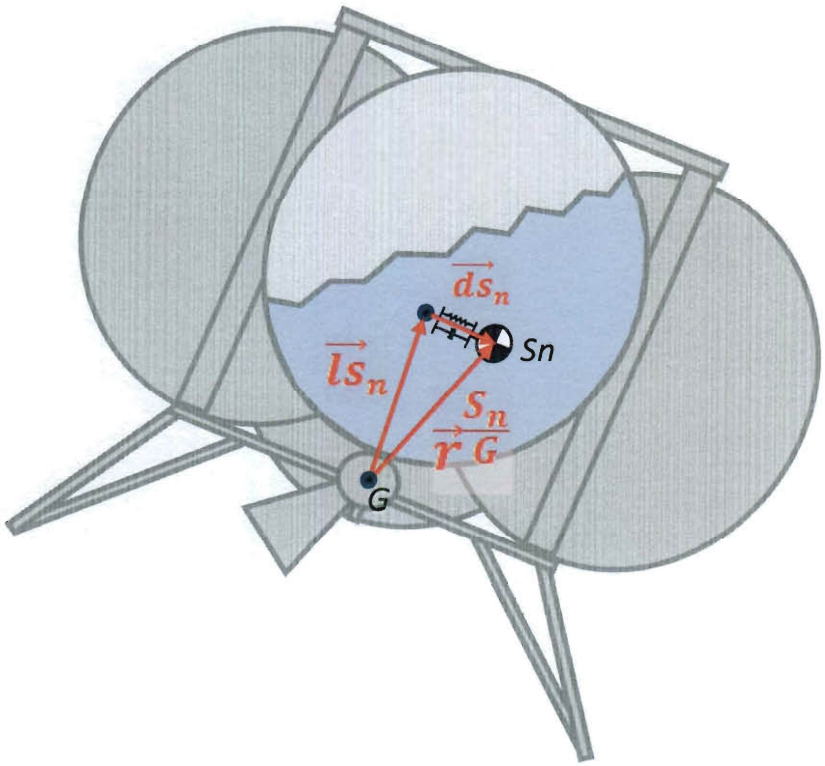

Figure 2-2: View of Slosh Vectors

where $\overrightarrow{ls_n}$ is the vector from the gimbal to the undisplaced 'equilibrium' position of the slosh mass. It is a mass parameter that is considered constant with respect to each time step. The vector from the undisplaced slosh location to its present location is denoted $\overrightarrow{ds_n}$. It is not considered as a quasi-static mass parameter, and therefore its time derivatives are not necessarily zero. The vector $\vec{r}^{\frac{S_n}{G}}$ is the sum of these two components, and is expressed by

$$\vec{r}^{\frac{S_n}{G}} = \overrightarrow{ls_n} + \overrightarrow{ds_n}. \tag{2.31}$$

The translational acceleration of each slosh mass is

$$ {}^{i}\vec{a}^{S_n} = \frac{{}^{i}d^2}{dt^2}\left(\vec{r}^{\frac{S_n}{i_o}}\right). \tag{2.32} $$

This is developed by first splitting the inertial slosh vector into an inertial to gimbal component and gimbal to mass component. This is done so that the acceleration of the slosh mass will be in terms of translation acceleration of the gimbal,

$$ \vec{r}^{\frac{S_n}{i_o}} = \vec{r}^{\frac{G}{i_o}} + \vec{r}^{\frac{S_n}{G}}. \tag{2.33} $$

As before, the derivative of this vector is simplified using the vector differentiation relationship. But this time there is a component of the body vector, $\overrightarrow{ds_n}$, that is not necessarily constant in the body frame,

$$ \begin{aligned} {}^{i}\vec{v}^{S_n} &= {}^{i}\vec{v}^{G} + \frac{{}^{i}d}{dt}\left(\vec{r}^{\frac{S_n}{G}}\right) \\ &= {}^{i}\vec{v}^{G} + \frac{{}^{b}d}{dt}\left(\overrightarrow{ls_n} + \overrightarrow{ds_n}\right) + {}^{i}\vec{\omega}^{b} \times \left(\overrightarrow{ls_n} + \overrightarrow{ds_n}\right) \\ &= {}^{i}\vec{v}^{G} + \left(\dot{ds_{n_y}} b\hat{y} + \dot{ds_{n_z}} b\hat{z}\right) + {}^{i}\vec{\omega}^{b} \times \left(\overrightarrow{ls_n} + \overrightarrow{ds_n}\right). \end{aligned} \tag{2.34} $$

Because of the quasi-static assumption of this derivation, $\overrightarrow{ls_n}$ is constant in the body frame, so its derivative is zero.

The translational acceleration of the nth slosh mass is given by

$$
\begin{aligned}
{}^{i}\vec{a}^{S_n} = {}^{i}\vec{a}^{G} + \left(d\ddot{s}_{n_y} b\hat{y} + d\ddot{s}_{n_z} b\hat{z}\right) + {}^{i}\vec{\omega}^{b} \times \left(d\dot{s}_{n} b\hat{y} + d\dot{s}_{n} b\hat{z}\right) \\
+ {}^{i}\vec{a}^{b} \times \left(\overrightarrow{ls_n} + \overrightarrow{ds_n}\right) + {}^{i}\vec{\omega}^{b} \times \left(\left(d\dot{s}_{n_y} b\hat{y} + d\dot{s}_{n_z} b\hat{z}\right)\right. \\
\left. + {}^{i}\vec{\omega}^{b} \times \left(\overrightarrow{ls_n} + \overrightarrow{ds_n}\right)\right) \qquad (2.35) \\
= {}^{i}\vec{a}^{G} + \left(d\ddot{s}_{n_y} b\hat{y} + d\ddot{s}_{n_z} b\hat{z}\right) + {}^{i}\vec{a}^{b} \times \left(\overrightarrow{ls_n} + \overrightarrow{ds_n}\right) + {}^{i}\vec{\omega}^{b} \\
\times \left(2\left(d\dot{s}_{n_y} b\hat{y} + d\dot{s}_{n_z} b\hat{z}\right) + {}^{i}\vec{\omega}^{b} \times \left(\overrightarrow{ls_n} + \overrightarrow{ds_n}\right)\right).
\end{aligned}
$$

2.3.4 Translational Acceleration of Gimbal

It is now possible to solve for the translational acceleration of the gimbal. The acceleration of the gimbal is chosen as a point of interest because it is fixed in both the body and engine frames, greatly simplifying the equations of motion. Furthermore, it is a straightforward task to define other points of the system with respect to the gimbal. Returning to the equation for the sum of external forces on the system (Equation (2.18)), the general form of this equation is

$$
\vec{F}^{sys} = \sum_{part} m^{part\,i}\vec{a}^{part}, \qquad (2.36)
$$

where *part* is every individual component associated with the vehicle, including propellant. For this problem the parts are the airframe, engine, and slosh masses,

$$
\vec{F}^{sys} = m^{a}\left({}^{i}\vec{a}^{Acm}\right) + m^{e}\left({}^{i}\vec{a}^{Ecm}\right) + \sum_{1}^{n}\left[m^{sn}\left({}^{i}\vec{a}^{S_n}\right)\right] \qquad (2.37)
$$

Plugging in the expressions for ${}^{i}\vec{a}^{Acm}$, ${}^{i}\vec{a}^{Ecm}$, and ${}^{i}\vec{a}^{S_n}$ derived in Section 2.3, the equation of the sum of external forces becomes. Thus,

$$\begin{aligned}\vec{F}^{sys} = m^a\left({}^i\vec{a}^G + {}^i\vec{\alpha}^b \times \vec{r}^{\frac{Acm}{G}} + {}^i\vec{\omega}^b \times \left({}^i\vec{\omega}^b \times \vec{r}^{\frac{Acm}{G}}\right)\right)\\ + m^e\left({}^i\vec{a}^G + {}^i\vec{\alpha}^e \times \vec{r}^{\frac{Ecm}{G}} + {}^i\vec{\omega}^e \times \left({}^i\vec{\omega}^e \times \vec{r}^{\frac{Ecm}{G}}\right)\right)\\ + \sum_1^n\Big[m^{sn}\left({}^i\vec{a}^G + \left(\ddot{ds}_n b\hat{y} + \ddot{ds}_n b\hat{z}\right) + {}^i\vec{\alpha}^b \times \left(\overrightarrow{ls_n} + \overrightarrow{ds_n}\right)\right.\\ \left. + {}^i\vec{\omega}^b \times \left(2\left(\dot{ds}_n b\hat{y} + \dot{ds}_n b\hat{z}\right) + {}^i\vec{\omega}^b \times \left(\overrightarrow{ls_n} + \overrightarrow{ds_n}\right)\right)\right)\Big].\end{aligned} \tag{2.38}$$

Keeping in mind that the purpose of the derivation is to develop a set of EOM's that can be placed in matrix format for ease of computer coding, it is necessary to rearrange the above equation so that all state accelerations and their coefficients are on the left side of the equation,

$$\begin{aligned}m^t\,{}^i\vec{a}^G + \left(m^a\left(-\vec{r}^{\frac{Acm}{G}}\right)^x + m^e\left(-\vec{r}^{\frac{Ecm}{G}}\right)^x + \sum_1^n m^{sn}\left(-\vec{r}^{\frac{Sn}{G}}\right)^x\right){}^i\vec{\alpha}^b\\ + m^e\left(-\vec{r}^{\frac{Ecm}{G}}\right)^x {}^b\vec{\alpha}^e + \sum_1^n m^{Sn}\ddot{\vec{d}}_{sn}\\ = \vec{F}^{sys} - m^a\left({}^i\vec{\omega}^b \times \left({}^i\vec{\omega}^b \times \vec{r}^{\frac{Acm}{G}}\right)\right)\\ - m^e\left(\left({}^i\vec{\omega}^b \times {}^b\vec{\omega}^e\right) \times \vec{r}^{\frac{Ecm}{G}} + {}^i\vec{\omega}^e \times \left({}^i\vec{\omega}^e \times \vec{r}^{\frac{Ecm}{G}}\right)\right)\\ - \sum_1^n\left[m^{sn}\left({}^i\vec{\omega}^b \times \left(2\dot{\vec{d}}_{sn} + {}^i\vec{\omega}^b \times \vec{r}^{\frac{Sn}{G}}\right)\right)\right],\end{aligned} \tag{2.39}$$

where $\vec{F}^{sys}$ is the sum of external forces acting on the body frame,

$$\vec{F}^{sys} = \vec{F}_{thrust} + \vec{F}_{drag} + m^t\vec{g}_{body} \tag{2.40}$$

2.4 Angular Acceleration of Body

This section develops the equation of motion for the angular acceleration of the vehicle body. The vehicle is a system of two rigid bodies: the airframe (vehicle body), and engine. Starting with the sum of the moments on the entire system, it is possible to solve for the angular acceleration of the vehicle body. The sum of moments on a system of rigid bodies is given by [22],

$$\vec{M}^{\overline{sys}}_{\overline{Ccm}} = \dot{\vec{H}}_A + \sum_i^n \dot{\vec{H}}_\iota + \sum_i^n \vec{\rho}_i \times m^i \ddot{\vec{\rho}}_\iota - \vec{\rho}_c \times m^t \ddot{\vec{\rho}}_c \,, \tag{2.41}$$

where $\dot{\vec{H}}_A$ Denotes the derivative of angular momentum of the main body (in this case the airframe) about its center of mass. In the notation previously established, this term is

$$\dot{\vec{H}}_A = \underline{I}^{\overline{A}}_{\overline{Acm}} \cdot {}^i\vec{\alpha}^b + {}^i\vec{\omega}^b \times \left(\underline{I}^{\overline{A}}_{\overline{Acm}} \cdot {}^i\vec{\omega}^b\right). \tag{2.42}$$

The symbol $\dot{\vec{H}}_\iota$ denotes the derivative of angular momentum of each additional body about its center of mass. For this derivation, the slosh masses are assumed to be point masses, and therefore have no inertia about their center of masses. Therefore the only additional body is the engine,

$$\sum_i^n \dot{\vec{H}}_\iota = \underline{I}^{\overline{E}}_{\overline{Ecm}} \cdot {}^i\vec{\alpha}^e + {}^i\vec{\omega}^e \times \left(\underline{I}^{\overline{E}}_{\overline{Ecm}} \cdot {}^i\vec{\omega}^e\right). \tag{2.43}$$

The symbol $\vec{\rho}_i$ denotes the vector from the main body CM to each respective CM of additional bodies, and $\vec{\rho}_c$ denotes the vector from the main body CM to the composite center of mass.

Using notation in accordance with the rest of this derivation, Equation (2.41) becomes

$$\begin{aligned}\vec{M}^{\frac{sys}{Ccm}} = \underline{I}^{\frac{A}{Acm}} \cdot {}^{i}\vec{\alpha}^{b} + {}^{i}\vec{\omega}^{b} \times \left(\underline{I}^{\frac{A}{Acm}} \cdot {}^{i}\vec{\omega}^{b}\right) + \underline{I}^{\frac{E}{Ecm}} \cdot {}^{i}\vec{\alpha}^{e} \\ + {}^{i}\vec{\omega}^{e} \times \left(\underline{I}^{\frac{E}{Ecm}} \cdot {}^{i}\vec{\omega}^{e}\right) + \vec{r}^{\frac{Ecm}{Acm}} \times m^{e}\left(\frac{{}^{i}d^{2}}{dt^{2}}\left(\vec{r}^{\frac{Ecm}{Acm}}\right)\right) \\ + \sum_{1}^{n} \vec{r}^{\frac{Sn}{Acm}} \times m^{Sn}\left(\frac{{}^{i}d^{2}}{dt^{2}}\left(\vec{r}^{\frac{Sn}{Acm}}\right)\right) \\ - \vec{r}^{\frac{Ccm}{Acm}} \times m^{t}\left(\frac{{}^{i}d^{2}}{dt^{2}}\left(\vec{r}^{\frac{Ccm}{Acm}}\right)\right).\end{aligned} \tag{2.44}$$

This equation can now be rearranged to be useful as an EOM for the angular acceleration of the vehicle. First it is necessary to expand the derivatives shown in the equation above. Using derivatives from Sections 2.3 is can be seen that the first derivative of $\vec{r}^{\frac{Ecm}{Acm}}$ is

$$\begin{aligned}\frac{{}^{i}d}{dt}\left(\vec{r}^{\frac{Ecm}{Acm}}\right) &= \frac{{}^{i}d}{dt}\left(\vec{r}^{\frac{Ecm}{G}} - \vec{r}^{\frac{Acm}{G}}\right) \\ &= {}^{i}\vec{\omega}^{e} \times \vec{r}^{\frac{Ecm}{G}} - {}^{i}\vec{\omega}^{b} \times \vec{r}^{\frac{Acm}{G}}.\end{aligned} \tag{2.45}$$

And the first derivative of $\vec{r}^{\frac{Sn}{Acm}}$ is

$$\begin{aligned}\frac{{}^{i}d}{dt}\left(\vec{r}^{\frac{Sn}{Acm}}\right) &= \frac{{}^{i}d}{dt}\left(\vec{r}^{\frac{Sn}{G}} - \vec{r}^{\frac{Acm}{G}}\right) \\ &= \dot{\vec{d}}_{Sn} + {}^{i}\vec{\omega}^{b} \times \vec{r}^{\frac{Sn}{G}} - {}^{i}\vec{\omega}^{b} \times \vec{r}^{\frac{Acm}{G}}.\end{aligned} \tag{2.46}$$

Therefore the second derivative of $\vec{r}^{\frac{Ecm}{Acm}}$ becomes

$$\begin{aligned}\frac{{}^{i}d^{2}}{dt^{2}}\left(\vec{r}^{\frac{Ecm}{Acm}}\right) = {}^{i}\vec{\alpha}^{e} \times \vec{r}^{\frac{Ecm}{G}} + {}^{i}\vec{\omega}^{e} \times \left({}^{i}\vec{\omega}^{e} \times \vec{r}^{\frac{Ecm}{G}}\right) \\ - {}^{i}\vec{\alpha}^{b} \times \vec{r}^{\frac{Acm}{G}} - {}^{i}\vec{\omega}^{b} \times \left({}^{i}\vec{\omega}^{b} \times \vec{r}^{\frac{Acm}{G}}\right).\end{aligned} \tag{2.47}$$

And the second derivative of $\vec{r}^{\,\frac{Sn}{Acm}}$ becomes

$$\frac{{}^{i}d^2}{dt^2}\left(\vec{r}^{\,\frac{Sn}{Acm}}\right) = \ddot{\vec{d}}_{Sn} + {}^{i}\vec{\alpha}^{b} \times \vec{r}^{\,\frac{Sn}{G}} + {}^{i}\vec{\omega}^{b} \times \left(2\dot{\vec{d}}_{Sn} + {}^{i}\vec{\omega}^{b} \times \vec{r}^{\,\frac{Sn}{G}}\right) - {}^{i}\vec{\alpha}^{b} \times \vec{r}^{\,\frac{Acm}{G}} - {}^{i}\vec{\omega}^{b} \times \left({}^{i}\vec{\omega}^{b} \times \vec{r}^{\,\frac{Acm}{G}}\right). \tag{2.48}$$

The second derivative of $\vec{r}^{\,\frac{Ccm}{Acm}}$ is (using the previously established definition for $\vec{r}^{\,\frac{Ccm}{G}}$)

$$\begin{aligned}
\frac{{}^{i}d^2}{dt^2}\left(\vec{r}^{\,\frac{Ccm}{Acm}}\right) &= \frac{{}^{i}d^2}{dt^2}\left(\vec{r}^{\,\frac{Ccm}{G}} - \vec{r}^{\,\frac{Acm}{G}}\right) \\
&= \frac{{}^{i}d^2}{dt^2}\left(\frac{m^a}{m^t}\vec{r}^{\,\frac{Acm}{G}} + \frac{m^e}{m^t}\vec{r}^{\,\frac{Ecm}{G}} + \sum_n \frac{m^{Sn}}{m^t}\vec{r}^{\,\frac{Sn}{G}} - \vec{r}^{\,\frac{Acm}{G}}\right) \\
&= \frac{m^a}{m^t}\left({}^{i}\vec{\alpha}^{b} \times \vec{r}^{\,\frac{Acm}{G}} + {}^{i}\vec{\omega}^{b} \times \left({}^{i}\vec{\omega}^{b} \times \vec{r}^{\,\frac{Acm}{G}}\right)\right) \\
&\quad + \frac{m^e}{m^t}\left({}^{i}\vec{\alpha}^{e} \times \vec{r}^{\,\frac{Ecm}{G}} + {}^{i}\vec{\omega}^{e} \times \left({}^{i}\vec{\omega}^{e} \times \vec{r}^{\,\frac{Ecm}{G}}\right)\right) \\
&\quad + \sum_n \frac{m^{Sn}}{m^t}\left(\ddot{\vec{d}}_{sn} + {}^{i}\vec{\alpha}^{b} \times \vec{r}^{\,\frac{Sn}{G}} + {}^{i}\vec{\omega}^{b} \times \left(2\dot{\vec{d}}_{sn} + {}^{i}\vec{\omega}^{b} \times \vec{r}^{\,\frac{Sn}{G}}\right)\right) \\
&\quad - \left({}^{i}\vec{\alpha}^{b} \times \vec{r}^{\,\frac{Acm}{G}} + {}^{i}\vec{\omega}^{b} \times \left({}^{i}\vec{\omega}^{b} \times \vec{r}^{\,\frac{Acm}{G}}\right)\right).
\end{aligned} \tag{2.49}$$

Using Equations (2.47)-(2.49), the equations for $\vec{M}^{\frac{sys}{Ccm}}$ becomes

$$\vec{M}^{\overrightarrow{sys}}_{\overrightarrow{Ccm}} = \underline{I}^{\overrightarrow{A}}_{\overrightarrow{Acm}} \cdot {}^{i}\vec{\alpha}^{b} + {}^{i}\vec{\omega}^{b} \times \left(\underline{I}^{\overrightarrow{A}}_{\overrightarrow{Acm}} \cdot {}^{i}\vec{\omega}^{b}\right) + \underline{I}^{\overrightarrow{E}}_{\overrightarrow{Ecm}} \cdot {}^{i}\vec{\alpha}^{e}$$

$$+ {}^{i}\vec{\omega}^{e} \times \left(\underline{I}^{\overrightarrow{E}}_{\overrightarrow{Ecm}} \cdot {}^{i}\vec{\omega}^{e}\right)$$

$$+ \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{Acm}} \times m^{e}\left({}^{i}\vec{\alpha}^{e} \times \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{G}} + {}^{i}\vec{\omega}^{e} \times \left({}^{i}\vec{\omega}^{e} \times \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{G}}\right)\right)$$

$$\underline{-\vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{Acm}} \times m^{e}\left({}^{i}\vec{\alpha}^{b} \times \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{G}} - {}^{i}\vec{\omega}^{b} \times \left({}^{i}\vec{\omega}^{b} \times \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{G}}\right)\right)}$$

$$+ \sum_{1}^{n}\left[\vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{Acm}} \times m^{Sn}\left(\ddot{\vec{d}}_{Sn} + {}^{i}\vec{\alpha}^{b} \times \vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{G}}\right.\right.$$

$$\left. + {}^{i}\vec{\omega}^{b} \times \left(2\dot{\vec{d}}_{Sn} + {}^{i}\vec{\omega}^{b} \times \vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{G}}\right)\right) \qquad (2.50)$$

$$\left.\underline{-\vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{Acm}} \times m^{Sn}\left({}^{i}\vec{\alpha}^{b} \times \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{G}} - {}^{i}\vec{\omega}^{b} \times \left({}^{i}\vec{\omega}^{b} \times \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{G}}\right)\right)}\right]$$

$$- \vec{r}^{\overrightarrow{Ccm}}_{\overrightarrow{Acm}} \times m^{t}\left(\frac{{}^{i}d^{2}}{dt^{2}}\left(\vec{r}^{\overrightarrow{Ccm}}_{\overrightarrow{G}}\right)\right)$$

$$\underline{+\vec{r}^{\overrightarrow{Ccm}}_{\overrightarrow{Acm}} \times m^{t}\left({}^{i}\vec{\alpha}^{b} \times \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{G}} + {}^{i}\vec{\omega}^{b} \times \left({}^{i}\vec{\omega}^{b} \times \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{G}}\right)\right)}.$$

The underlined portions of Equation (2.50) can be reduced to

$$\left(m^{t}\vec{r}^{\overrightarrow{Ccm}}_{\overrightarrow{Acm}} - m^{e}\vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{Acm}} - \sum_{n} m^{Sn}\vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{Acm}}\right)$$

$$\times \left({}^{i}\vec{\alpha}^{b} \times \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{G}} + {}^{i}\vec{\omega}^{b} \times \left({}^{i}\vec{\omega}^{b} \times \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{G}}\right)\right) \qquad (2.51)$$

It can be shown that this component equals zero by using an alternative, but equivalent, expression for finding the position of the composite center of mass (this time defined from the airframe center of mass) from Equation (8-49) of [22],

$$\vec{r}_{\overline{Acm}}^{\,\overline{Ccm}} = \frac{1}{m^t}\left(m^e \vec{r}_{\overline{Acm}}^{\,\overline{Ecm}} + \sum_n m^{Sn}\, \vec{r}_{\overline{Acm}}^{\,\overline{Sn}}\right). \tag{2.52}$$

Therefore, the equation for $\vec{M}^{\overline{sys}}_{\overline{Ccm}}$ reduces to

$$\begin{aligned}
\vec{M}^{\overline{sys}}_{\overline{Ccm}} &= \underline{I}^{\overline{A}}_{\overline{Acm}} \cdot {}^{i}\vec{\alpha}^{b} + {}^{i}\vec{\omega}^{b} \times \left(\underline{I}^{\overline{A}}_{\overline{Acm}} \cdot {}^{i}\vec{\omega}^{b}\right) + \underline{I}^{\overline{E}}_{\overline{Ecm}} \cdot {}^{i}\vec{\alpha}^{e} \\
&+ {}^{i}\vec{\omega}^{e} \times \left(\underline{I}^{\overline{E}}_{\overline{Ecm}} \cdot {}^{i}\vec{\omega}^{e}\right) \\
&+ \left(\vec{r}_{\overline{Ccm}}^{\,\overline{Ecm}} + \vec{r}_{\overline{Acm}}^{\,\overline{Ccm}}\right) \times m^{e}\left({}^{i}\vec{\alpha}^{e} \times \vec{r}^{\,\overline{Ecm}}_{\overline{G}} + {}^{i}\vec{\omega}^{e} \times \left({}^{i}\vec{\omega}^{e} \times \vec{r}^{\,\overline{Ecm}}_{\overline{G}}\right)\right) \\
&+ \sum_{1}^{n}\left[\left(\vec{r}_{\overline{Ccm}}^{\,\overline{Sn}} + \vec{r}_{\overline{Acm}}^{\,\overline{Ccm}}\right) \times m^{Sn}\left(\ddot{\vec{d}}_{Sn} + {}^{i}\vec{\alpha}^{b} \times \vec{r}^{\,\overline{Sn}}_{\overline{G}}\right.\right. \\
&\qquad \left.\left. + {}^{i}\vec{\omega}^{b} \times \left(2\dot{\vec{d}}_{Sn} + {}^{i}\vec{\omega}^{b} \times \vec{r}^{\,\overline{Sn}}_{\overline{G}}\right)\right)\right] \\
&- \vec{r}_{\overline{Acm}}^{\,\overline{Ccm}} \times \left[m^{a}\left({}^{i}\vec{\alpha}^{b} \times \vec{r}^{\,\overline{Acm}}_{\overline{G}} + {}^{i}\vec{\omega}^{b} \times \left({}^{i}\vec{\omega}^{b} \times \vec{r}^{\,\overline{Acm}}_{\overline{G}}\right)\right)\right. \\
&\qquad + m^{e}\left({}^{i}\vec{\alpha}^{e} \times \vec{r}^{\,\overline{Ecm}}_{\overline{G}} + {}^{i}\vec{\omega}^{e} \times \left({}^{i}\vec{\omega}^{e} \times \vec{r}^{\,\overline{Ecm}}_{\overline{G}}\right)\right) \\
&\qquad + m^{Sn}\left(\ddot{\vec{d}}_{Sn} + {}^{i}\vec{\alpha}^{b} \times \vec{r}^{\,\overline{Sn}}_{\overline{G}}\right. \\
&\qquad\qquad \left.\left. + {}^{i}\vec{\omega}^{b} \times \left(2\dot{\vec{d}}_{Sn} + {}^{i}\vec{\omega}^{b} \times \vec{r}^{\,\overline{Sn}}_{\overline{G}}\right)\right)\right]
\end{aligned} \tag{2.53}$$

Dropping terms that add to zero yields

$$
\begin{aligned}
\vec{M}^{\overrightarrow{sys}}_{\overrightarrow{Ccm}} &= \underline{I}^{\overrightarrow{A}}_{\overrightarrow{Acm}} \cdot {}^{i}\vec{\alpha}^{b} + {}^{i}\vec{\omega}^{b} \times \left(\underline{I}^{\overrightarrow{A}}_{\overrightarrow{Acm}} \cdot {}^{i}\vec{\omega}^{b} \right) \\
&\quad + \underline{I}^{\overrightarrow{E}}_{\overrightarrow{Ecm}} \cdot {}^{i}\vec{\alpha}^{e} + {}^{i}\vec{\omega}^{e} \times \left(\underline{I}^{\overrightarrow{E}}_{\overrightarrow{Ecm}} \cdot {}^{i}\vec{\omega}^{e} \right) \\
&+ \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{Ccm}} \times m^{e} \left({}^{i}\vec{\alpha}^{e} \times \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{G}} + {}^{i}\vec{\omega}^{e} \times \left({}^{i}\vec{\omega}^{e} \times \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{G}} \right) \right) \\
&+ \sum_{1}^{n} \left[\vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{Ccm}} \times m^{Sn} \left(\ddot{\vec{d}}_{Sn} + {}^{i}\vec{\alpha}^{b} \times \vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{G}} \right.\right. \\
&\quad \left.\left. + {}^{i}\vec{\omega}^{b} \times \left(2\dot{\vec{d}}_{Sn} + {}^{i}\vec{\omega}^{b} \times \vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{G}} \right) \right) \right] \\
&+ \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{Ccm}} \times m^{a} \left({}^{i}\vec{\alpha}^{b} \times \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{G}} + {}^{i}\vec{\omega}^{b} \times \left({}^{i}\vec{\omega}^{b} \times \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{G}} \right) \right).
\end{aligned}
\tag{2.54}
$$

This is the equation for the sum of external moments on the system about the composite center of mass of the vehicle. In order to place this equation in a matrix equation containing all the state accelerations, it is necessary to rearrange it to have state accelerations and their coefficients on the left hand side. That is,

$$\left(\underline{\underline{I}}^{\underline{\underline{A}}}_{\overrightarrow{Acm}} + \underline{\underline{I}}^{\underline{\underline{E}}}_{\overrightarrow{Ecm}} + m^a \vec{r}^{\overrightarrow{Acm}^x}_{\overrightarrow{Ccm}} \vec{r}^{\overrightarrow{G}^x}_{\overrightarrow{Acm}} + m^e \vec{r}^{\overrightarrow{Ecm}^x}_{\overrightarrow{Ccm}} \vec{r}^{\overrightarrow{G}^x}_{\overrightarrow{Ecm}} \right.$$

$$\left. + \sum_n m^{Sn} \vec{r}^{\overrightarrow{Sn}^x}_{\overrightarrow{Ccm}} \vec{r}^{\overrightarrow{G}^x}_{\overrightarrow{Sn}} \right) {}^i\vec{\alpha}^b$$

$$+ \left(\underline{\underline{I}}^{\underline{\underline{E}}}_{\overrightarrow{Ecm}} + m^e \vec{r}^{\overrightarrow{Ecm}^x}_{\overrightarrow{Ccm}} \vec{r}^{\overrightarrow{G}^x}_{\overrightarrow{Ecm}} \right) {}^b\vec{\alpha}^e + \sum_n \left(m^{Sn} \vec{r}^{\overrightarrow{Sn}^x}_{\overrightarrow{Ccm}} \right) \ddot{\vec{d}}_{Sn}$$

$$= \vec{M}^{\overrightarrow{sys}}_{\overrightarrow{Ccm}} - {}^i\vec{\omega}^b \times \left(\underline{\underline{I}}^{\underline{\underline{A}}}_{\overrightarrow{Acm}} \cdot {}^i\vec{\omega}^b \right) + \left(\underline{\underline{I}}^{\underline{\underline{E}}}_{\overrightarrow{Ecm}} \cdot {}^i\vec{\omega}^b \times {}^b\vec{\omega}^e \right)$$

$$- {}^i\vec{\omega}^e \times \left(\underline{\underline{I}}^{\underline{\underline{E}}}_{\overrightarrow{Ecm}} \cdot {}^i\vec{\omega}^e \right) \tag{2.55}$$

$$- \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{Ccm}} \times m^e \left({}^i\vec{\omega}^e \times \left({}^i\vec{\omega}^e \times \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{G}} \right) \right)$$

$$- \sum_1^n \left[\vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{Ccm}} \times m^{Sn} \left({}^i\vec{\omega}^b \times \left(2\dot{\vec{d}}_{Sn} + {}^i\vec{\omega}^b \times \vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{G}} \right) \right) \right]$$

$$- \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{Ccm}} \times m^a \left({}^i\vec{\omega}^b \times \left({}^i\vec{\omega}^b \times \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{G}} \right) \right)$$

$$- m^e \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{Ccm}} \times \left(\left({}^i\vec{\omega}^b \times {}^b\vec{\omega}^e \right) \times \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{G}} \right),$$

where $\vec{M}^{\overrightarrow{sys}}_{\overrightarrow{Ccm}}$ is the sum of external moments on the system about the composite center of mass,

$$\vec{M}^{\overrightarrow{sys}}_{\overrightarrow{Ccm}} = -\left(\vec{r}^{\overrightarrow{Ccm}}_{\overrightarrow{G}} \right) \times \vec{F}_{thrust}. \tag{2.56}$$

For this case the only external force that applies a moment of the system is thrust. Gravity does not induce a moment because it is applied at the composite center of mass.

2.5 Angular Acceleration of Engine

The engine is a modeled as a separate rigid body that rotates as a pendulum connected at the gimbal. It is modeled in such a way that it only rotates around the engine frame $e\hat{y}$ and $e\hat{z}$ directions. That is, the engine cannot roll around its centerline.

Equation (4-70) from [22] considers the angular momentum of a system of particles relative to an arbitrary point p,

$$\vec{M}_p - \vec{\rho}_c \times m\,\ddot{\vec{r}}_p = \dot{\vec{H}}_p\,. \tag{2.57}$$

For this derivation, p is the gimbal point so $\vec{M}_p$ refers to the sum of moments on the engine, about the gimbal, $\vec{M}^{E}_{G}$. The symbol $\vec{\rho}_c$ denotes the vector from the gimbal to the engine center of mass, $\vec{r}^{\,Ecm}_{\ G}$. The symbol m denotes the mass of the engine, m^e. The acceleration of point p, in this case the gimbal, is denoted by $\ddot{\vec{r}}_p$ and is equivalent to ${}^i\vec{a}^G$. Finally, $\dot{\vec{H}}_p$ is the derivative of angular momentum about point p (the gimbal), which in the notation of this derivation is equivalent to

$$\dot{\vec{H}}_p = \underline{I}^{E}_{G} \cdot {}^i\vec{\alpha}^e + {}^i\vec{\omega}^e \times \left(\underline{I}^{E}_{G} \cdot {}^i\vec{\omega}^e\right), \tag{2.58}$$

where $\underline{I}^{E}_{G}$ is the inertia dyadic of the engine about the gimbal, and can be calculated using the parallel axis theorem,

$$\begin{aligned}\underline{I}^{E}_{G} = \underline{I}^{E}_{Ecm} + m^e \Bigg[&\left(\vec{r}^{\,Ecm}_{\ G} \cdot b\hat{x}\right)^2 b\hat{x}b\hat{x} + \left(\vec{r}^{\,Ecm}_{\ G} \cdot b\hat{y}\right)^2 b\hat{y}b\hat{y} \\ &+ \left(\vec{r}^{\,Ecm}_{\ G} \cdot b\hat{z}\right)^2 b\hat{z}b\hat{z}\Bigg].\end{aligned} \tag{2.59}$$

Inserting relevant notation into Equation (2.57) it is possible to develop an equation for the sum of the moments on the engine around the gimbal (using the expansion for ${}^{i}\vec{\alpha}^{e}$ from Equation (2.10)),

$$\begin{aligned}\vec{M}_{G}^{E} &= \underline{I}_{G}^{E} \cdot {}^{i}\vec{\alpha}^{e} + {}^{i}\vec{\omega}^{e} \times \left(\underline{I}_{G}^{E} \cdot {}^{i}\vec{\omega}^{e}\right) + \vec{r}_{G}^{Ecm} \times \left(m^{e}\, {}^{i}\vec{a}^{G}\right) \\ &= \underline{I}_{G}^{E} \cdot {}^{b}\vec{\alpha}^{e} + \underline{I}_{G}^{E} \cdot \left({}^{i}\vec{\alpha}^{b} + {}^{i}\vec{\omega}^{b} \times {}^{b}\vec{\omega}^{e}\right) + {}^{i}\vec{\omega}^{e} \times \left(\underline{I}_{G}^{E} \cdot {}^{i}\vec{\omega}^{e}\right) \\ &\quad + \vec{r}_{G}^{Ecm} \times \left(m^{e}\, {}^{i}\vec{a}^{G}\right).\end{aligned} \tag{2.60}$$

This equation can be rearranged using cross product matrix notation, with the state accelerations on the left hand side,

$$\begin{aligned}& m^{e}\, \vec{r}_{Ecm}^{\,G\,x}\, {}^{i}\vec{a}^{G} + \left(\underline{I}_{G}^{E}\right) {}^{i}\vec{\alpha}^{b} + \left(\underline{I}_{G}^{E}\right) {}^{b}\vec{\alpha}^{e} \\ &= \vec{M}_{G}^{E} - \underline{I}_{G}^{E} \cdot \left({}^{i}\vec{\omega}^{b} \times {}^{b}\vec{\omega}^{e}\right) - {}^{i}\vec{\omega}^{e} \times \left(\underline{I}_{G}^{E} \cdot {}^{i}\vec{\omega}^{e}\right) \\ &\quad - \vec{r}_{Ecm}^{\,G} \times \left(m^{e}\left(\left({}^{i}\vec{\omega}^{b} \times {}^{b}\vec{\omega}^{e}\right) \times \vec{r}_{G}^{Ecm}\right)\right),\end{aligned} \tag{2.61}$$

where $\vec{M}_{G}^{E}$ is the sum of external moments on the engine about the gimbal. For this case there is a moment from gravity, and from an external torque, $\vec{\tau}_{\beta}$, which is the torque applied by the control system in order to swivel the nozzle to a desired rotation,

$$\vec{M}_{G}^{E} = \vec{r}_{G}^{Ecm} \times m^{e}\vec{g}_{body} + \vec{\tau}_{\beta}. \tag{2.62}$$

2.6 Acceleration of n^{th} Slosh Mass Displacement

So far, EOM's have been developed for three relevant state accelerations: translational acceleration of the gimbal, rotational acceleration of the body, and rotational acceleration of the engine. These EOM's are all coupled, and in the case of translational and rotational acceleration of the body, are also coupled to the translational acceleration of each slosh mass displacement. This section develops the EOM for this acceleration component.

To begin, the external forces on a given slosh mass are equal to the mass of that slosh times the acceleration of that slosh (for this case with respect to the inertial frame),

$$\vec{F}^{S_n} = m^{S_n}\left({}^i\vec{a}^{S_n}\right). \tag{2.63}$$

Using the definition for ${}^i\vec{a}^{S_n}$ developed in Section 2.3.3, this equation expands to

$$\begin{aligned}\vec{F}^{S_n} = m^{S_n}\Bigg({}^i\vec{a}^{G} + \left(\ddot{ds}_{n_y} b\hat{y} + \ddot{ds}_{n_z} b\hat{z}\right) + {}^i\vec{\alpha}^{b} \times \left(\vec{ls}_n + \vec{ds}_n\right) \\ + {}^i\vec{\omega}^{b} \times \left(2\left(\dot{ds}_{n_y} b\hat{y} + \dot{ds}_{n_z} b\hat{z}\right) + {}^i\vec{\omega}^{b} \times \left(\vec{ls}_n + \vec{ds}_n\right)\right)\Bigg).\end{aligned} \tag{2.64}$$

Next it is possible to solve for the acceleration of the n^{th} slosh mass in terms of the body frame, $\ddot{\vec{ds}}_n$,

$$\begin{aligned}\ddot{\vec{ds}}_n = \begin{bmatrix} 0 \\ \ddot{ds}_{n_y} \\ \ddot{ds}_{n_z} \end{bmatrix} = \frac{1}{m^{S_n}}\Bigg[\vec{F}^{S_n} - m^{S_n}\Bigg({}^i\vec{a}^{G} + {}^i\vec{\alpha}^{b} \times \vec{r}^{\frac{S_n}{G}} \\ + {}^i\vec{\omega}^{b} \times \left(2\left(\dot{ds}_{n_y} b\hat{y} + \dot{ds}_{n_z} b\hat{z}\right) + {}^i\vec{\omega}^{b} \times \vec{r}^{\frac{S_n}{G}}\right)\Bigg)\Bigg].\end{aligned} \tag{2.65}$$

Note that the propellant does not slosh in the $b\hat{x}$ direction. This is a physical constraint that is mathematically imposed by adding a constraining force detailed below. Now this

equation is rearranged for matrix format, using cross product matrix notation with the state accelerations on the left hand side,

$$m^{S_n}{}^i\vec{a}^G + m^{S_n}\left(\vec{r}^{\overrightarrow{Sn}}\right)^x {}^i\vec{\alpha}^b + m^{S_n}\ddot{\vec{ds}}_n = \vec{F}^{S_n} - m^{S_n}\left({}^i\vec{\omega}^b \times \left(2\dot{\vec{ds}}_n + {}^i\vec{\omega}^b \times \vec{r}^{\overrightarrow{S_n}}_{\overrightarrow{G}}\right)\right) \tag{2.66}$$

$\vec{F}^{S_n}$ is the sum of external forces on the n[th] slosh mass. These include the force of gravity, and the forces associated with spring and damper used to model the slosh, and a constraining force,

$$\vec{F}^{S_n} = m^{S_n}\left(-\omega_{Sn}{}^2\overrightarrow{ds}_n - 2\zeta_{Sn}\omega_{Sn}\dot{\vec{ds}}_n + \vec{g}_{body}\right) + F^{constraint}b\hat{x}. \tag{2.67}$$

It is assumed that the spring damper models all connection forces between the slosh mass and airframe in the $b\hat{y}$ and $b\hat{z}$. $F^{constraint}$ is constraining force imposed to make the $b\hat{x}$ slosh state zero and is defined by

$$F^{constraint} = -\left(\frac{1}{m^{S_n}}\left[\vec{F}^{S_n} - m^{S_n}\left({}^i\vec{a}^G + {}^i\vec{\alpha}^b \times \vec{r}^{\overrightarrow{S_n}}_{\overrightarrow{G}} + {}^i\vec{\omega}^b \times \left(2\left(\dot{ds}_{n_y}b\hat{y} + \dot{ds}_{n_z}b\hat{z}\right) + {}^i\vec{\omega}^b \times \vec{r}^{\overrightarrow{S_n}}_{\overrightarrow{G}}\right)\right)\right]\right) \cdot b\hat{x}. \tag{2.68}$$

2.7 Mass Matrix Format of Equations of Motion

As previously stated, the purpose of this derivation is to develop a set of equations of motion capable of being coded using computer software for the purpose of creating a simulation of the vehicle dynamics. The EOM's derived are coupled with one another;

therefore solving each equation individually creates algebraic loops. To avoid these loops, it is necessary to use previous states in coupled equations. This method is not optimal because the most current value of a state is not used in the calculation of other states.

Alternatively, the EOM's can be solved simultaneously. This is achieved by using a mass matrix format, in which the EOM's are placed in to the following format:

$$M\ddot{X} = \tilde{F}, \tag{2.69}$$

where M is a matrix of coefficients, akin to a mass matrix. $\tilde{F}$ is a force vector, containing external force terms and nonlinear terms acting on the system. The state vector, $\ddot{X}$, contains all relevant state accelerations,

$$\ddot{X} = \begin{bmatrix} {}^{i}\vec{a}^{G} \\ {}^{i}\vec{\alpha}^{b} \\ {}^{b}\vec{\alpha}^{e} \\ \ddot{d}_{s1} \\ \vdots \\ \ddot{d}_{sn} \end{bmatrix}. \tag{2.70}$$

These states from top to bottom are: translational acceleration of the gimbal with respect to the inertial frame, angular acceleration of the body with respect to the inertial frame, angular acceleration of the engine with respect to the body frame, and translational acceleration from equilibrium of the n^{th} slosh mass. This format makes solving the equations simultaneously a simple task: invert the M matrix and multiply by $\tilde{F}$ to find the state accelerations $\ddot{X}$.

Using this notation, the relevant equations of motion found in Equations (2.39), (2.55), (2.61), and (2.66) are assembled in matrix format,

$$\begin{bmatrix} m^t \mathrm{I}_{3x3} & S_G^t & m^e\left(\vec{r}^{\,\frac{G}{Ecm}}\right)^x & m^{S_1}\mathrm{I}_{3x3} & \cdots & m^{S_n}\mathrm{I}_{3x3} \\ 0_{3x3} & \underline{I}^{\frac{t}{Ccm}} & \underline{I}^{\frac{E}{Ccm}} & m^{S_1}\left(\vec{r}^{\,\frac{S_1}{Ccm}}\right)^x & \cdots & m^{S_n}\left(\vec{r}^{\,\frac{S_n}{Ccm}}\right)^x \\ m^e\left(\vec{r}^{\,\frac{G}{Ecm}}\right)^x & \underline{I}^{\frac{E}{G}} & \underline{I}^{\frac{E}{G}} & 0_{3x3} & \cdots & 0_{3x3} \\ m^{S_1}\mathrm{I}_{3x3} & m^{S_1}\left(\vec{r}^{\,\frac{G}{S1}}\right)^x & 0_{3x3} & m^{S_1}\mathrm{I}_{3x3} & \cdots & 0_{3x3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ m^{S_n}\mathrm{I}_{3x3} & m^{S_n}\left(\vec{r}^{\,\frac{G}{S_n}}\right)^x & 0_{3x3} & 0_{3x3} & \cdots & m^{S_n}\mathrm{I}_{3x3} \end{bmatrix} \begin{bmatrix} {}^i\vec{a}^G \\ {}^i\vec{\alpha}^b \\ {}^b\vec{\alpha}^e \\ \ddot{\vec{d}}_{s1} \\ \vdots \\ \ddot{\vec{d}}_{sn} \end{bmatrix} = \begin{bmatrix} \tilde{f}_s \\ \tilde{m}_s \\ \tilde{m}_e \\ \tilde{f}_{s1} \\ \vdots \\ \tilde{f}_{s_n} \end{bmatrix}, \tag{2.71}$$

where I_{3x3} is a 3x3 identity matrix, and 0_{3x3} is a 3x3 matrix populated with zeros. The symbol S_G^t denotes the total system first moment of mass taken around the gimbal,

$$S_G^t = m^a\left(-\vec{r}^{\,\frac{Acm}{G}}\right)^x + m^e\left(-\vec{r}^{\,\frac{Ecm}{G}}\right)^x + \sum_1^n m^{S_n}\left(-\vec{r}^{\,\frac{S_n}{G}}\right)^x. \tag{2.72}$$

The total vehicle inertia about the composite center of mass, $\underline{I}^{\frac{t}{Ccm}}$, is

$$\begin{aligned} \underline{I}^{\frac{t}{Ccm}} = \underline{I}^{\frac{A}{Acm}} + \underline{I}^{\frac{E}{Ecm}} + m^a\left(\vec{r}^{\,\frac{Acm}{Ccm}}\right)^x\left(\vec{r}^{\,\frac{G}{Acm}}\right)^x \\ + m^e\left(\vec{r}^{\,\frac{Ecm}{Ccm}}\right)^x\left(\vec{r}^{\,\frac{G}{Ecm}}\right)^x + \sum_1^n m^{S_n}\left(\vec{r}^{\,\frac{S_n}{Ccm}}\right)^x\left(\vec{r}^{\,\frac{G}{S_n}}\right)^x. \end{aligned} \tag{2.73}$$

The engine inertia taken about the composite center of mass, $\underline{I}^{\frac{E}{Ccm}}$, is calculated using the engine inertia about the engine center of mass and the parallel axis theorem,

$$\underline{I}^{\frac{E}{Ccm}} = \underline{I}^{\frac{E}{Ecm}} + m^e\left(\vec{r}^{\,\frac{Ecm}{Ccm}}\right)^x\left(\vec{r}^{\,\frac{G}{Ecm}}\right)^x. \tag{2.74}$$

The engine inertia taken about the gimbal, $\underline{I}^{\overline{G}}_{\overline{E}}$, is calculated using the engine inertia about the engine center of mass, and the parallel axis theorem,

$$\underline{I}^{\overline{G}}_{\overline{E}} = \underline{I}^{\overline{Ecm}}_{\overline{E}} + m^e \left(\vec{r}^{\overline{Ecm}}_{\overline{G}}\right)^x \left(\vec{r}^{\overline{Ecm}}_{\overline{G}}\right)^x. \tag{2.75}$$

The set of forcing terms applied to the translational acceleration of the gimbal (right hand side (RHS) of Equation (2.39)), $\tilde{f}_s$, is

$$\begin{aligned}\tilde{f}_s = \vec{F}^{sys} &- m^a\left({}^i\vec{\omega}^b \times \left({}^i\vec{\omega}^b \times \vec{r}^{\overline{Acm}}_{\overline{G}}\right)\right) \\ &- m^e\left(\left({}^i\vec{\omega}^b \times {}^b\vec{\omega}^e\right) \times \vec{r}^{\overline{Ecm}}_{\overline{G}} + {}^i\vec{\omega}^e \times \left({}^i\vec{\omega}^e \times \vec{r}^{\overline{Ecm}}_{\overline{G}}\right)\right) \\ &- \sum_1^n \left[m^{sn}\left({}^i\vec{\alpha}^b \times \vec{r}^{\overline{Sn}}_{\overline{G}} + {}^i\vec{\omega}^b \times \left(2\dot{\vec{d}}_{sn} + {}^i\vec{\omega}^b \times \vec{r}^{\overline{Sn}}_{\overline{G}}\right)\right)\right].\end{aligned} \tag{2.76}$$

The set of forcing terms applied to the angular acceleration of the body (RHS of Equation (2.55)), $\tilde{m}_s$, is

$$\begin{aligned}\tilde{m}_s = \vec{M}^{\overline{sys}}_{\overline{Ccm}} &- {}^i\vec{\omega}^b \times \left(\underline{I}^{\overline{A}}_{\overline{Acm}} \cdot {}^i\vec{\omega}^b\right) + \left(\underline{I}^{\overline{E}}_{\overline{Ecm}} \cdot {}^i\vec{\omega}^b \times {}^b\vec{\omega}^e\right) \\ &- {}^i\vec{\omega}^e \times \left(\underline{I}^{\overline{E}}_{\overline{Ecm}} \cdot {}^i\vec{\omega}^e\right) - \vec{r}^{\overline{Ecm}}_{\overline{Ccm}} \times m^e\left({}^i\vec{\omega}^e \times \left({}^i\vec{\omega}^e \times \vec{r}^{\overline{Ecm}}_{\overline{G}}\right)\right) \\ &- \sum_1^n \left[\vec{r}^{\overline{Sn}}_{\overline{Ccm}} \times m^{Sn}\left({}^i\vec{\omega}^b \times \left(2\dot{\vec{d}}_{Sn} + {}^i\vec{\omega}^b \times \vec{r}^{\overline{Sn}}_{\overline{G}}\right)\right)\right] \\ &- \vec{r}^{\overline{Acm}}_{\overline{Ccm}} \times m^a\left({}^i\vec{\omega}^b \times \left({}^i\vec{\omega}^b \times \vec{r}^{\overline{Acm}}_{\overline{G}}\right)\right) \\ &- m^e\vec{r}^{\overline{Ecm}}_{\overline{Ccm}} \times \left(\left({}^i\vec{\omega}^b \times {}^b\vec{\omega}^e\right) \times \vec{r}^{\overline{Ecm}}_{\overline{G}}\right).\end{aligned} \tag{2.77}$$

The set of forcing terms applied to the angular acceleration of the engine (RHS of Equation (2.61)), $\tilde{m}_e$, is

$$\tilde{m}_e = \vec{M}^{\frac{E}{G}} - \underline{I}^{\frac{E}{G}} \cdot \left({}^{i}\vec{\omega}^{b} \times {}^{b}\vec{\omega}^{e} \right) - {}^{i}\vec{\omega}^{e} \times \left(\underline{I}^{\frac{E}{G}} \cdot {}^{i}\vec{\omega}^{e} \right) - \vec{r}^{\frac{G}{Ecm}} \times \left(m^{e} \left(\left({}^{i}\vec{\omega}^{b} \times {}^{b}\vec{\omega}^{e} \right) \times \vec{r}^{\frac{Ecm}{G}} \right) \right). \tag{2.78}$$

The symbol $\tilde{f}_{S_n}$ denotes the set of forcing terms applied to the acceleration of slosh mass displacement (RHS of Equation (2.66)),

$$\tilde{f}_{S_n} = \vec{F}^{S_n} - m^{S_n} \left({}^{i}\vec{\omega}^{b} \times \left(2\dot{\vec{ds}}_n + {}^{i}\vec{\omega}^{b} \times \vec{r}^{\frac{Sn}{G}} \right) \right). \tag{2.79}$$

2.8 Simulation

The matrix format developed allows for simultaneous calculation of state accelerations. This is ideal for computer simulations, as algebraic loops due to the coupled nature of the individual EOM's are not a concern. The matrix set of equations is coded in computer software and solved to find state accelerations. These accelerations are integrated to find velocities, and again to find states.

Figure 2-3 shows a top level view of a Simulink model for this type of vehicle. Note that it is organized in the same manner as the generic G&C block diagram shown in Figure 1-1. The EOM's are contained in the "6DOF Dynamics" subsystem, and are coded in an embedded MATLAB file.

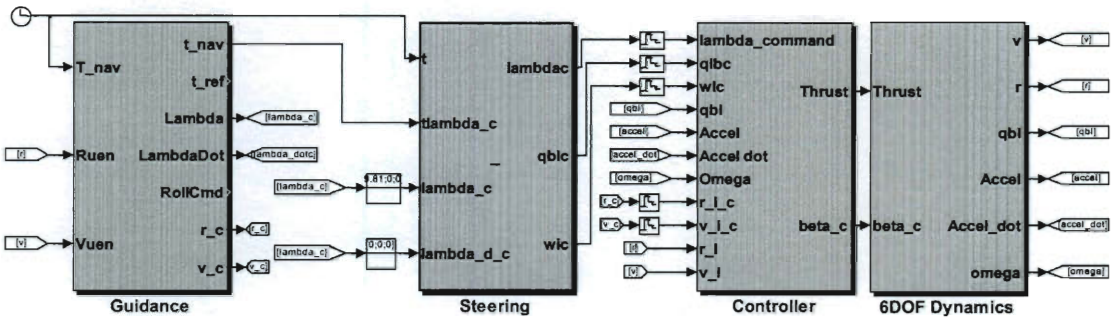

Figure 2-3: Top Level View of 6DOF Lander Simulink Model

Before the matrix format EOM's are coded though, a few modifications must be made in order for them to work properly in simulation. First, it is necessary to convert inertia dyadics to matrices to facilitate math operations in software (see Equations 11 and 12 of [19])

Rather than coding a constraining force $F^{constraint}$ to negate slosh in the $b\hat{x}$ direction, the $b\hat{x}$ states of the slosh equations are simply removed. In practice, this means removing the entire first row and column of every slosh state from the matrix set of EOM's, so that each slosh state is a 2x1 vector containing only $b\hat{y}$ and $b\hat{z}$ information.

Lastly, rather than developing a method to command an engine torque, $\vec{\tau}_\beta$, the engine gimbal angle can simply be commanded. This is known as prescribing gimbal rotation (as opposed to calculating it), and is discussed below.

2.8.1 Prescribed Gimbal Rotation

In order to simulate the previously developed EOM's using a G&C model, a method for calculating the control torque on the engine, $\vec{\tau}_\beta$, is required. This torque is an

output of the controller, and an accurate model of the gimbal system is necessary to determine the magnitude of the torque.

A simpler method for controlling the engine rotation in simulation is to simply prescribe the value of engine rotation angle, β. Taking this approach assumes an ideal engine model: the engine gimbal can achieve exactly the β angle that the controller specifies (note that a more realistic engine model can be implemented if this assumption is not sufficient). Using this method, the controller outputs the desired engine rotation angle, which is then used in the EOM's. Values for engine angular velocity and acceleration are found by differentiating the β signal coming out of the controller. A filter should be used in this differentiation so that unrealistic spikes of angular velocity or acceleration do not occur.

If the engine rotation angle is prescribed, the EOM for engine rotation is no longer necessary and must be removed from the matrix format. This does not mean that engine dynamics are removed entirely; they are still present in the other EOM's, except that engine rotation angle is no longer considered a state. The engine dynamics now appear as forcing terms on the other EOM's. In order to remove the engine dynamics EOM, the corresponding row and column for the engine rotation state, ${}^{b}\vec{\alpha}^{e}$, is removed, yielding

$$\begin{bmatrix} m^t \underline{I}_{3x3} & \underline{S}_G^t & m^{S_1}\underline{I}_{3x3} & \cdots & m^{S_n}\underline{I}_{3x3} \\ 0_{3x3} & \underline{I}^{\frac{t}{Ccm}} & m^{S_1}\left(\vec{r}^{\frac{Ccm}{S1}}\right)^x & \cdots & m^{S_n}\left(\vec{r}^{\frac{Ccm}{Sn}}\right)^x \\ m^{S_1}\underline{I}_{3x3} & m^{S_1}\left(\vec{r}^{\frac{G}{S1}}\right)^x & m^{S_1}\underline{I}_{3x3} & \cdots & \underline{0}_{3x3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m^{S_n}\underline{I}_{3x3} & m^{S_n}\left(\vec{r}^{\frac{G}{Sn}}\right)^x & \underline{0}_{3x3} & \cdots & m^{S_n}\underline{I}_{3x3} \end{bmatrix} \begin{bmatrix} {}^i\vec{a}^G \\ {}^i\vec{\alpha}^b \\ \ddot{\vec{d}}_{s1} \\ \vdots \\ \ddot{\vec{d}}_{sn} \end{bmatrix} = \begin{bmatrix} \tilde{f}_s^* \\ \tilde{m}_s^* \\ \tilde{f}_{s1}^* \\ \vdots \\ \tilde{f}_{Sn}^* \end{bmatrix}.$$

Doing this causes the terms of the $\tilde{F}$ vector to change, as terms that were once part of the engine state are now forcing terms. The elements of the new $\tilde{F}^*$ vector—denoted with an asterisk—are listed below.

The set of forcing terms applied to the translational acceleration of the gimbal including engine dynamics, $\tilde{f}_s^*$, is

$$\begin{aligned} \tilde{f}_s^* = \vec{F}^{sys} &- m^a\left({}^i\vec{\omega}^b \times \left({}^i\vec{\omega}^b \times \vec{r}^{\frac{Acm}{G}}\right)\right) \\ &- m^e\left(\left({}^i\vec{\omega}^b \times {}^b\vec{\omega}^e\right) \times \vec{r}^{\frac{Ecm}{G}} + {}^i\vec{\omega}^e \times \left({}^i\vec{\omega}^e \times \vec{r}^{\frac{Ecm}{G}}\right)\right) \\ &- \sum_1^n m^{sn}\left({}^i\vec{\omega}^b \times \left(2\dot{\vec{ds}}_n + {}^i\vec{\omega}^b \times \vec{r}^{\frac{Sn}{G}}\right)\right) \\ &- m^e\vec{r}^{\frac{G}{Ecm}} \times {}^b\vec{\alpha}^e. \end{aligned} \tag{2.80}$$

The set of forcing terms applied to the angular acceleration of the body with engine dynamics, $\tilde{m}_s^*$, is

$$\tilde{m}_s^* = \vec{M}^{\overrightarrow{sys}}_{\overrightarrow{Ccm}} - {}^i\vec{\omega}^b \times \left(\underline{I}^{\underline{A}}_{\overrightarrow{Acm}} \cdot {}^i\vec{\omega}^b\right) + \left(\underline{I}^{\underline{E}}_{\overrightarrow{Ecm}} \cdot {}^i\vec{\omega}^b \times {}^b\vec{\omega}^e\right)$$

$$- {}^i\vec{\omega}^e \times \left(\underline{I}^{\underline{E}}_{\overrightarrow{Ecm}} \cdot {}^i\vec{\omega}^e\right) - \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{Ccm}} \times m^e \left({}^i\vec{\omega}^e \times \left({}^i\vec{\omega}^e \times \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{G}}\right)\right)$$

$$- \sum_1^n \left[\vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{Ccm}} \times m^{Sn} \left({}^i\vec{\omega}^b \times \left(2\dot{\vec{d}}_{Sn} + {}^i\vec{\omega}^b \times \vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{G}}\right)\right)\right] \qquad (2.81)$$

$$- \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{Ccm}} \times m^a \left({}^i\vec{\omega}^b \times \left({}^i\vec{\omega}^b \times \vec{r}^{\overrightarrow{Acm}}_{\overrightarrow{G}}\right)\right)$$

$$- m^e \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{Ccm}} \times \left(\left({}^i\vec{\omega}^b \times {}^b\vec{\omega}^e\right) \times \vec{r}^{\overrightarrow{Ecm}}_{\overrightarrow{G}}\right) - \underline{I}^{\underline{E}}_{\overrightarrow{Ccm}} \cdot {}^b\vec{\alpha}^e.$$

The set of forcing terms applied to the displacement of slosh mass, $\tilde{f}_{S_n}$, does not change as engine dynamics are not directly coupled with this EOM,

$$\tilde{f}^*_{S_n} = \tilde{f}_{S_n} = \vec{F}^{S_n} - m^{S_n} \left({}^i\vec{\omega}^b \times \left(2\dot{\vec{d}}_{S_n} + {}^i\vec{\omega}^b \times \vec{r}^{\overrightarrow{Sn}}_{\overrightarrow{G}}\right)\right). \qquad (2.82)$$

2.8.1 Mass Parameters Calculation

The EOM's derived are subject to the quasi-static assumption discussed in Section 2, and as such they do not account for changing mass over time. Modeling depleting propellant mass is a critical aspect of simulation, so it is necessary to compute mass parameters separately. In the simulation, these parameters, including propellant mass, propellant CM location, and total mass are updated every time step and are passed to the EOM's as a constant for that time step.

In order to calculate propellant mass, Equation (2-3) from [23] is used to solve for the change in propellant mass, $\dot{m}_{prop}$,

$$\int \dot{m}_{prop}\, dt = \frac{\int F^{thrust} dt}{I_{sp} g_0}. \tag{2.83}$$

The time integral of $\dot{m}_{prop}$ is equal to the amount of propellant consumed, and can be subtracted from initial propellant mass to find the current propellant mass. In order to find the mass of oxidizer and fuel remaining, the mass of remaining propellant is simply multiplied by the oxidizer ratio for the oxidizer mass, and fuel ratio for the fuel mass [23]. The total current mass of the vehicle is the current mass of propellant plus the dry mass of the vehicle.

The EOM's also require knowledge of the location of the undisplaced slosh mass CM, which changes as propellant is depleted. The parameters of a partially filled spherical tank necessary to find the CM location of the undisplaced propellant are shown in Figure 2-4.

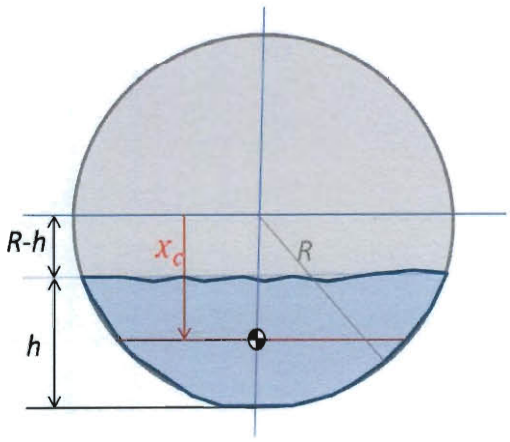

Figure 2-4: Parameters of Partially Filled Spherical Tank

In order to find x_c, the distance from the center of the tank to the undisplaced slosh mass CM, the following equation is used to find the height of the fuel, h,

$$h = \left(1 - 2\cos\left(\frac{\pi + cos^{-1}(1-2\xi)}{3}\right)\right)R. \tag{2.84}$$

Where $\xi = \frac{m}{m_0}$ and is the ratio of current mass of propellant to mass of propellant in a full tank ($m_0 = \frac{4}{3}\pi\rho R^3$, ρ is propellant density). x_c therefore equals

$$x_c = -\frac{3(2R-h)^2}{4(3R-h)} \text{ [24]}. \tag{2.85}$$

Finally, it is possible to find $\overrightarrow{ls_n}$, the vector from the gimbal to the undisplaced slosh mass CM,

$$\overrightarrow{ls_n} = \overrightarrow{Rs_n} + x_c \cdot b\hat{x}. \tag{2.86}$$

Where $\overrightarrow{Rs_n}$ is the vector from the gimbal to the center of the propellant tank. $\overrightarrow{ls_n}$ can now be used in the vehicle EOM's.

2.8.2 Planet Centered Gravity Model

For entry and landing applications where the flight path of the vehicle covers a large ground track and/or significant altitude change, it is necessary to use a planet centered inertial frame in order to model changing gravity. This means that the origin of the inertial frame is placed at the center of the planet that the vehicle is operating on.

Using a planet centered inertial frame, the acceleration of gravity changes proportionally with the mass of the planet, and inversely with the square of the distance from the origin inertial frame (center of planet). Whereas before with a flat Earth assumption, where the gravity vector pointed downwards in the inertial $i\hat{X}$ direction, the gravity vector now points from the body in question to the origin of the inertial frame.

To implement a planet centered gravity model, the acceleration of gravity is calculated using

$$g = \frac{-GM^{planet}}{\left\| \vec{r}^{\frac{G}{i_o}} \right\|^2}, \tag{2.87}$$

where G is the gravitational constant (= 6.6742×10^{-11} N·m^2/kg^2), M^{planet} is the mass of the planet, and $\vec{r}^{\frac{Ccm}{i_o}}$ is the vector from the origin of the inertial frame to the composite center of mass of the vehicle in question.

The equation above is a scalar for the acceleration of gravity, which can be converted into a vector for the force of gravity in the inertial frame,

$$\vec{F}^i_{grav} = m^t g \frac{\vec{r}^{\frac{G}{i_o}}}{\left\| \vec{r}^{\frac{G}{i_o}} \right\|} \text{ [25].} \tag{2.88}$$

This is the total mass of the vehicle, times the acceleration of gravity, times the unit vector from the inertial frame to the composite center of mass of the vehicle.

The equations of motion for the vehicle are calculated in the body frame, which means the force of gravity in the body frame must be used. The direction cosine matrix from inertial to body, $DCM_{i\to b}$, is multiplied by the force of gravity in the inertial frame to obtain the force of gravity in the body frame,

$$\vec{F}^b_{grav} = DCM_{i\to b}\vec{F}^i_{grav}\,. \tag{2.89}$$

This vector can now be applied to the EOM's of the vehicle, just it was applied using a flat Earth assumption.

2.8.3 Modeling Atmospheric Drag and Wind

For a vehicle operating on a planet with an atmosphere, such as Earth or Mars, the aerodynamic force of drag should be taken into account. Drag is modeled as an external force on the system. Specifically, the term for the force of drag, $\vec{F}_{drag}$, appears in the equation of motion for translational acceleration of the body, ${}^{i}\vec{a}^{G}$.

The force of drag is assumed to act at the composite center of mass of the vehicle. As a result, it does not create a moment on the vehicle. This assumption only holds true if the cross section of the vehicle facing the direction of the relative wind is symmetric about the composite center of mass. $\vec{F}_{drag}$ is an additional term of $\vec{F}^{sys}$, the sum of external forces on the system.

The equation for the force of drag of a fluid on an object is

$$\vec{F}_{drag} = -\frac{1}{2}\rho A C_d(\vec{v}\cdot\vec{v})\frac{\vec{v}}{\|v\|}\ [26]. \tag{2.90}$$

where the density of the fluid is denoted by ρ, A is the reference area of object traveling through fluid (m^2), C_d is the coefficient of drag of the object (*unitless*), and $\vec{v}$ is the velocity of object (m/s).

For the analysis herein, wind is modeled as an additional drag force on the vehicle. The wind velocity vector is added to the velocity vector of the vehicle to create a velocity relative to wind vector that is used in the force of drag equation above. For this analysis, wind speed and direction is modeled using stochastic techniques.

3 Guidance Methods

The goal of guidance is to develop a trajectory for a vehicle to get from an initial state to a desired target state in a specified time of flight, t_{go}. For the guidance algorithm developed herein, the initial and final states are composed of a position, velocity, and acceleration:

$$\vec{r}_0, \vec{v}_0, \vec{a}_0 - initial\ position, velocity, and\ acceleration$$

$$\vec{r}_t, \vec{v}_t, \vec{a}_t - target\ position, velocity, and\ acceleration$$

Guidance works by solving for an acceleration profile, $\vec{a}(t)$, that will take the vehicle from its initial state to the target state. The time functions for velocity, $\vec{v}(t)$, and position, $\vec{r}(t)$, are simply the integral and double integral of $\vec{a}(t)$, with respect to time. The target state is a point on these curves, with time value corresponding to the time of flight, t_{go}, to get from the initial state to the final state,

$$\begin{aligned} \vec{a}(t_{go}) &= \vec{a}_t \\ \vec{v}(t_{go}) &= \vec{v}_t \\ \vec{r}(t_{go}) &= \vec{r}_t. \end{aligned} \tag{3.1}$$

In addition to solving for the acceleration profile, t_{go} must either be solved for, or specified. This is an important point as it will be seen below that if $t_{go}$is not specified, there can be more unknowns than equations. If this is the case, either constraints must be added, or a search method used to find t_{go}.

Guidance views the vehicle as a point mass, so the vehicle body orientation is not taken into consideration. Guidance does not specify vehicle rotation; it is up to steering

and the control to solve for necessary vehicle orientation in order to meet the acceleration requirements dictated by guidance.

To solve for the acceleration profile to get the vehicle to its target state, the initial and target states become the boundary conditions (BC's) for a two point boundary value problem (BVP) (see Figure 3-1).

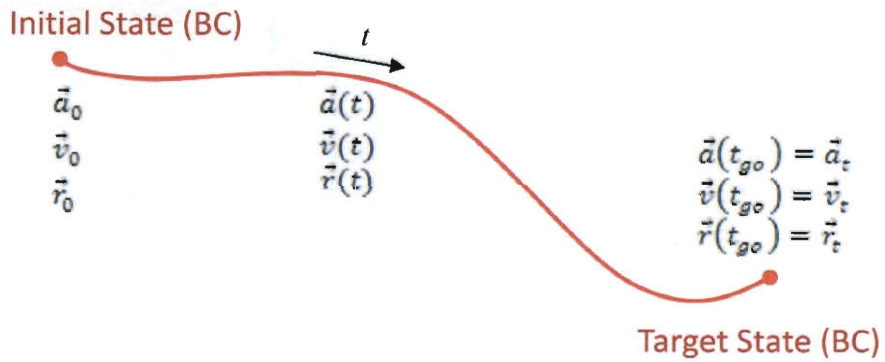

Figure 3-1: Setup for Two Point Boundary Value Problem

3.1 Acceleration Profiles

The acceleration profile for a trajectory is any acceleration function that connects the two BC's shown in Figure 3-1. Commonly, the acceleration profile is chosen to be a polynomial over time. Though any order polynomial can be used, only cubic, quadratic, and linear polynomials are examined herein. These three polynomials are sufficient to show how the initial and target acceleration boundary conditions can be free or constrained, depending on the polynomial chosen.

3.1.1 Linear Acceleration Profile

If a linear acceleration profile is chosen the vehicle acceleration as a function of time is given by

$$\vec{a}(t) = \vec{C}_0 + \vec{C}_1 t. \tag{3.2}$$

The functions for velocity and position are therefore:

$$\vec{v}(t) = \vec{C}_0 t + \frac{1}{2}\vec{C}_1 t^2 + \vec{v}_0, \tag{3.3}$$

$$\vec{r}(t) = \frac{1}{2}\vec{C}_0 t^2 + \frac{1}{6}\vec{C}_1 t^3 + \vec{v}_0 t + \vec{r}_0. \tag{3.4}$$

For these equations the unknowns are $\vec{C}_0$, $\vec{C}_1$, and t. Both acceleration BCs, $\vec{a}_0$, and $\vec{a}_t$ are *free*, and need not be specified. Once again this works under the assumption that the vehicle can achieve instantaneous acceleration changes at BCs.

For this case, $\vec{C}_0$ and $\vec{C}_1$ contain six unknowns, but there are now only six solvable equations as $\vec{a}(t_{go}) = \vec{a}_t$ is free. So just as before, either t_{go} must be specified or the system must be further constrained.

If a search method is used to specify t_{go}, the explicit solution for the coefficients of the linear acceleration profile are:

$$\begin{bmatrix}\vec{C}_0 \\ \vec{C}_1\end{bmatrix} = \begin{bmatrix} t_{go} I_{3x3} & \frac{1}{2} t_{go}^2 I_{3x3} \\ \frac{1}{2} t_{go}^2 I_{3x3} & \frac{1}{6} t_{go}^3 I_{3x3}\end{bmatrix}^{-1} \left(\begin{bmatrix}\vec{v}_t \\ \vec{r}_t\end{bmatrix} - \begin{bmatrix} \vec{v}_0 \\ \vec{v}_0 t_{go} + \vec{r}_0\end{bmatrix}\right) \tag{3.5}$$

3.1.2 Quadratic Acceleration Profile

If a quadratic acceleration profile is chosen the vehicle acceleration as a function of time is given by the equation

$$\vec{a}(t) = \vec{C}_0 + \vec{C}_1 t + \vec{C}_2 t^2. \tag{3.6}$$

Therefore the functions for velocity and position are:

$$\vec{v}(t) = \vec{C}_0 t + \frac{1}{2}\vec{C}_1 t^2 + \frac{1}{3}\vec{C}_2 t^3 + \vec{v}_0, \tag{3.7}$$

$$\vec{r}(t) = \frac{1}{2}\vec{C}_0 t^2 + \frac{1}{6}\vec{C}_1 t^3 + \frac{1}{12}\vec{C}_2 t^4 + \vec{v}_0 t + \vec{r}_0. \tag{3.8}$$

For this case the unknowns are $\vec{C}_0$, $\vec{C}_1$, $\vec{C}_2$ and t. This time $\vec{a}_0$ is *free*, meaning it does not have to be specified as a BC. This is valid given the assumption that the vehicle thruster is capable of providing an instantaneous acceleration change at the initial BC. As with the cubic profile, there are nine equations, ten unknowns, so a search method must be used for t_{go}, or an additional constraint added.

Assuming t_{go} is specified by a search method, the solution for the coefficients $\vec{C}_0$, $\vec{C}_1$, $\vec{C}_2$ is

$$\begin{bmatrix} \vec{C}_0 \\ \vec{C}_1 \\ \vec{C}_2 \end{bmatrix} = \begin{bmatrix} I_{3x3} & t_{go}I_{3x3} & t_{go}^2 I_{3x3} \\ t_{go}I_{3x3} & \frac{1}{2}t_{go}^2 I_{3x3} & \frac{1}{3}t_{go}^3 I_{3x3} \\ \frac{1}{2}t_{go}^2 I_{3x3} & \frac{1}{6}t_{go}^3 I_{3x3} & \frac{1}{12}t_{go}^4 I_{3x3} \end{bmatrix}^{-1} \left(\begin{bmatrix} \vec{a}_t \\ \vec{v}_t \\ \vec{r}_t \end{bmatrix} - \begin{bmatrix} 0_{3x1} \\ \vec{v}_0 \\ \vec{v}_0 t_{go} + \vec{r}_0 \end{bmatrix} \right), \tag{3.9}$$

where 0_{3x1} is a 3x1 vector of zeros. This is the solution for the coefficients of a cubic acceleration profile, given t_{go}. The initial acceleration BC, $\vec{a}_0$, is free, but the target acceleration, $\vec{a}_t$, must be specified.

3.1.3 Cubic Acceleration Profile

Starting with a cubic acceleration profile, the acceleration as a function of time is given by

$$\vec{a}(t) = \vec{C}_0 + \vec{C}_1 t + \vec{C}_2 t^2 + \vec{C}_3 t^3, \tag{3.10}$$

where $\vec{C}_0, \vec{C}_1, \vec{C}_2$, and $\vec{C}_3$ are coefficient vectors that must be solved for. For this case neither $\vec{a}_0$ or $\vec{a}_t$ are free, therefore $\vec{C}_0 = \vec{a}_0$. Equation (3.10) can be integrated once, and integrated again, to find equations for velocity and position, respectively,

$$\vec{v}(t) = \vec{C}_0 t + \frac{1}{2}\vec{C}_1 t^2 + \frac{1}{3}\vec{C}_2 t^3 + \frac{1}{4}\vec{C}_3 t^4 + \vec{v}_0, \tag{3.11}$$

$$\vec{r}(t) = \frac{1}{2}\vec{C}_0 t^2 + \frac{1}{6}\vec{C}_1 t^3 + \frac{1}{12}\vec{C}_2 t^4 + \frac{1}{20}\vec{C}_3 t^5 + \vec{v}_0 t + \vec{r}_0. \tag{3.12}$$

Given a three dimensional problem space, these functions are 3x1 vectors containing x, y, and z components. The coefficients $\vec{C}_0(=\vec{a}_0), \vec{C}_1, \vec{C}_2$, and $\vec{C}_3$ are therefore 3x1 vectors, yielding nine unknowns plus the time of flight variable t, for a total of ten unknowns. The terms for $\vec{a}(t)$, $\vec{v}(t)$, and $\vec{r}(t)$ together represent nine equations, less than the number of unknowns. So the system of equations must be further constrained, or at least one variable must be specified. The approach taken here is to use a search method to find the minimum possible time of flight, t_{go}, that meets given vehicle acceleration constraints. This value of t_{go} is then substituted for t in the system of equations above to find the coefficients for the cubic acceleration profile.

Using the specified t_{go} derived from the search described below, explicit solutions for the coefficients $\vec{C}_1, \vec{C}_2$, and $\vec{C}_3$ can be found,

$$\begin{bmatrix} \vec{C}_1 \\ \vec{C}_2 \\ \vec{C}_3 \end{bmatrix} = \begin{bmatrix} t_{go} I_{3x3} & t_{go}^2 I_{3x3} & t_{go}^3 I_{3x3} \\ \frac{1}{2} t_{go}^2 I_{3x3} & \frac{1}{3} t_{go}^3 I_{3x3} & \frac{1}{4} t_{go}^4 I_{3x3} \\ \frac{1}{6} t_{go}^3 I_{3x3} & \frac{1}{12} t_{go}^4 I_{3x3} & \frac{1}{20} t_{go}^5 I_{3x3} \end{bmatrix}^{-1} \left(\begin{bmatrix} \vec{a}_t \\ \vec{v}_t \\ \vec{r}_t \end{bmatrix} - \begin{bmatrix} \vec{C}_0 \\ \vec{C}_0 t_{go} + \vec{v}_0 \\ \frac{1}{2}\vec{C}_0 t_{go}^2 + \vec{v}_0 t_{go} + \vec{r}_0 \end{bmatrix} \right), \tag{3.13}$$

where I_{3x3} is a 3x3 identity matrix. Equation (3.13) is the solution to the coefficients of a cubic acceleration profile, given t_{go}.

Note that for this case both acceleration BC's, $\vec{a}_0$ and $\vec{a}_t$, must be specified. This is significant because when a multiple waypoint trajectory is being followed, the target acceleration $\vec{a}_t$ of one waypoint becomes the initial $\vec{a}_0$ for the trajectory to get to the next waypoint. This fact insures that there are no acceleration discontinuities over the acceleration profile of the entire flight, unlike the other acceleration profiles generated by other methods detailed below. This is advantageous because the engine never has to provide an instantaneous change of acceleration, which may not be possible.

3.1.4 Constant Acceleration Profile for Minimum Fuel Maneuvers

The three-dimensional (3D) minimum-fuel problem for the powered terminal descent of a Mars Lander is formulated in [18]. According to [18], the most general thrust magnitude profile for a minimum fuel landing has a max-min-max structure. The most fuel efficient thrust profile is always of the max-min nature, where the engine is always at maximum or minimum thrust.

This section develops a method for a vehicle to translate either vertically or horizontally using only maximum or minimum thrust settings.

3.1.4.1 Vertical Maneuvers (Ascent and Descent)

The optimal thrust program for a minimum fuel lunar landing is examined in [10]. The development of the minimum fuel solution for a one dimensional (1D) landing (vertical descent) is detailed:

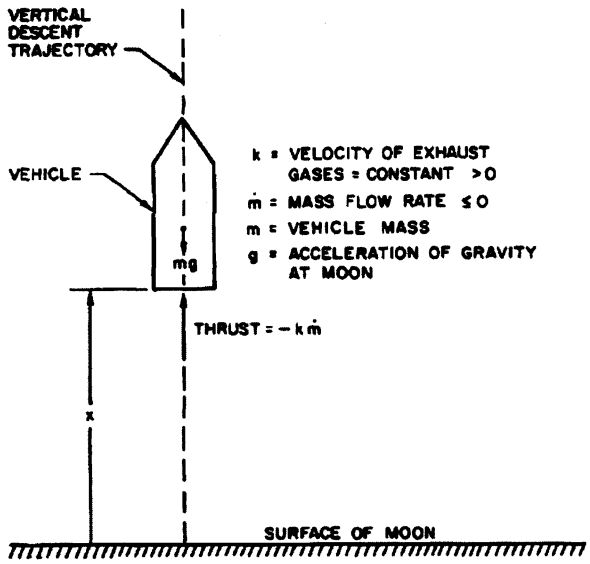

Figure 3-2: Terminal Phase of Lunar Soft Landing [10]

The following assumptions are taken for analysis: 1.) The only forces acting of the vehicle are weight and thrust, 2.) Thrust direction is parallel to the descent of the vehicle (1D), 3.) The planet is flat in the vicinity of the landing, 4.) Gravity is constant, 5.) Velocity of exhaust gases is constant with respect to the vehicle, and 6.) The engine provides mass flow rates between zero and a set upper limit [10].

According to [10], "the optimal thrust program consists of either full thrust form initiation of the mission until touchdown, or a period of zero thrust (free-fall) followed by full thrust until touchdown". Essentially, for a vertical descent problem, the fuel optimal solution consists of the vehicle's engine being either on (at its maximum setting) or off (at its minimum setting).

In order to ensure a soft landing (v_t is near zero), for minimum fuel use, a descending vehicle starts in a free fall (at minimum thrust setting), and then at the

appropriate time, switches to maximum thrust so that it touches down with approximately zero velocity:

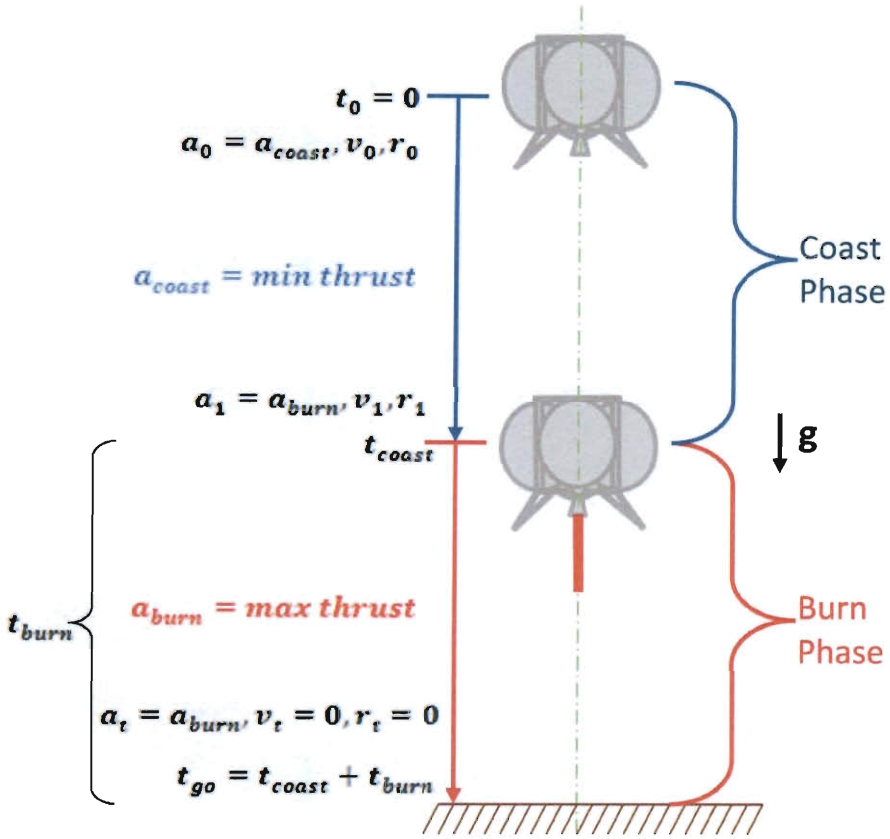

Figure 3-3: Setup for Min-Max Acceleration Vertical Descent

For the scenario shown in Figure 3-3, the vehicle starts from a known initial state (v_o and r_o), with the engine off ($a_o = a_{coast}$). The goal of the descent maneuver is to get the vehicle to ground level ($r_t = 0$), at a velocity of zero ($v_t = 0$). This target state is achieved by initiating the engine at maximum thrust at time $t = t_{coast}$ giving the vehicle a constant acceleration of a_{burn} until $t = t_{go} = t_{coast} + t_{burn}$ when the vehicle touches down. The acceleration, velocity, and position vs. time plots for this scenario resemble:

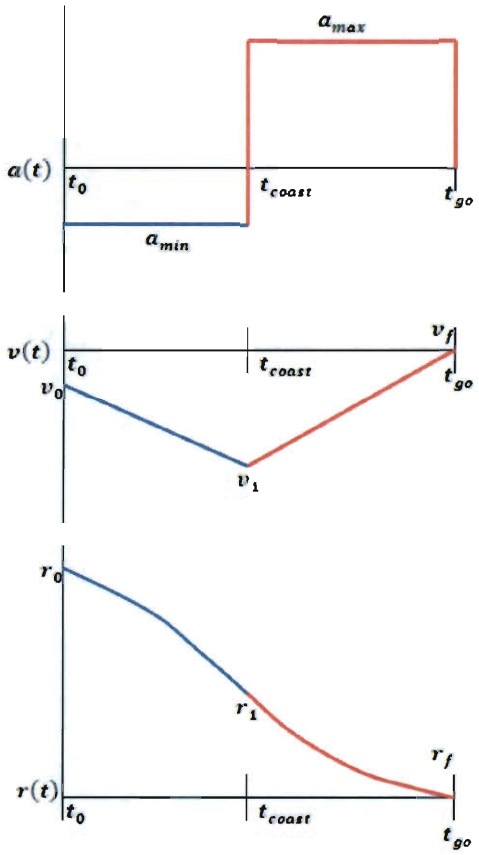

Figure 3-4: Acceleration, Velocity, and Position vs. Time Plots for Vertical Descent

Given the initial and final states for this scenario, the unknowns are r_1, v_1, t_{coast}, and t_{burn}. These unknowns can be explicitly found by setting up the kinematic equations of motion for the vehicle during the two phases of the descent scenario. To do this, the vehicle is assumed to be a point mass, and the only forces acting on the vehicle are thrust and gravity. Thrust is a constant value corresponding to the minimum throttle setting for the coast phase, and the maximum throttle setting for the burn phase. The kinematic acceleration for the coast and burn phase is therefore

$$a_{coast} = \frac{F_{min}^{thrust}}{m^t} + g, \tag{3.14}$$

$$a_{burn} = \frac{F_{max}^{thrust}}{m^t} + g. \tag{3.15}$$

where g is the acceleration of gravity (a negative number).

Setting up the kinematic equations of motion for the coast phase yields

$$v_1 = v_0 + a_{coast}t_{coast}, \tag{3.16}$$

$$r_1 = r_0 + v_0 t_{coast} + \frac{1}{2}a_c t_{coast}^2. \tag{3.17}$$

The kinematic equations of motion for the burn phase are

$$v_f = v_1 + a_{burn}t_{burn}, \tag{3.18}$$

$$r_f = r_1 + v_1 t_{burn} + \frac{1}{2}a_{burn}t_{burn}^2. \tag{3.19}$$

There are a total of four equations, four unknowns, with the relevant unknowns being t_{coast} and t_{burn}, the amount of time guidance will command the controller to keep the engine off and on, respectively.

Substituting Equation (3.16) into Equation (3.17) for t_{coast}, an expression for r_1 in terms of v_1 is found,

$$r_1 = r_0 + v_0\left(\frac{v_1 - v_0}{a_{coast}}\right) + \frac{1}{2}a_{coast}\left(\frac{v_1 - v_0}{a_{coast}}\right)^2. \tag{3.20}$$

This expression is substituted into Equation (3.19) for r_1. Equation (3.18) is also substituted into the same equation for t_{burn} to yield an expression in terms of only v_1 and constants,

$$r_f = r_0 + v_0\left(\frac{v_1 - v_0}{a_{coast}}\right) + \frac{1}{2}a_{coast}\left(\frac{v_1 - v_0}{a_{coast}}\right)^2 + v_1\left(\frac{v_f - v_1}{a_{burn}}\right) + \frac{1}{2}a_{burn}\left(\frac{v_f - v_1}{a_{burn}}\right)^2. \tag{3.21}$$

Factoring out the v_1 terms and simplifying yields

$$0 = \left(\frac{1}{2a_{coast}} - \frac{1}{2a_{burn}}\right)v_1^2 + \left(r_0 - r_f + \frac{v_0^2}{2a_{coast}} + \frac{v_f^2}{2a_{burn}}\right). \tag{3.22}$$

Finally, the quadratic equation is used to solve for v_1,

$$v_1 = \frac{\pm\sqrt{-4\left(\frac{1}{2a_{coast}} - \frac{1}{2a_{burn}}\right)\left(r_0 - r_f + \frac{v_0^2}{2a_{coast}} + \frac{v_f^2}{2a_{burn}}\right)}}{2\left(\frac{1}{2a_{coast}} - \frac{1}{2a_{burn}}\right)}. \tag{3.23}$$

This equation yields two solutions for v_1, one positive and one negative. Only the negative value of v_1 is correct as the vehicle must be moving downwards at the end of the coast phase.

The negative value for v_1 can now be inserted into the following expressions for t_{coast} and t_{burn},

$$t_{coast} = \frac{v_1 - v_0}{a_{coast}}, \tag{3.24}$$

$$t_{burn} = \frac{v_f - v_1}{a_{burn}}. \tag{3.25}$$

These are the explicit solutions for the coast time and burn time in a vertical landing scenario.

The method can be used for ascent if the burn phase and coast phase are switched so that the vehicle starts with its engine on for lift-off.

3.1.4.2 Down Range Maneuvers

Down range maneuvers, where the vehicle is moving in the horizontal plane, are considerably more complicated than vertical maneuvers with regards to fuel use optimality. This is because gravity does not act as a decelerating force for lateral maneuvers, so the vehicle must accelerate and decelerate itself. Taking into account [14]'s development of the max-min-max thrust structure for fuel efficiency, a fuel efficient lateral maneuver consists of three phases:

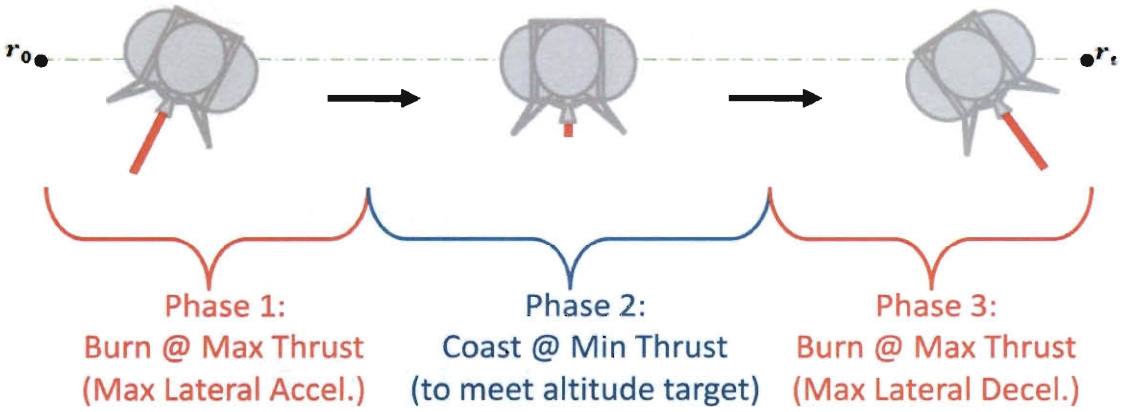

Figure 3-5: Phases of Fuel Efficient Lateral (Down Range) Maneuver

The corresponding acceleration, velocity, and position profiles for this maneuver resemble:

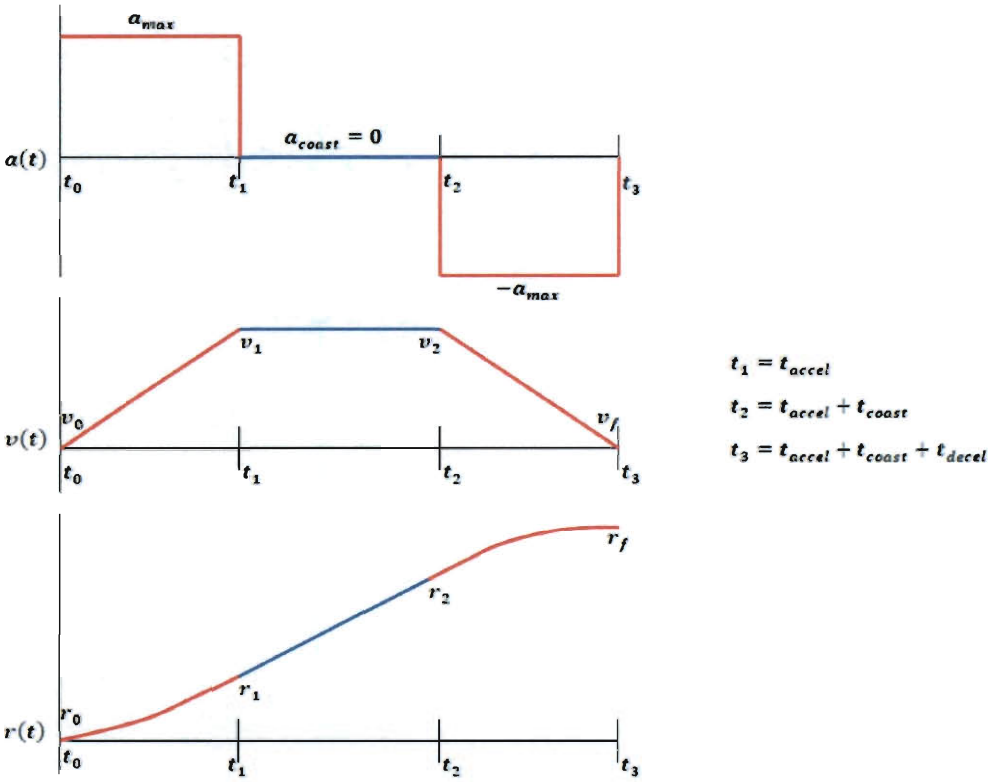

Figure 3-6: Acceleration, Velocity, and Position vs. Time Plots for Lateral Maneuver

For the case shown above, a_{max} is the maximum lateral acceleration of the vehicle, given thrust, pitch, and gimbal constraints. Assuming a point mass for the vehicle, the kinematic equations of motion for this maneuver are:

$$
\begin{aligned}
v_1 &= v_0 + a_{max} t_{accel} \\
r_1 &= r_0 + v_0 t_1 + \frac{1}{2} a_{max} t_{accel}^2 \\
v_2 &= v_1 + a_{coast} t_{coast} \\
r_2 &= r_1 + v_1 t_{coast} + \frac{1}{2} a_{coast} t_{coast}^2 \\
v_f &= v_2 - a_{max} t_{decel} \\
r_f &= r_2 + v_2 t_{decel} - \frac{1}{2} a_{max} t_{decel}^2 .
\end{aligned}
\tag{3.26}
$$

The unknowns for this system of equations are: r_1, v_1, r_2, v_2, t_{accel}, t_{coast}, and t_{decel}. For optimum fuel use efficiency, an optimization method should be used to find system unknowns that minimize fuel, as there is one more unknown than equation. For simplicity in this analysis, t_{coast} is set to zero so that an explicit solution can be found. This yields a max-max thrust maneuver resembling:

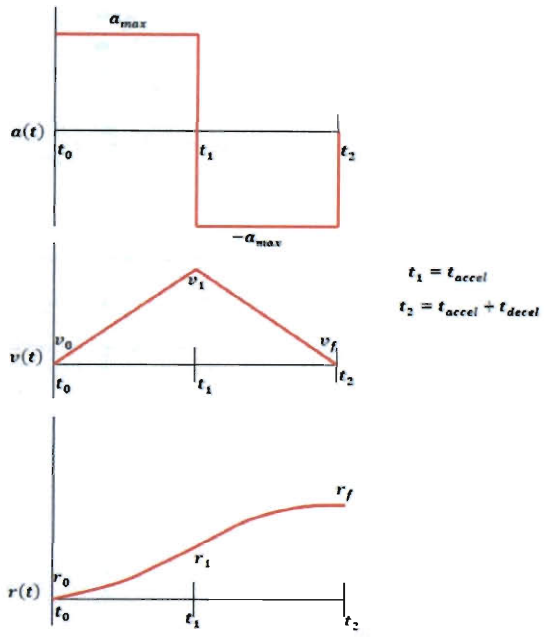

Figure 3-7: State Time Plots for Lateral Maneuver with No Coast Phase

When t_{coast} is set to zero, there are the same amount of unknowns as equation and the system can be solved in the same manner as for vertical maneuvers, shown in Section 3.1.4.1. The solution for v_1 for this maneuver is

$$v_1 = \frac{a_{max}}{2}\sqrt{-\frac{4}{a_{max}}\left(r_0 - r_f - \frac{v_0^2+v_f^2}{2a_{max}}\right)}. \tag{3.27}$$

And the solutions for t_{accel} and t_{decel} in terms of v_1 are

$$\begin{aligned} t_{accel} &= \frac{v_1 - v_0}{a_{max}}, \\ t_{decel} &= \frac{v_1 - v_f}{a_{max}}. \end{aligned} \tag{3.28}$$

This method for commanding down range acceleration and the method of vertical ascent and descent presented above only take into account one dimension of movement

coincident with the path of the maneuver. However, it may still be necessary to make course corrections or small maneuvers in the other two dimensions. Using the max-min-max thrust structure for these small movements may not be implementable, because of thruster response rates (gimbal and thrust magnitude rates). Another method is necessary to command accelerations in these other directions of movement.

A simple method to achieve this is to use a polynomial acceleration profile in directions other than the direction of the large maneuver. This combination of max-min-max thrust and polynomial acceleration yields efficient fuel use in the direction of flight that requires the most fuel, and also allows accelerations to be commanded in other directions, if necessary.

As an example, if a vehicle is making a large downrange maneuver, it would use constant acceleration profile consisting of maximum and minimum thrust commands in the downrange direction, and a polynomial acceleration profile in the cross range and vertical directions. Using a linear acceleration profile for the cross range and vertical course corrections, the solution to the $\vec{C}_0$ and $\vec{C}_1$ constants (used to develop acceleration commands for the controller) would be

$$\vec{C}_0 = \begin{cases} \pm a_{max}, & \begin{matrix} down\ range \\ (direction\ of\ maneuver) \end{matrix} \\ \dfrac{\left((-2v_t - 4v_0)t_{go} + 6(r_t - r_0)\right)}{t_{go}^2}, & crossrange\ and\ vertical \end{cases}$$

$$\vec{C}_1 = \begin{cases} 0, & \begin{matrix} down\ range \\ (direction\ of\ maneuver) \end{matrix} \\ \dfrac{\left(6(v_t + v_0)t_{go} - 12(r_t - r_0)\right)}{t_{go}^3}, & crossrange\ and\ vertical \end{cases} . \tag{3.29}$$

The cross range and vertical terms shown above are equivalent to the linear acceleration coefficients of Equation (3.5). The $\pm a_{max}$ term of the $\vec{C}_0$ coefficient refers to either maximum acceleration or deceleration, and is determined using the process described above.

As a final note, this analysis assumes that the engine can turned on or off instantaneously. In reality this is not the case, as the inherent complexity of rocket engines produce non-linear interactions that lead to inevitable delays in throttle control [27]. A good engine model is necessary to analyze the effects of these interactions on what kind of acceleration change the engine can actually provide. Ideally, such an engine model would be incorporated in guidance to insure that the commanded rate of change of acceleration (jerk) is not unrealistic.

3.2 Computing Time of Flight

For all of the acceleration profiles discussed, there is one more unknown than equation. In order to overcome this, one of the unknowns must be prescribed or separately computed, through a search method or otherwise. The time of flight variable, t_{go}, is a convenient unknown to specify because its value dictates operating time of the vehicle, which in turn has a significant effect on fuel use.

If a desired time of flight is known, t_{go} can simply be prescribed, and the coefficients for the acceleration profile computed. Alternatively, a local search method can be used to find the minimum t_{go} that is possible given physical constraints. Or a range of t_{go}'s can be examined, and the one yielding a feasible acceleration profile with the least estimated fuel use can be chosen.

Regardless of the method used to specify t_{go} or any other unknown, it is necessary to check whether or not a calculated acceleration profile meets the physical constraints of the vehicle. This is done to ensure that the trajectory passed from guidance to is feasible.

3.2.1 Applying Physical Constraints to Generate Feasible Trajectories

Just because a particular t_{go} is specified does not mean the calculated acceleration profile meets the physical constraints of what the vehicle is capable of providing. Such constraints include maximum acceleration that is proportional to the maximum thrust of the engine, minimum acceleration that is roughly equal to gravity (imposed so that the vehicle does not flip over in an attempt to accelerate downwards), other flight path constraints, etc.

For example, if a very small t_{go} is chosen, a very high peak acceleration will be necessary to get the vehicle from its initial state to the target state. If the vehicle thruster is not capable of providing this acceleration, the trajectory will not be followed and the target state may not be reached.

Given that any rocket engine has a maximum thrust level, there is a maximum possible acceleration for a vehicle, proportional to this thrust level and gravity. This can be seen by examining Newton's second law for the entire system,

$$m^t \left\| \vec{a}^{vehicle} \right\| = F^{thrust} - m^t \|\vec{g}\|. \tag{3.30}$$

This equation shows that the total acceleration of the vehicle times its total mass is equal to the force of thrust minus the force of gravity (assuming only thrust and gravity are

acting on the vehicle). Rearranging this equation yields the maximum possible acceleration for the vehicle

$$\left\|\vec{a}_{max}^{vehicle}\right\| = \frac{F_{max}^{thrust} - m^t\|\vec{g}\|}{m^t}. \tag{3.31}$$

Regardless of the method used to find an acceleration profile, the maximum value of the profile must be less than or equal to that of Equation (3.31) to ensure feasibility of the profile.

Another acceleration constraint that is useful to apply is a no flip constraint. In a situation where a vehicle is traveling downwards, it would be highly undesirable to have the vehicle flip over in order to use the thruster to accelerate downwards, because of the control issues involved with this. In order to keep the vehicle in an upright orientation at all times, a no-flip constraint can be applied by imposing a minimum acceleration in the vertical direction. This constraint is similar to the maximum acceleration constraint of Equation (3.31),

$$\left(a_{min}^{vehicle}\right)\hat{\imath} = \left(\frac{F_{min}^{thrust} - m^t\|\vec{g}\|}{m^t}\right)\hat{\imath}. \tag{3.32}$$

The minimum acceleration in the local vertical direction ($\hat{\imath}$) is equal the minimum thrust level minus the force of gravity divided by the total vehicle mass. The minimum value in the vertical direction of a computed acceleration profile must be greater than or equal to the minimum value for acceleration shown in Equation (3.32).

Other physical limits can be accounted for by applying constraints to the acceleration profile, such as constraints in the horizontal plane for how much the thruster

can swivel or the vehicle can tilt. Together these constraints help guidance develop an acceleration profile for a feasible trajectory.

3.2.2 Using Local Search to Find Minimum Feasible t_{go}

If the time of flight t_{go} must be selected in such a way to minimize flight time, a simple local search algorithm can be implemented to find $\boldsymbol{t_{go}}$. A local search method is a metaheuristic used solve optimization problems subject to constraints by iteratively improving the solution until constraints are met. It is ideal for this application as it provides a relatively simple solution to an otherwise computationally difficult optimization problem. The method, as applied to this problem, relies on starting from a small value of t_{go}, and successively increasing it until constraints are met. The algorithm for this is as follows:

Step 1) Pick target state for vehicle (target position and velocity, also target acceleration if quadratic or cubic acceleration profile is used, see Section 3.1)

Step 2) Initialize t_{go} to zero, or some minimum value for time of flight.

Step 3) If a polynomial acceleration profile is being used, solve for coefficients of acceleration equation using initial state, target state, and current value of t_{go} (see Section 3.1).

Step 4) Solve for the acceleration profile from time equals zero to t_{go}, and check to see if acceleration constraints are met over the entire profile (see Section 3.2.1).

Step 5) If all constraints are met, STOP. Otherwise, use new value of $t_{go} = t_{go} + increment$, and repeat Steps 3 and 4 until constraints are met.

Note that the minimum time of flight acceleration profile is not necessarily the minimum fuel use profile. For a discussion of estimating fuel use, see Section 3.5.1.

3.2.3 Explicit Solutions for t_{go} by Specifying Other Unknowns

It is possible to explicitly define t_{go} if one or more other unknowns are chosen. An explicit solution for t_{go} based on the Apollo guidance law is provided in [28]. For this solution of t_{go}, a quadratic acceleration profile is used and acceleration in the vertical direction is constrained to be linear. Adding this constraint sets the vertical component of the $\vec{C}_2$ vector to zero, which reduces the system (in the vertical direction) to be equivalent to that found in Section 3.1.1. In the vertical direction, there are now three unknowns (C_0, C_1, and t_{go}), and three equations (see Equations (3.2), (3.3), and (3.4)). Using the vertical components of $\vec{r}_0$, $\vec{v}_0$, $\vec{r}_t$, $\vec{v}_t$, and $\vec{a}_t$ in these equations, it is possible to solve for t_{go},

$$t_{go} = \begin{cases} \dfrac{2v_t + v_o}{a_t} + \sqrt{\left(\dfrac{2v_t + v_o}{a_t}\right)^2 + \dfrac{6(r_0 - r_t)}{a_t}}, & a_t \neq 0 \\ \dfrac{3(r_t - r_0)}{v_0 + 2v_t}, & a_t = 0 \end{cases} . \tag{3.33}$$

While this solution for t_{go} is less computationally intensive than using a search method to find t_{go}, it does not guarantee a feasible trajectory will be generated. This is because the solution does not take into account physical constraints of the system discussed in Section 3.2.1. A good way to use this explicit solution would be to use Equation (3.33) as a starting point for t_{go}, solve for a trajectory, and check in constraints are met. If they

are, use this value of t_{go}; if not, use a search method with this t_{go} as the initial guess to find a feasible trajectory.

3.3 Implementing Guidance

Using the guidance methods presented here so far, it is possible to specify one or more waypoints of a trajectory for a vehicle to follow. If only one waypoint is specified, guidance will calculate a trajectory for the vehicle to get from its initial state to that target. Alternatively, intermediate waypoints can be used to create a general shape for the trajectory, and guidance will fill in the path between waypoints (a two point boundary value problem is solved for each two points). A typical landing trajectory is composed of an approach phase to get the vehicle from its initial state to a point directly above the landing site, and a vertical phase in which the vehicle descends to the landing site [29].

Trajectory commands can be commanded open loop or closed loop. Open loop trajectory commands are referenced from the nominal guidance trajectory determined a priori. Using this setup alone, there is no feedback of state information; however a feedback corrector can be used to adjust open-loop guidance commands. The correction command is closed loop, though the trajectory information is still nominal/open loop. This is Apollo approach guidance, and is typically how guidance is implemented.

Trajectory commands can also be calculated in real time, using closed loop feedback information. To do this, state information is fed back to guidance and is used as the initial state every time guidance re-computes a new trajectory. Guidance can be set to re-compute at any desired interval. The use of this type of guidance allows for real time

update of target waypoints as guidance is continuously re-computing the vehicle's trajectory.

3.3.1 Apollo Approach Guidance

Apollo Approach guidance references commands from a nominal trajectory for the vehicle to follow. This means a complete set of guidance commands (see Section 3.4) are calculated a priori to the flight. These commands are then referenced throughout the flight at the specified guidance rate. The guidance rate may or may not be the same as the controller rate. If the guidance rate is slower than the controller rate, $\vec{\lambda}$ can be updated using $\dot{\vec{\lambda}}$ (see Equation (3.37)). Figure 3-8 shows a simplified block diagram of open loop guidance using the legacy architecture for G&C:

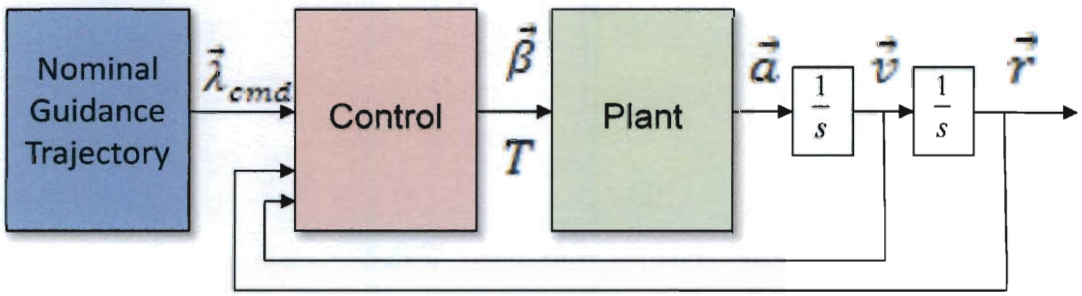

Figure 3-8: Open Loop Guidance

Guidance outputs only nominal trajectory commands; it has no information on the vehicle's current state, therefore it cannot correct for course deviations. This means it is entirely up to the controller to make course corrections.

A closed loop corrector can be added to guidance to correct commands based on current state information. For this, the actual trajectory information is still open loop, but

it is referenced with feedback information to create a correction factor that is added to the nominal command.

Figure 3-9 shows this approach, using a simple proportional derivative (PD) controller to develop a correction factor, $\vec{a}_{corr}$:

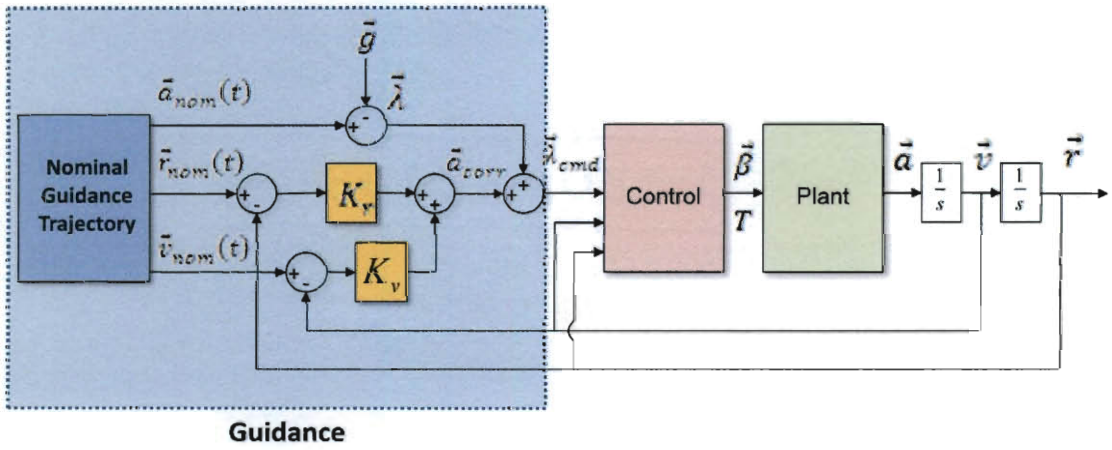

Figure 3-9: Open Loop Guidance with Closed Loop $\vec{\lambda}$ Corrector (Legacy Architecture)

In this figure $\vec{a}_{nom}(t)$, $\vec{v}_{nom}(t)$, and $\vec{r}_{nom}(t)$ are the nominal acceleration, velocity, and position commands referenced from the trajectory developed a priori. The error of the nominal position and velocity commands and their respective actual states are multiplied by corresponding PID gains to create $\vec{a}_{corr}$. The proportional gain is K_r, and the derivative gain is K_v. The nominal acceleration command $\vec{\lambda}$ is combined with the correction factor $\vec{a}_{corr}$ to yield a corrected command $\vec{\lambda}_{cmd}$.

The proportional and derivative gains can be chosen using traditional control methods. Alternatively, time varying gains can be derived from the acceleration profile. This is how Apollo chose gains; using Apollo's quadratic acceleration profile, K_r is $\frac{12}{t_{go}^2}$

and K_v is $\frac{6}{t_{go}}$ [5]. The problem with this method is that these gains are inversely proportional to t_{go}, so as the vehicle reaches its target, and t_{go} goes to zero, the gains will approach infinity. A simple workaround for this problem is to enforce a minimum value of t_{go}, such that the gain does not go above a certain value.

3.3.2 Real Time Guidance

A disadvantage of Apollo approach guidance is the inability to change the waypoints/targets of the vehicle once it is in flight, as it references a set of nominal commands. By passing back state information to guidance, it is possible to compute trajectories in real time, which allows waypoints to be changed during flight. Figure 3-10 shows the setup for this:

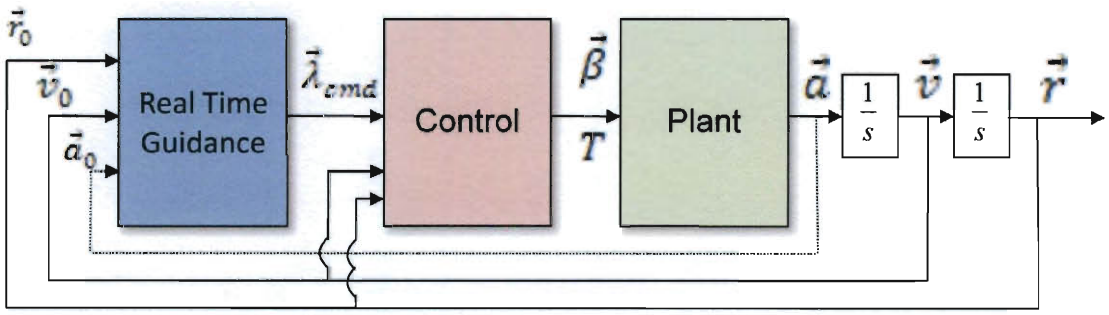

Figure 3-10: Real Time Guidance

Current position, velocity, and acceleration data that is passed back to guidance becomes the initial state for the trajectory boundary value problem (acceleration data may or may not be needed depending on the acceleration profile used).

Real time guidance cannot re-compute new trajectories at the same rate of the controller, because of the fact that it is computationally intensive to solve a trajectory

boundary value problem if a search method is being used to find t_{go}. Instead, new trajectories must be computed at a much slower rate than the controller.

The controller still needs a command every cycle though. Equation (3.37) could be used to update $\vec{\lambda}$ using $\dot{\vec{\lambda}}$, but this only provides an estimate of $\vec{\lambda}$, at best. Instead, a different method of splitting guidance into two subsystems can be used to deliver $\vec{\lambda}$ commands at the same rate of the controller:

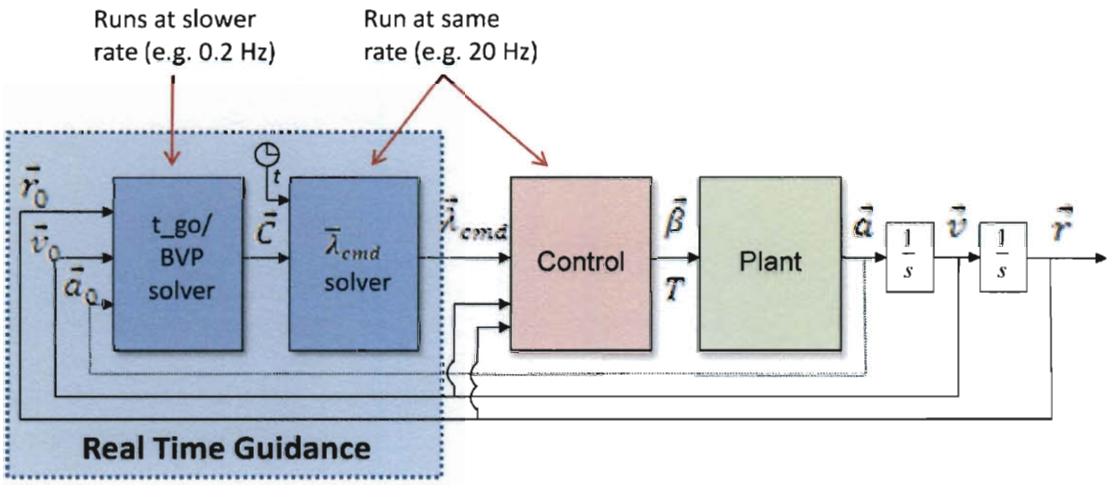

Figure 3-11: Real Time Guidance Using Two Subsystems

Instead of calculating a single $\vec{\lambda}$ every time a trajectory is computed, this method allows a $\vec{\lambda}$ command to be computed every control cycle, even though the rest of guidance is running a slower rate. To do this, the trajectory boundary value problem only solves for a set of coefficients $\vec{C}$ for the acceleration profile, and does not go so far as to solve for a specific $\vec{\lambda}$ command. These coefficients can then be plugged into the polynomial acceleration equation to solve for a $\vec{\lambda}$ command for any time t.

3.3.2.1 High Fidelity Trajectory Following Using Real Time Guidance

For a situation where the vehicle must follow a specific course rather than plot its own course between a few waypoints, open loop guidance with a $\vec{\lambda}$ corrector has traditionally been used. For this case the specific trajectory is determined and loaded into guidance a priori, and $\vec{\lambda}$ corrector makes corrections on open loop guidance commands during the flight.

An alternative to this method is to use waypoint following real time guidance, without a $\vec{\lambda}$ corrector. This method relies on re-computing a trajectory every guidance loop, rather than using a corrector to update the open loop trajectory command.

The configuration for this is not that different than the real time guidance method described above. Instead of a few waypoints being uploaded to guidance a priori, the entire trajectory is discretized and each trajectory point is uploaded in real time. The discretized trajectory points act just as regular waypoints would, except now there is a new target waypoint every time guidance is calculated.

For regular real time guidance, a few waypoints are known (and are usually spread far apart), and guidance is free to determine both the trajectory and time of flight to get to the next waypoint. To facilitate high fidelity trajectory following, the time of flight for each guidance calculation is prescribed in order for the vehicle to be on the correct path at the correct time. To implement this, rather than using a search method to find t_{go}, t_{go} is set to be the time in between guidance calculations. This ensures that guidance gets to the next discretized trajectory point at the right time. Because a search

method is not used, the guidance calculation is not nearly as computationally intensive, and can be done much faster.

Guidance is still free to calculate a trajectory to get to the next discretized point, but the distance and time in between points is so small that it follows the prescribed path almost exactly.

As is sometimes the case when t_{go} is prescribed, there is a chance guidance will generate an acceleration profile that is not feasible. This problem can be effectively avoided by insuring a priori that the acceleration profile for the prescribed trajectory meets the acceleration constraints of the vehicle.

3.4 Generating Guidance Commands

Once an implementation method has been selected, t_{go} has been computed, and the coefficients of the acceleration profile found (or t_{burn} and t_{coast} have been computed for a step thrust profile), it is a simple matter to generate a trajectory command that can be used by steering. As discussed in Chapter 1, the legacy GNC architecture requires that guidance pass a command for $\|\vec{a}_T\|$, $\vec{\lambda}$, $\dot{\vec{\lambda}}$, t_λ, and ϕ_{cmd}^{roll}.

The acceleration the vehicle must provide in order to follow the desired trajectory, $\vec{a}_T$, is equal to

$$\vec{a}_T = \vec{a} - \vec{g}\,. \tag{3.34}$$

Where $\vec{a}$ is the kinematic acceleration of the vehicle as dictated by the guidance acceleration profile, and $\vec{g}$ is the gravity vector acting on the vehicle (a *negative* acceleration).

The magnitude of commanded acceleration, $\|\vec{a}_T\|$, is a unscaled throttle command. It is scaled by mass in the controller to determine thrust,

$$F^{thrust} = m^t \|\vec{a}_T\|. \tag{3.35}$$

The commanded acceleration direction command, $\vec{\lambda}$, is simply the unit vector of the $\vec{a}_T$ command,

$$\vec{\lambda} = \frac{\vec{a}_T}{\|\vec{a}_T\|}. \tag{3.36}$$

This is converted by the steering subroutine into a quaternion (body state) command and is referenced with the actual body state in the controller to generate the gimbal angle command for the engine.

The time derivative of $\vec{\lambda}$, $\dot{\vec{\lambda}}$, is converted by the steering subroutine into a rotation (body) rate command which is referenced to the actual body rate command in the controller to generate the gimbal angle command for the engine. $\dot{\vec{\lambda}}$ can also be used to update the initial $\vec{\lambda}$ command at time *t*,

$$\vec{\lambda}_{cmd}(t) = \vec{\lambda} + \dot{\vec{\lambda}}(t - t_\lambda). \tag{3.37}$$

The time associated with the value of $\vec{\lambda}$ is t_λ. If guidance generates a $\vec{\lambda}$ for every $t = t_\lambda$, this update is not necessary.

A roll command (rotation around the vehicle's vertical axis) is denoted by ϕ_{cmd}^{roll}. It is normally set to zero, unless it is required that the vehicle be oriented in a particular direction.

3.5 Vehicle Footprint Analysis

Due to the nature of terrain on the Moon and Mars, lander missions face a high probability of encountering austere landing sites. Current technology provides an incomplete knowledge of the terrain of any specific landing site chosen, so it is possible that a predetermined landing site may not be feasible [30]. It is important for a vehicle to have a complete understanding of the possible area in which it can land—its footprint—in case is needs to divert to an alternate landing site [31]. Autonomous landers must be capable discerning the acceptability of potential landing sites, and re-computing a trajectory to reach the best option. Figure 3-12 portrays this in the scope of the entire descent profile for an autonomous lander on Mars.

A method for generating a terrain map using Lidar and Radar from which a lander can choose a safe landing site to avoid terrain hazards is described in [28]. But before this terrain map can be used to determine a landing site, the lander must have an understanding of how far down or cross range it can travel without running out of propellant. The lander must be able to compute this footprint for any given state in order to know where to check for safe landing terrain and re-compute a trajectory.

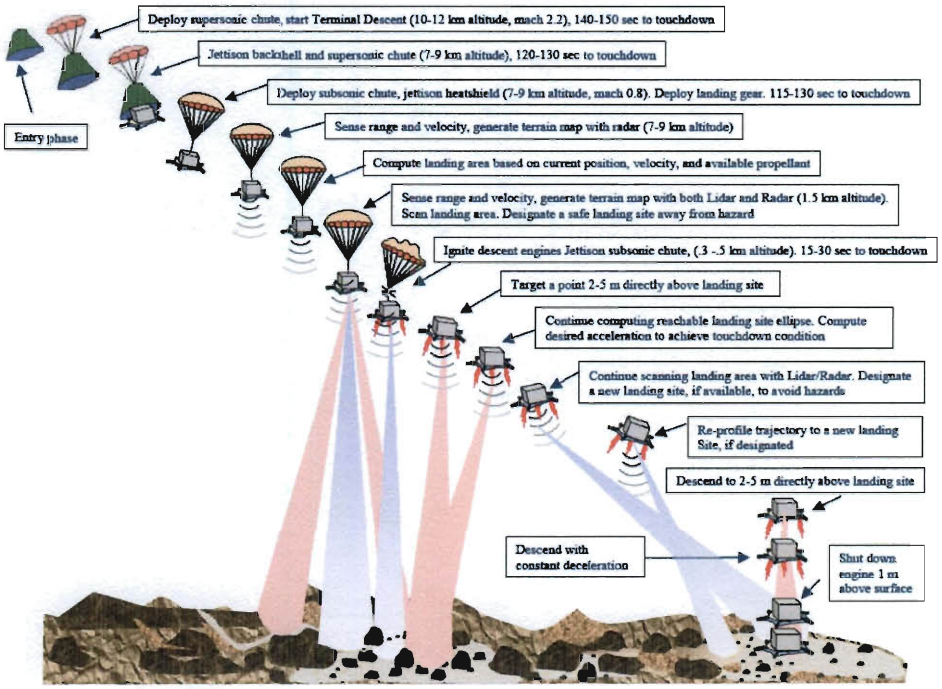

Figure 3-12: Landing Scenario Using Footprint Estimation for Planetary Lander [28]

Determination of a footprint is a complicated problem that can be computationally intensive depending on the method used. To make the matter more challenging, the footprint must be able to be calculated onboard and in real time, because of constantly changing initial conditions that are difficult to predict a priori [32]. It may not be feasible for vehicle's flight computer to calculate a highly accurate footprint in real time given computer and time constraints. The goal of the footprint software, therefore, is to rapidly generate a footprint for the vehicle that is as accurate as possible.

The problem can be examined using two approaches: using simple ballistics and using a guidance-based method [33]. For the ballistics approach, the JPL team applies the theory of elementary ballistics to the problem, using the assumption that the vehicle's trajectory is ballistic. The vehicle is in free fall, and at the last possible moment the

engine is initiated to slow the vehicle to a soft landing. While this scenario may be essentially fuel optimal, as discussed in Section 3.1.4, it is not realistic as guidance will inevitably command thrust maneuvers during the assumed ballistic phase. This fact is shown in their results: the ballistic approach consistently over predicts the more accurate guidance-based method. Although the ballistic approach is computationally fast, it only yields a very rough estimate of a vehicle's footprint [33].

The alternative to the ballistic method is a guidance-based method, which is the method examined herein. This method relies on re-running the guidance algorithm for trajectories of increasing distance down range. This process continues until the algorithm determines that the distance downrange is unfeasible. This distance becomes the edge of the footprint. The feasibility criterion for this method is fuel use: if the algorithm estimates that a trajectory uses more fuel than is available, that trajectory is deemed unfeasible.

3.5.1 Estimating Propellant Use

Estimating the propellant usage of a potential trajectory is necessary in order to determine the feasibility of that trajectory. A trajectory may be feasible given the acceleration constraints of a vehicle, but if it uses more propellant than the vehicle has, it cannot be flown. The outer boundary of a footprint, therefore, is determined largely by how far the vehicle can go before running out of propellant.

Guidance develops a trajectory by determining the acceleration profile for the course of the vehicle's flight. Given parameters of the vehicle and engine, it is a simple task to derive the estimated amount of propellant used to fly a trajectory.

First, the amount of thrust required to fly a trajectory can be approximated using the Newtonian equation,

$$F^{thrust}(t) = m^t(t)\|\vec{a}_T(t)\|. \tag{3.38}$$

Next, Equation 1.7 from [34] is used to estimate the amount of propellant used,

$$I_{sp} = \frac{F}{\dot{m}_{prop} g_0}. \tag{3.39}$$

Where I_{sp} is the engine's specific impulse (s), F is thrust (N), $\dot{m}_{prop}$ is the propellant mass flow rate (kg/s), and g_0 is 9.807 m/s^2. This equation can be rearranged and integrated in time to find the mass of propellant consumed,

$$m_{propUsed} = \int \dot{m}_{prop}\, dt = \int \frac{F^{thrust}(t)}{I_{sp} g_0} dt = \int \frac{m^t(t)\|\vec{a}_T(t)\|}{I_{sp} g_0} dt. \tag{3.40}$$

This integration can be done numerically for footprint analysis using the following algorithm:

For t from 0 to t_{go} at a step of Δt:

Step 1) Calculate norm of commanded acceleration, $\|\vec{a}_T(t)\|$ (using polynomial acceleration profile, for example)

Step 2) Calculate the thrust required using Equation (3.38)

Step 3) Calculate the propellant used for current step:

$$m_{propUsed} = \frac{m^t(t)\|\vec{a}_T(t)\|}{I_{sp} g_0} \Delta t \tag{3.41}$$

Step 4) Update total mass estimate: $m^t_{est}(t) = m^t_{est}(t - \Delta t) - m_{propUsed}$

End

3.5.2 Complete Algorithm for Guidance Based Footprint Analysis

Using this method to estimate propellant, it is possible to estimate the maximum reachable range of the vehicle. Figure 3-13 shows the top-level algorithm in pseudocode for determining maximum reachable range in one direction.

```
(1) WHILE (lowest_estimated_propellant _use) < (propellant_available)
    (2) Increment Footprint Boundary (r_downrange) = (r_downrange) + (step);
    (3) FOR (t_go) from (min_t_go) to (max_t_go)
        (4) Generate C coefficients for trajectory acceleration profile (trajectory);
        (5) IF (trajectory) satisfies (acceleration_constraints)
            (6) FOR (t) from (0) to (t_go)
                Estimate Propellant Use (estimated_propellant _use);
            END
            (7) IF (estimated_propellant _use) < (lowest_estimated_propellant _use)
                (lowest_estimated_propellant _use) = (estimated_propellant _use) ;
            END
        END
    END
END
```

Figure 3-13: Algorithm for Determining Maximum Reachable Range in One Direction

The algorithm in Figure 3-13 works by calculating trajectories for increasingly down range values of target position, and checking for feasibility. Step (1) is a loop over the entire algorithm which continues as long as feasible trajectories are being generated. Step (2) increments the footprint boundary at the desired step ($r_{downrange}$ is the target position of the footprint boundary in the direction of travel). Step (3) is a loop that is run for a range of time of flight values t_{go}. A range of t_{go}'s should be examined because the maximum reachable range is sensitive to t_{go} and can drastically change given on a longer or shorter t_{go} [33]. Alternatively, t_{go} can be explicated specified or otherwise chosen, but doing this may not produce trajectories indicative of the vehicle's maximum reachable range. Step (4) generates the C coefficients for a trajectory acceleration profile

(see Section 3.1), given an initial condition, current loop value of t_{go}, and a target condition containing current loop value of $r_{downrange}$. Step (5) is a conditional statement that checks if the generated trajectory meets physical acceleration constraints (see Section 3.2.1). If constraints are met, Step (6) estimates propellant use for the trajectory, using the algorithm from Section 3.5.1. Step (7) is a conditional used to store the trajectory with the lowest estimated propellant use so it can be checked by the conditional of the outer loop of Step (1). Once the conditional of Step (1) is no longer satisfied (meaning estimated propellant use is greater than propellant available), the loop stops and the footprint boundary is the current value of $r_{downrange}$.

If the vehicle has zero initial velocity, it is sufficient to run this algorithm once as the boundary of the footprint is circular with a radius of $r_{downrange}$. If the vehicle has an initial velocity, the footprint will be elliptical, and the algorithm must be run multiple times for various directions (down range, cross range, back range, etc.) in order to estimate the boundary of the ellipse.

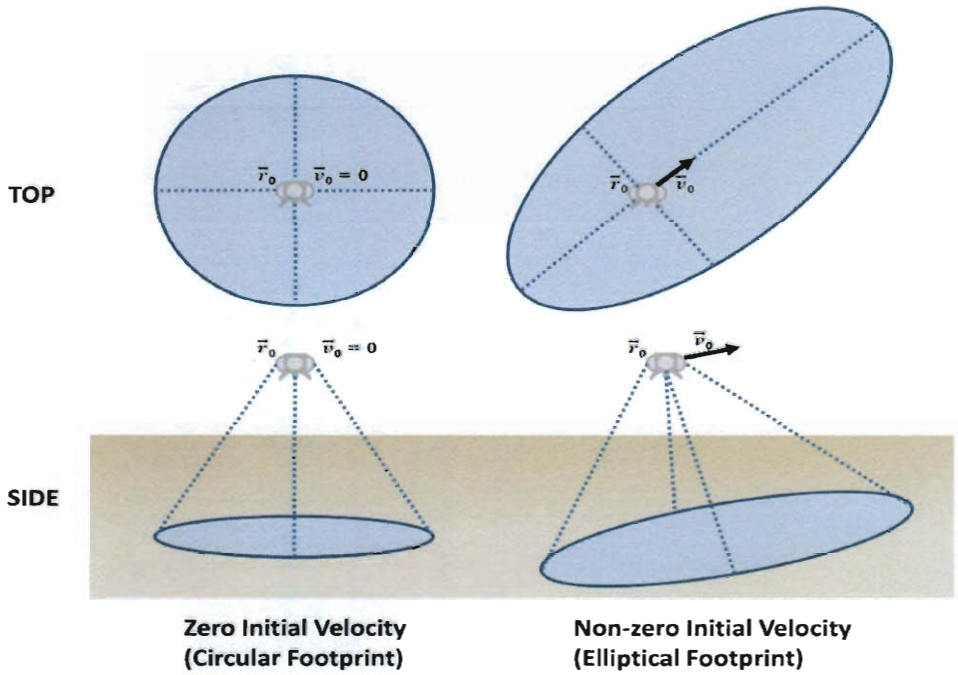

Figure 3-14: Illustration of Circular and Elliptical Footprints

This method is computationally intensive due to the number of loops that must be run to converge on a solution. But it is relatively simple to implement, and is sufficient to examine how physical parameters and initial conditions affect potential vehicle footprints.

4 Results

In order to discern the effectiveness of the methods detailed in Chapter 3, various trajectories are tested using each method and the results of each compared. The methods can be implemented with or without vehicle dynamics (detailed in Chapter 2). It is most useful, however, to implement the methods within a complete GNC architecture including dynamics, in order to develop a more complete understanding of the robustness of each algorithm.

For the results presented herein, the controller (using proportional plus derivative control on rotational state of vehicle to determine gimbal angle [7]) and steering methods are held constant. A legacy GNC architecture is used; guidance outputs an acceleration command in the form of $\vec{\lambda}$ and thrust magnitude, and steering and control convert it to a gimbal angle and thrust command.

For the plant, six degree of freedom (6DOF) vehicle dynamics based on the equations of motion derived in Chapter 2 are used. A vehicle with four propellant slosh masses and a single gimbaled engine is used. All other vehicle dimensions, masses, and characteristics are arbitrary.

In order to more easily compare fuel use of each method, the fuel use for each test of each trajectory is normalized with respect to the first method examined for that trajectory. This is referred to as relative propellant use.

4.1 Comparison of Acceleration Profiles

This section compares the results given by the three polynomial acceleration profiles derived in Section 3.1 (cubic, quadratic, and linear), as well as the constant acceleration profile derived in Section 3.1.4. Two maneuvers are examined: a pure vertical descent, and a pure horizontal translation. For these maneuvers the propellant tanks are about a third full.

4.1.1 Vertical Descent Maneuver

The simplest relevant test case is a vertical descent maneuver, whereupon the vehicle starts at an initial altitude with zero velocity, and descends vertically to ground level with a final velocity of zero (See Figure 4-1).

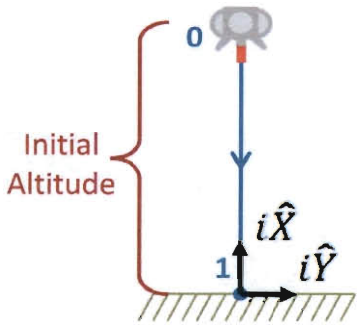

Figure 4-1: Vertical Descent Maneuver

This maneuver allows for direct comparison of relevant characteristics of the various acceleration profiles derived in Chapter 3. These acceleration profiles are: cubic polynomial acceleration, quadratic polynomial acceleration, linear polynomial acceleration, and constant min-max acceleration.

The maneuver is first examined kinematically, for which the vehicle is assumed to be a point mass. No steering or control is necessary because vehicle dynamics are not

taken into account. Each guidance algorithm is used to generate an ideal acceleration profile that is compared to the results of other profiles.

For these results, the vehicle starts at an altitude of a 100 m with zero velocity and descends, reaching zero velocity as it touches ground level. The kinematic acceleration profile generated by each method is shown below:

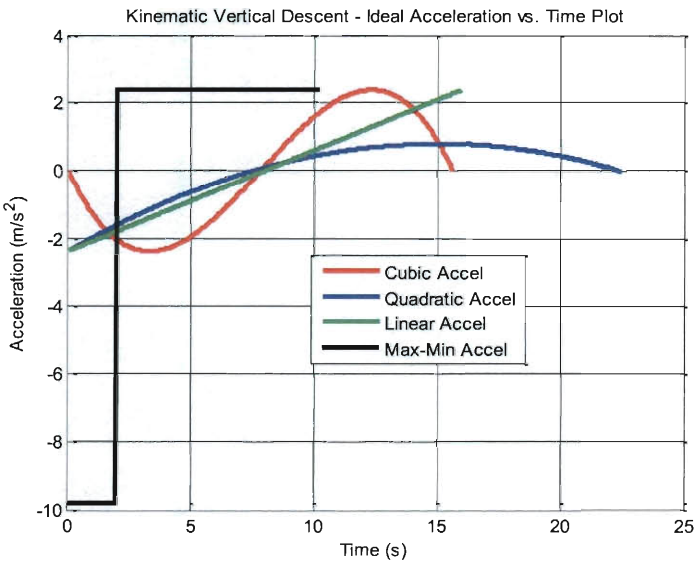

Figure 4-2: Ideal Acceleration vs. Time for Kinematic Vertical Descent

The kinematic velocity profile for each method is shown below:

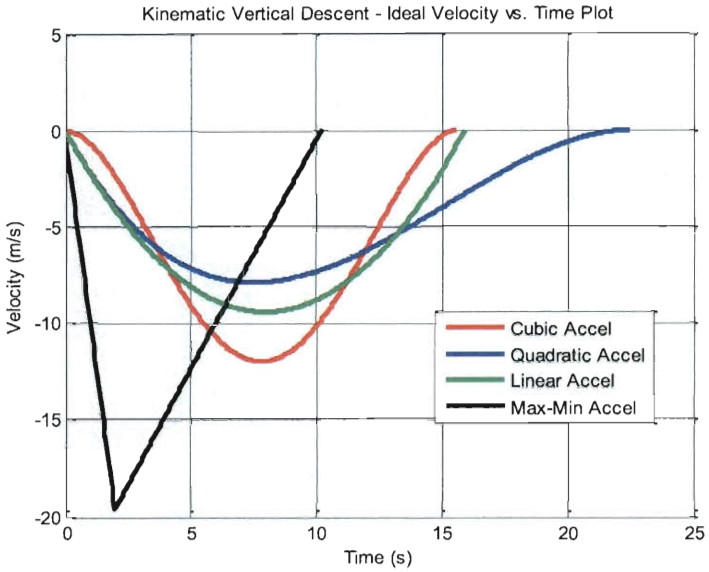

Figure 4-3: Ideal Velocity vs. Time for Kinematic Vertical Descent

Finally, the kinematic position profile for each method is shown below:

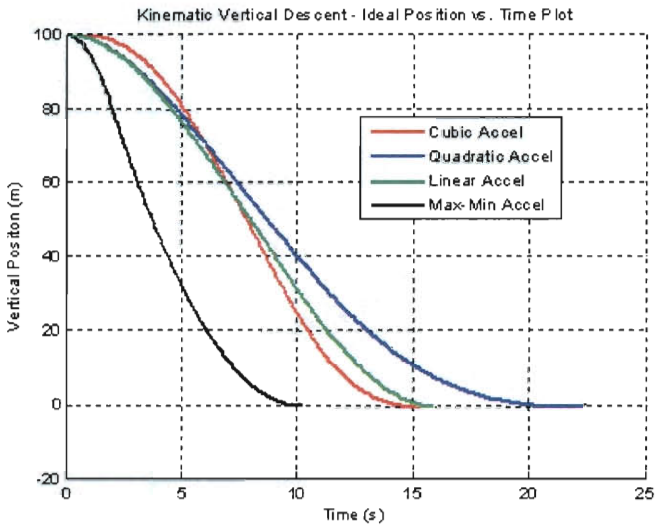

Figure 4-4: Ideal Position vs. Time for Kinematic Vertical Descent

The relevant flight parameters for each method are summarized below:

Method	Time of Flight (s)	Max. Downward Velocity (m/s)	Relative Prop Use
Linear Accel. Profile	15.90	9.43	1.00
Quadratic Accel Profile	22.45	7.92	1.41
Cubic Accel. Profile	15.60	12.02	0.98
Min-Max Accel Profile	10.20	19.62	0.64

Table 4-1: Summary of Kinematic Vertical Descent Results

As postulated in Chapter 3, the min-max acceleration profile has better fuel use than any of the polynomial acceleration profiles. In fact, its relative fuel use is only 64% of the linear acceleration profile used by the Apollo Moon Lander.

The vertical descent maneuver is also examined using 6DOF vehicle dynamics. To do this, the entire GNC architecture described at the start of Chapter 4 is implemented. For this maneuver, guidance outputs acceleration commands open loop.

As the maneuver is purely vertical in the inertial X ($i\hat{X}$) direction, only motion in this direction is plotted. Acceleration, velocity, and position in the horizontal plane is zero. The acceleration vs. time trace using 6DOF dynamics for each method is shown below:

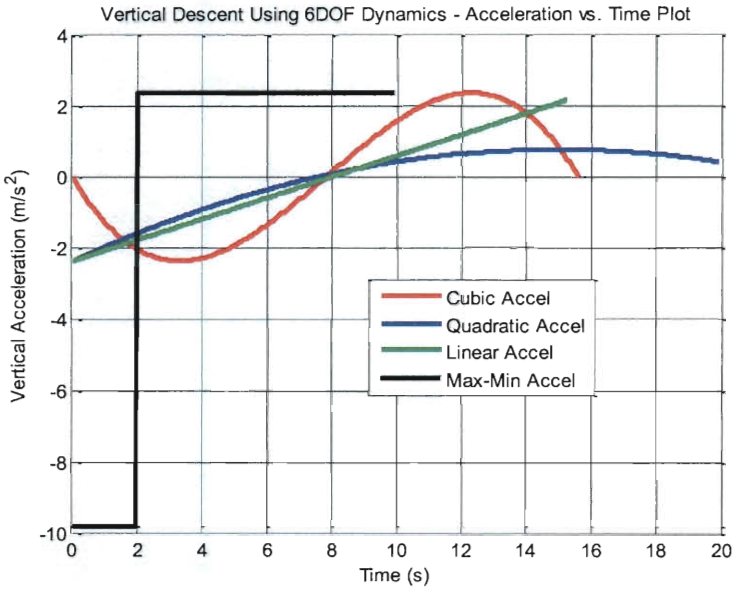

Figure 4-5: Acceleration vs. Time for Vertical Descent Using 6DOF Dynamics

The velocity vs. time trace using 6DOF dynamics for each method is shown below:

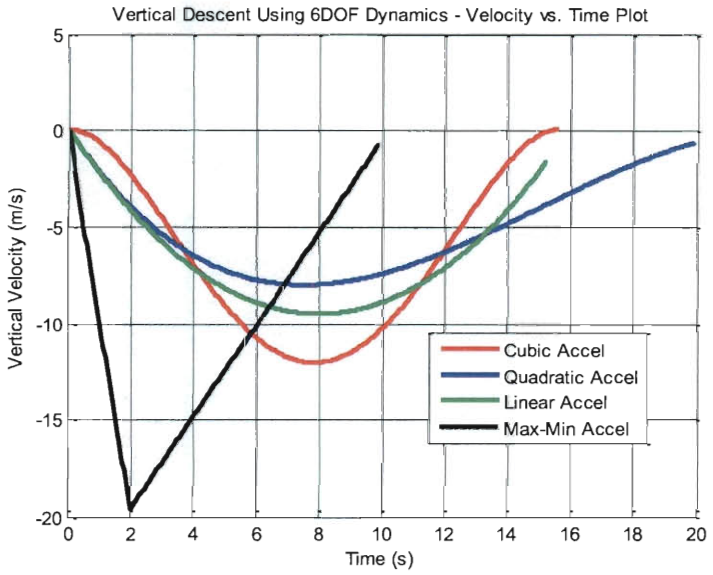

Figure 4-6: Velocity vs. Time for Vertical Descent Using 6DOF Dynamics

The position vs. time trace using 6DOF dynamics for each method is shown below:

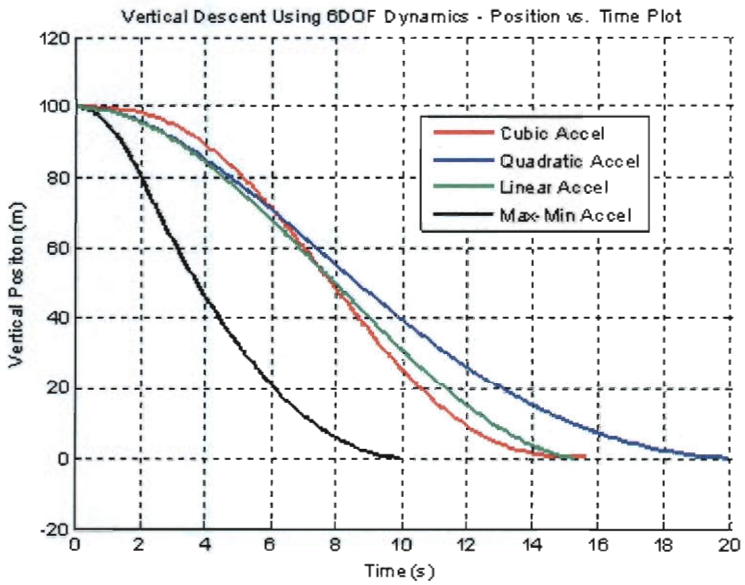

Figure 4-7: Position vs. Time for Vertical Descent Using 6DOF Dynamics

A summary of the relevant flight characteristics is shown below:

Method	**Time of Flight (s)**	**Max. Downward Velocity (m/s)**	**Landing Velocity (m/s)**	**Relative Prop Use**
Linear Accel. Profile	15.22	9.47	-1.60	1.00
Quadratic Accel Profile	19.87	7.96	-0.63	1.30
Cubic Accel. Profile	15.60	12.00	0.00	1.02
Min-Max Accel Profile	9.91	19.62	-0.70	0.66

Table 4-2: Summary of Vertical Descent Results Using 6DOF Dynamics

The results for the vertical maneuver using 6DOF dynamics are very similar to the kinematic solution. This is not surprising because the thrust command is fed forward through the controller from the acceleration command of guidance, and there is no gimbal

command necessary. As such, the role of steering and control is negligible. The results can even more closely correlate to the kinematic solution if guidance and control rates are adjusted. There is no propellant slosh displacement for this maneuver, which is not surprising as the maneuver is only in the inertial vertical direction, and propellant slosh is constrained to zero in the vertical direction.

Kinematic results sufficiently correlate to 6DOF dynamics results for this maneuver and in general, so from here forward, only 6DOF dynamics are used to examine trajectories. While kinematic solutions present ideal trajectories, they are less useful because they do not take into account specific vehicle dynamics such as propellant slosh and engine gimbal.

4.1.2 Horizontal Translation Maneuver

The next test case is a pure horizontal maneuver. For this, the vehicle starts at an initial altitude with zero velocity, and travels a specified distance down range (100 meters for this test case) while maintaining altitude and reaching zero velocity at the target.

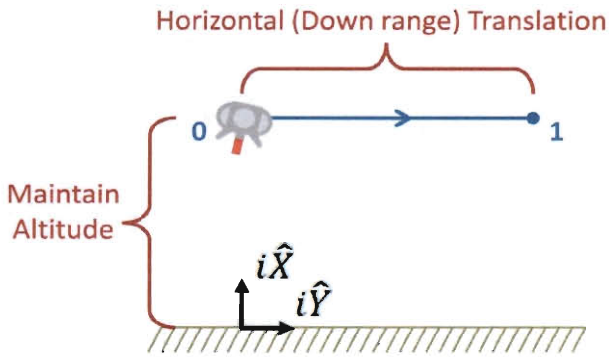

Figure 4-8: Horizontal Translation Maneuver

As with the vertical descent maneuver, each polynomial acceleration profile is examined, as well as the max-max constant acceleration profile derived in Section 3.1.4.2. Though the simulation takes into account three dimensions, the maneuver only occurs in the inertial Y ($i\hat{Y}$) direction. As there are only displacements in this direction, only displacements in $i\hat{Y}$ and rotations about $i\hat{Z}$ are plotted below. Displacements and rotations in other directions are zero.

The acceleration of the vehicle in $i\hat{Y}$ for each method is shown below:

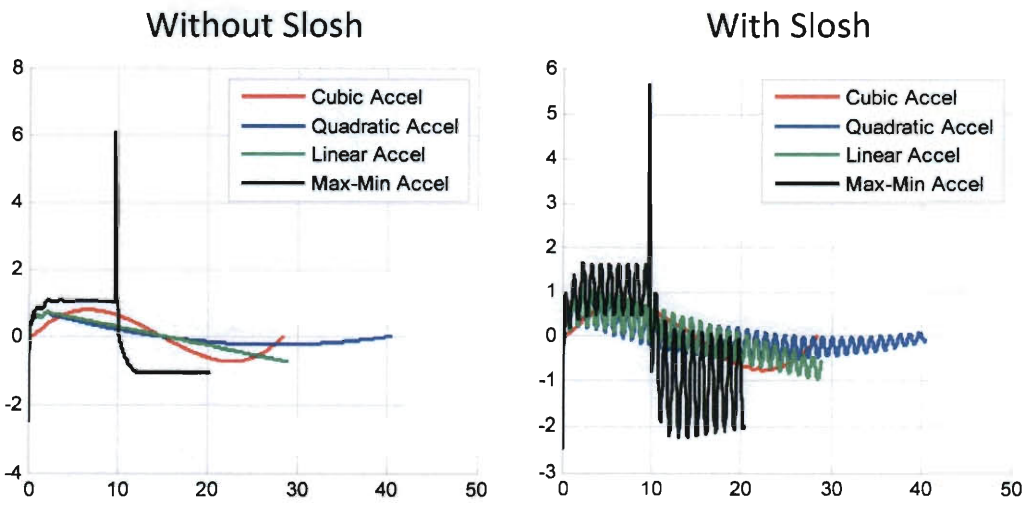

Figure 4-9: Acceleration vs. Time for Horizontal Translation Maneuver

The oscillating nature of each acceleration trace on the right is a result of propellant slosh. The cubic acceleration profile is the smoothest because it exhibits the least sloshing, as can be seen in Table 4-3.

The velocity of the vehicle in $i\hat{Y}$ for each method is shown below:

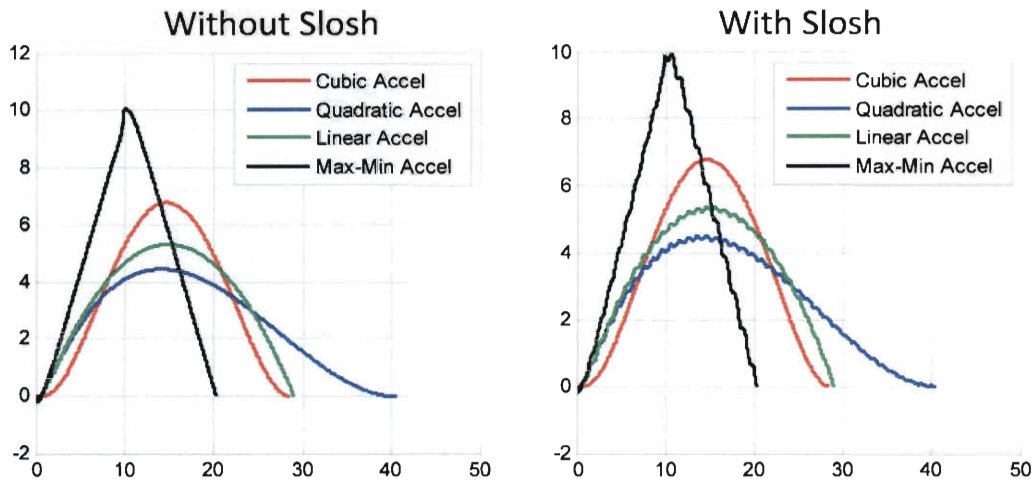

Figure 4-10: Velocity vs. Time for Horizontal Translation Maneuver

The position of the vehicle in $i\hat{Y}$ for each method is shown below:

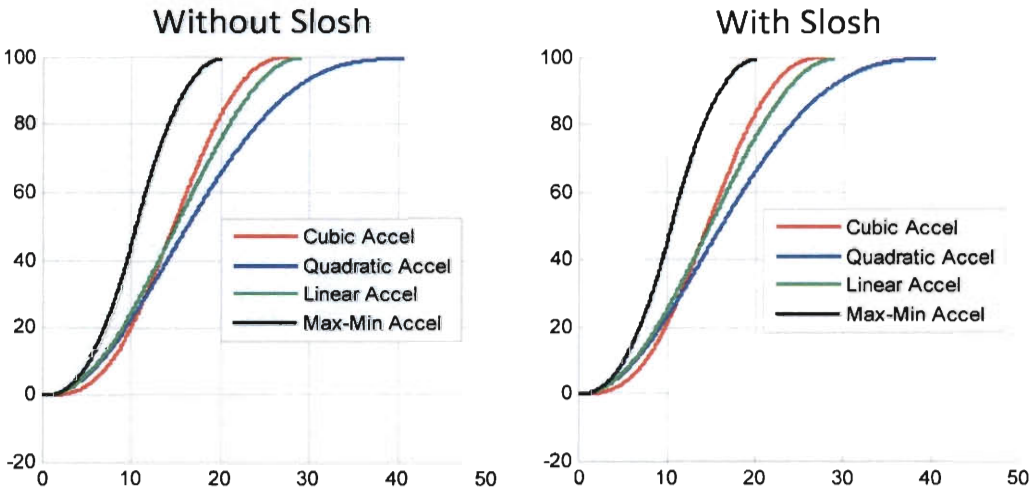

Figure 4-11: Position vs. Time for Horizontal Translation Maneuver

Finally, it's useful to examine the yaw attitude (tilt) of the vehicle about the $i\hat{Z}$ direction (φ_z):

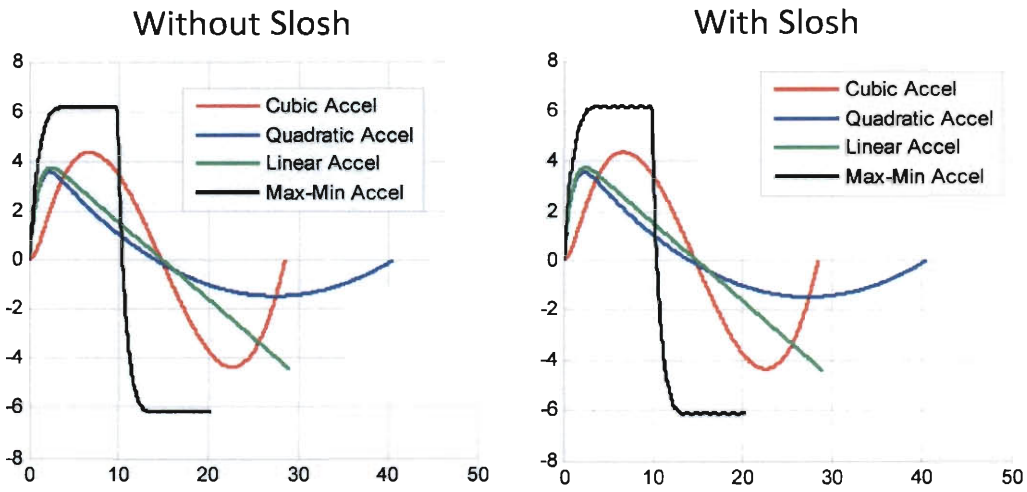

Figure 4-12: Yaw (φ_z) vs. Time for Horizontal Translation Maneuver

A summary of the results for this maneuver are shown in Table 4-3:

Method	**Time of Flight (s)**	**Max. Velocity (m/s)**	**Max Slosh Displacement (m)**	**Max Yaw Angle (degrees)**	**Max Gimbal Angle (degrees)**	**Relative Propellant Use**
Linear Accel. Profile	29.00	5.37	2.74E-2	4.44	3.83	1.00
Quadratic Accel Profile	40.50	4.51	2.74E-2	3.59	3.82	1.36
Cubic Accel. Profile	28.46	6.78	1.75E-3	4.37	0.13	0.98
Max-Max Accel Profile	20.30	9.99	9.00E-2	6.20	12.13	0.72

Table 4-3: Summary of Results for Horizontal Maneuver Using 6DOF Dynamics

As with the vertical maneuver, the constant (max-max) acceleration profile provides the best fuel use and shortest time of flight. However, this is achieved at the cost of the highest yaw angle, gimbal angle, and slosh displacement (as a result of having the highest rotation rate and acceleration), which makes it the most aggressive profile and therefore most likely to cause the vehicle to go unstable.

The cubic acceleration profile is a good alternative to the max-max profile. It has good fuel use and time of flight, but by far the least gimbal angle and slosh displacement. This is due to the fact that initial and target acceleration boundary constraints are imposed with a cubic acceleration profile. Because there is never an instantaneous change (discontinuity) in acceleration, slosh is minimized.

4.2 Comparison of Guidance Implementation Methods

In this section, two guidance implementation methods are examined: Apollo approach 'lambda control,' and real time guidance. For real time guidance implementation, three acceleration profiles are examined: cubic, quadratic, and linear.

Three trajectories are examined: a jump trajectory with and without wind, a high fidelity trajectory, and a moon descent trajectory.

4.2.1 Jump Trajectory – No Wind

The first trajectory examined is a "jump" trajectory, which refers to a simple flight profile where the vehicle starts at ground level and follows three waypoints to get to its target state. The three waypoints comprise a jump maneuver consisting of a vertical ascent phase, horizontal translation/descent phase, and vertical descent phase. These phases are representative of the type of maneuvering a vehicle would do in a planetary landing application. For this maneuver the propellant tanks are about a third full.

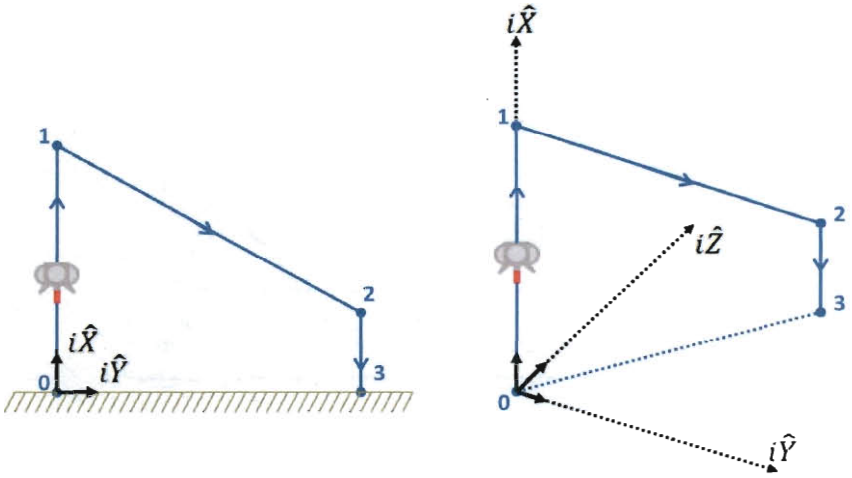

Figure 4-13: Jump Trajectory

For the jump trajectory analyzed, the first phase consists of a vertical ascent to an altitude of 50 meters ($i\hat{X}$ direction), followed by a 10 second hover. The second phase (waypoint one to two) consists of a simultaneous translation to 75 meters in the $i\hat{Y}$ direction, -50 meters in the $i\hat{Z}$ direction, and a descent to 10 meters in the $i\hat{X}$ direction. This is followed by another 10 second hover. The final phase (waypoint two to three) consists of a vertical descent to ground level ($i\hat{X} = 0$).

For the lambda corrector method, AKA lambda control (discussed in Section 3.3.1), guidance is used to calculate a nominal kinematic trajectory, which is fed to the controller in flight. 6DOF results for position vs. time of the vehicle using lambda control and real time guidance (using acceleration profiles from Section 3.1) are shown in Figure 4-14.

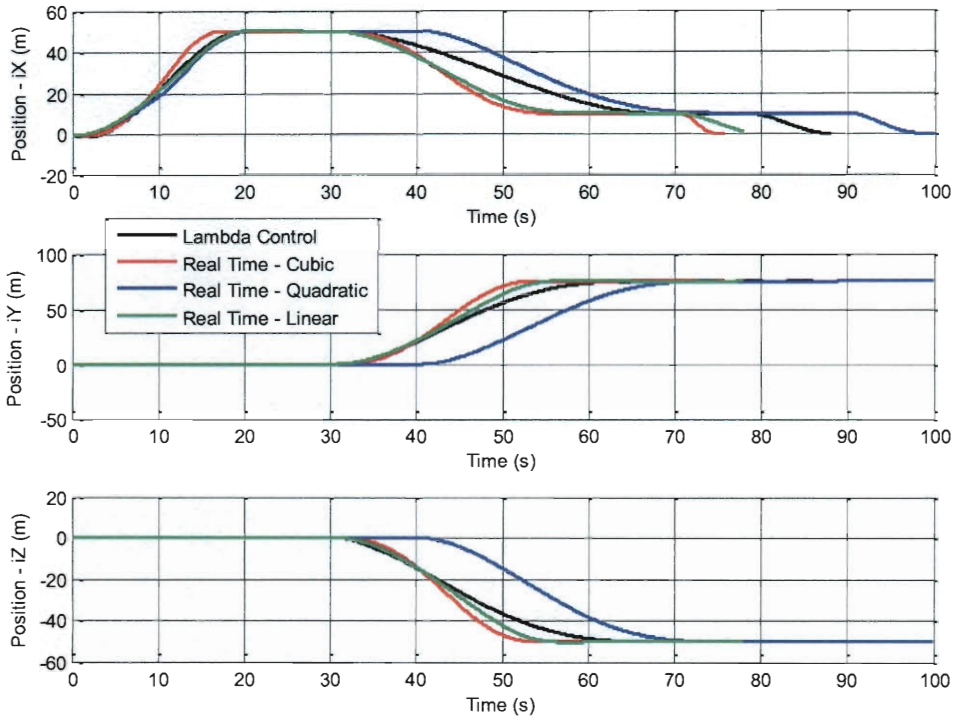

Figure 4-14: Position of Vehicle vs. Time for Jump Trajectory

The real time methods using polynomial acceleration provide 6DOF trajectories that are similarly close to their nominal kinematic counterparts. But for this simple trajectory, position error is not relevant. Relevant flight characteristics are time of flight, propellant use, and landing magnitude error, which are shown in Table 4-4.

Method	Time of Flight (s)	Relative Propellant Use	Landing Error Magnitude (m)
Lambda Control	88.00	1.00	0.05
Linear Accel. Profile	77.85	0.90	0.03
Quadratic Accel Profile	99.90	1.11	0.04
Cubic Accel. Profile	75.65	0.88	0.01

Table 4-4: Summary of Flight Characteristics for Jump Trajectory – No Wind

These characteristics provide a broad overview of the relative merits of each guidance method. In reality, it is useful to examine how the attitude (rotation) of the vehicle changes in flight. The more a vehicle rotates, the more chance there is of it going unstable, and the more the engine must gimbal to compensate. In addition, vehicle rotation contributes significantly to propellant slosh, which can further cause instabilities in the vehicle. For these reasons, it is ideal for guidance to limit how much a vehicle has to rotate in order to follow a trajectory.

The attitude of the vehicle about each inertial axis with respect to time is shown in

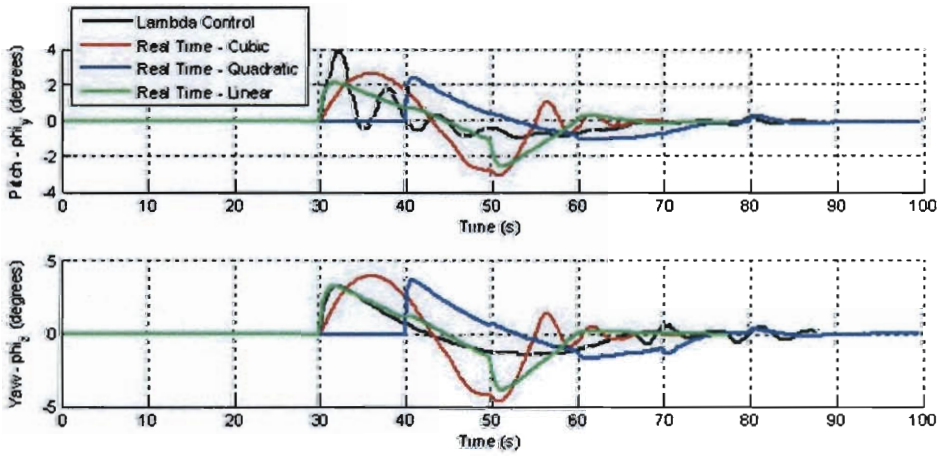

Figure 4-15. Note that for this simulation there is no roll command or control, nor is the roll attitude relevant to the successful operation of the vehicle. The roll that occurs is a result of coupled gimbal of the engine about the $b\hat{y}$ and $b\hat{z}$ axes.

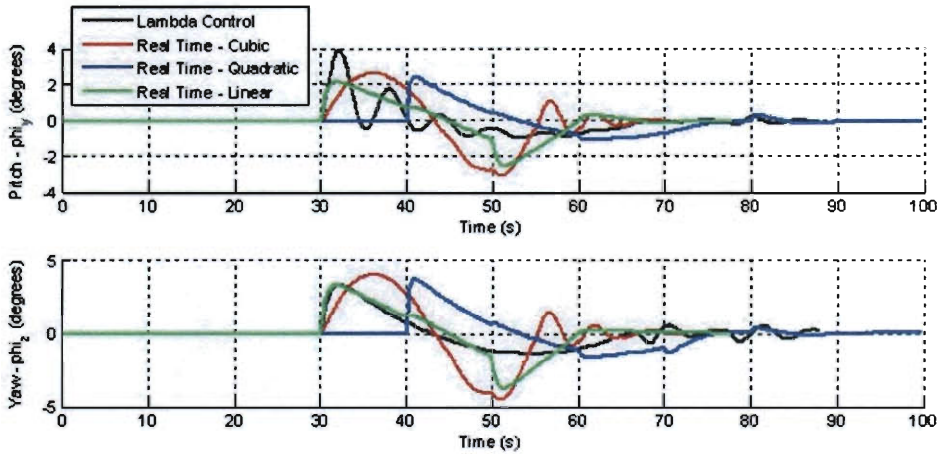

Figure 4-15: Jump Trajectory Attitude of Vehicle vs. Time Plot

The maximum attitude values for each axis and each method are shown in Table 4-5. In addition, the maximum propellant slosh that occurs during flight is shown for each method.

Method	Max +/- Attitude (degrees)		Max Slosh - Any Direction (m)
	Pitch	Yaw	
Lambda Control	3.91	3.31	2.39E-2
Linear Profile	2.53	3.74	4.05E-2
Quadratic Profile	2.44	3.66	7.84E-2
Cubic Profile	3.00	4.56	1.74E-2

Table 4-5: Maximum Flight Characteristics for Jump Trajectory – No Wind

These results show that the most propellant slosh does not correlate directly with large values of attitude. Large slosh does seem to correlate with large values of angular velocity of the body, which makes sense as this parameter shows up in the equation of

motion for slosh. Figure 4-16 shows the angular velocity of the vehicle in the pitch and yaw directions. This is not the only parameter that affects slosh though; the equation of motion for slosh shows several other variables that affect how the propellant is displaced.

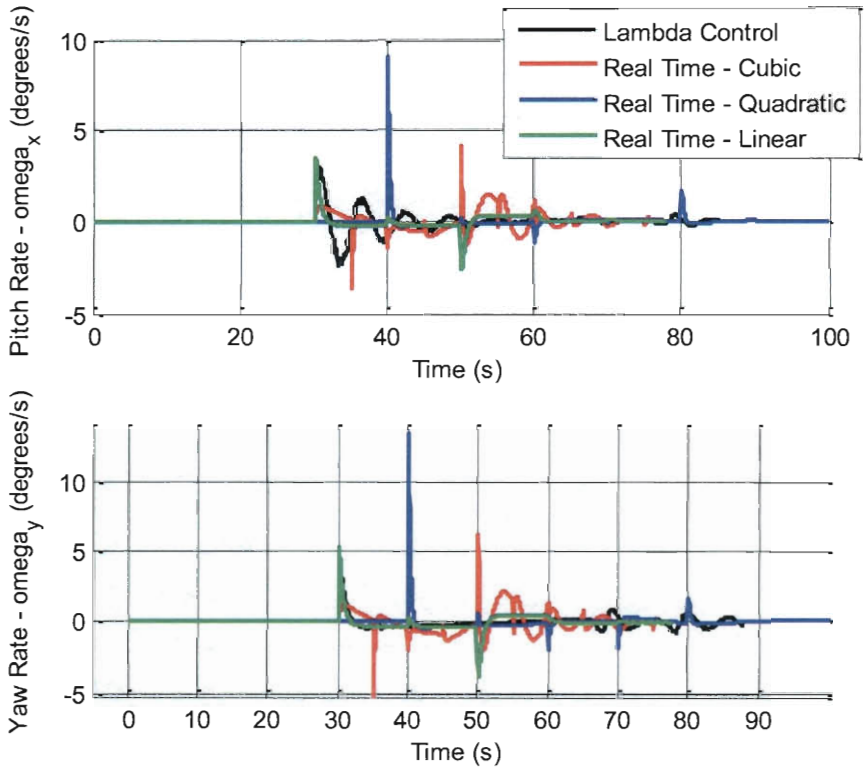

Figure 4-16: Angular Velocity of Vehicle Body vs. Time for jump Trajectory

4.2.2 Jump Trajectory with Wind

A vehicle operating on a planet with an atmosphere experiences aerodynamic disturbance forces, namely drag and wind. The model for how drag force is applied to the vehicle is discussed in Section 2.8.3, and is a function of the vehicle's velocity. Given this drag force function, wind is simply an additional velocity (with a magnitude and direction) that is added to the vehicle's velocity before drag force is calculated.

In reality, wind speed and direction is difficult to predict with certainty, and has a tendency to change quickly. As such, it is useful to examine the effects of wind using Monte Carlo analysis in order to develop a complete understanding of how wind force affects the vehicle.

A Monte Carlo analysis is run using 500 simulations, each time varying the wind speed using the random number functions of MATLAB. Wind speed is modeled using a normal (Gaussian) distribution, with a mean of 4.47 m/s (10 MPH) in both the $+i\hat{Y}$ and $+i\hat{Z}$ directions, and a standard deviation of 1.12 m/s (2.5 MPH), and is sampled every 0.5 seconds. A sample time history of wind speed is provided in

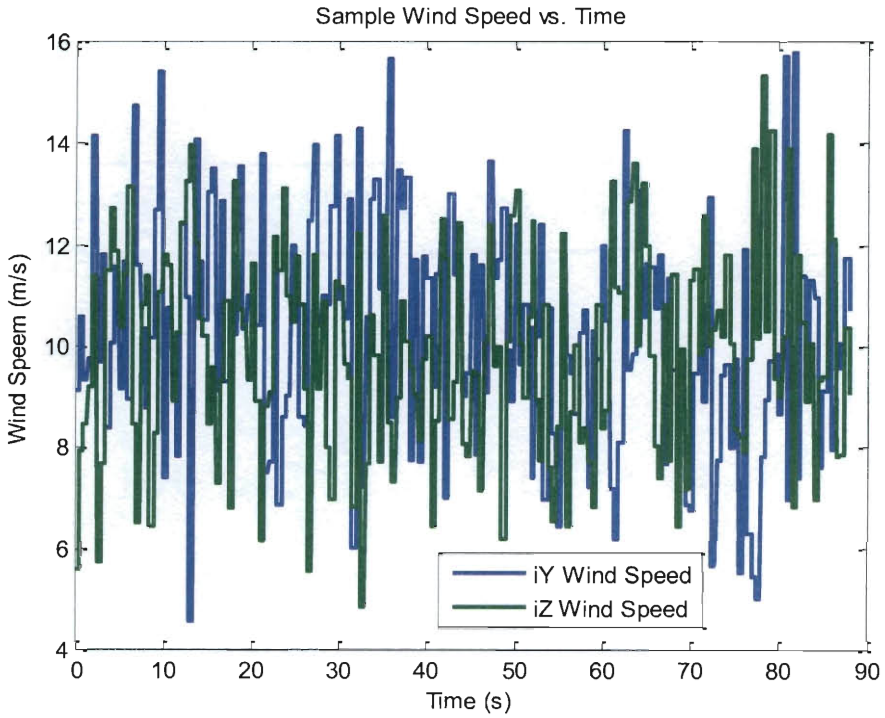

Figure 4-17: Sample Wind Speed Time History

The position vs. time plots for each method are shown in Figure 4-18. The real time methods exhibit more course deviation in general, due to the fact that they can only update for course deviations every five seconds (the rate at which guidance is set to recalculate a trajectory).

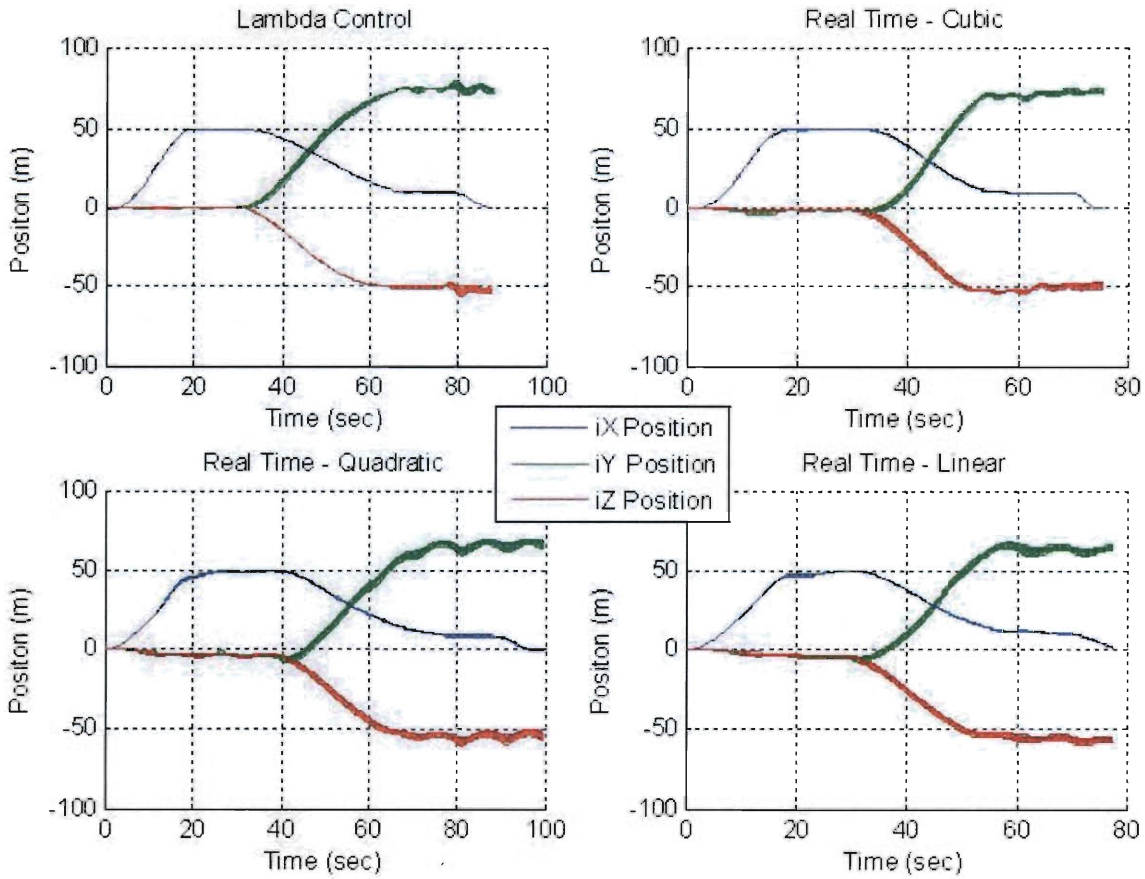

Figure 4-18: Position vs. Time Plots for Wind Monte Carlo Analysis

In addition to the position data, the maximum position error is recorded for each simulation, as well as the maximum vehicle attitude angles, maximum propellant slosh, and propellant used. Histograms showing the relative frequency of the absolute value of

maximum position error for each method are shown in Figure 4-19, Figure 4-20, Figure 4-21, and Figure 4-22.

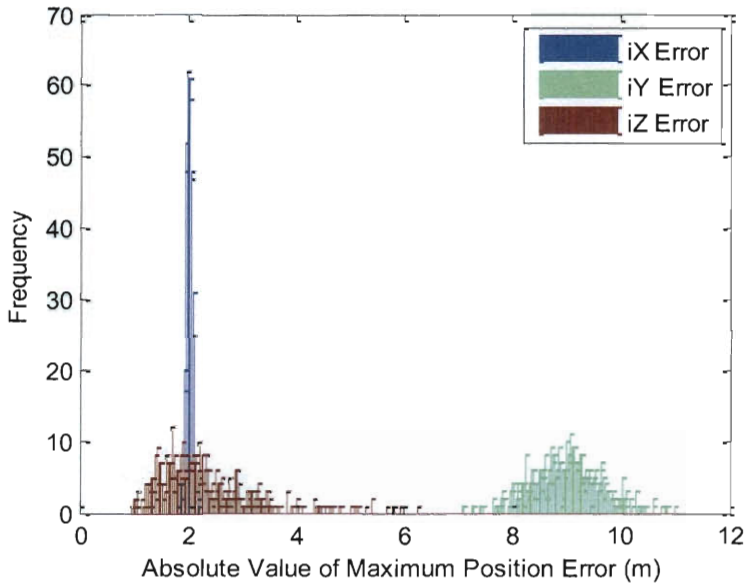

Figure 4-19: Histogram of Maximum Position Error Using Lambda Control Method

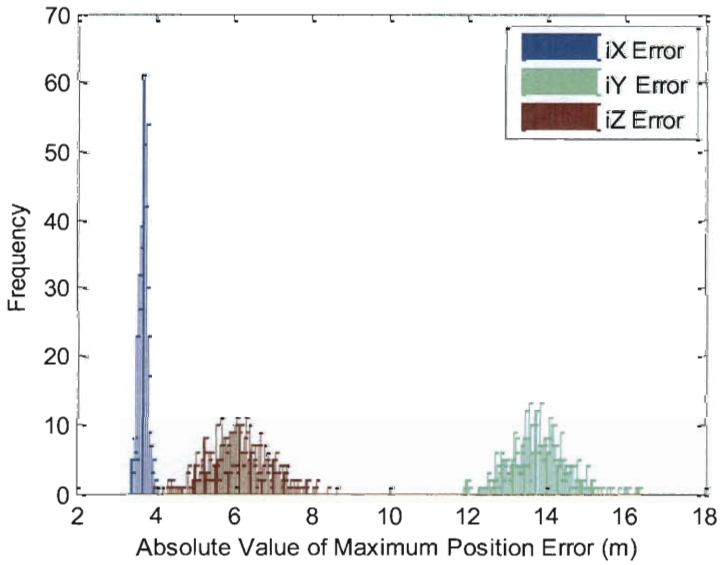

Figure 4-20: Histogram of Max Position Error Using Real Time Cubic Acceleration

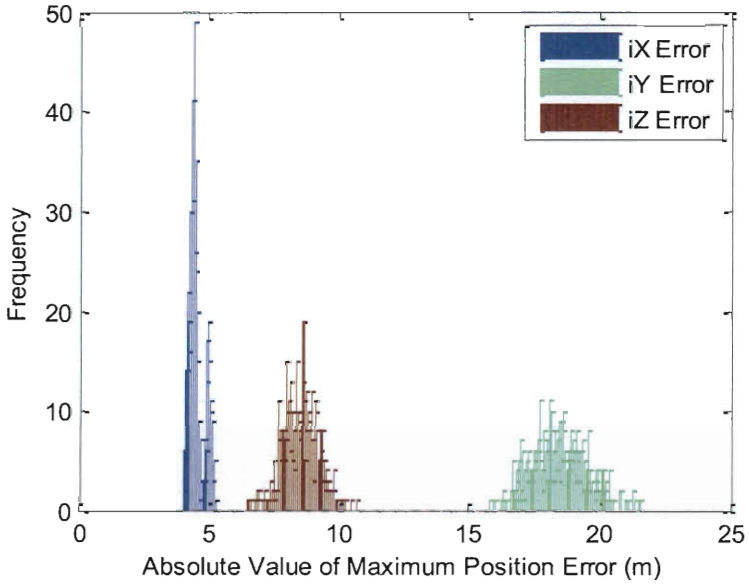

Figure 4-21: Histogram of Max Position Error Using Real Time Quad. Acceleration

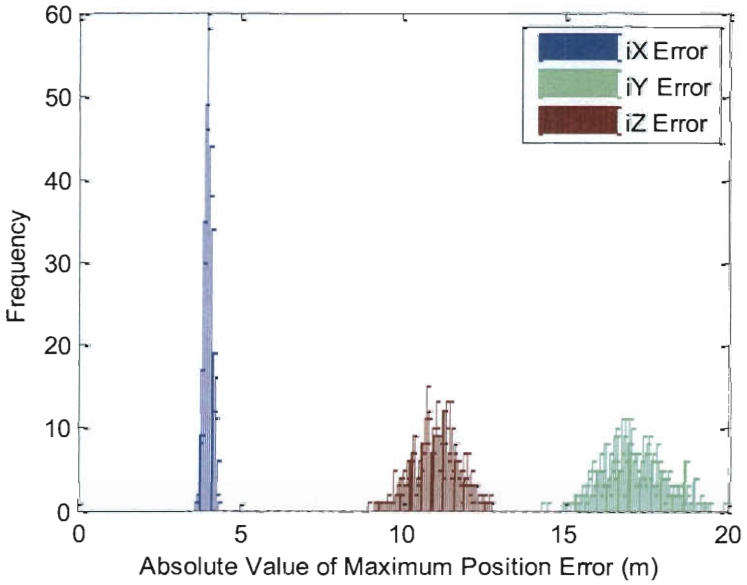

Figure 4-22: Histogram of Max Position Error Using Real Time Linear Acceleration

As expected, the error for each method in each direction is approximately normally distributed, as a result of the normally distributed wind force. The maximum position error statistics are summarized in Table 4-6.

Method	Mean of Max +/- Position Error (m)			Standard Deviation of Max +/- Position Error (m)		
	$i\hat{X}$	$i\hat{Y}$	$i\hat{Z}$	$i\hat{X}$	$i\hat{Y}$	$i\hat{Z}$
Lambda Control	2.00	8.98	2.25	0.06	0.63	0.87
Linear Profile	3.96	17.03	10.97	0.12	0.90	0.69
Quadratic Profile	4.53	18.42	8.49	0.30	1.03	0.66
Cubic Profile	3.66	13.75	6.09	0.11	0.70	0.76

Table 4-6: Position Error Statistics from Monte Carlo of Jump Trajectory w/ Wind

The lambda control method provides the lowest mean of maximum error. This is likely because lambda control makes corrections on the lambda command using feedback information (position and velocity) at a faster rate (every 0.5 seconds) than the real time methods. The real time methods calculate a new trajectory using feedback information every 5 seconds, leaving a larger gap of time where state disturbances such as those do to wind go uncorrected. Increasing the rate at which new trajectories are calculated for real time guidance reduces maximum position error, as disturbances are more frequently accounted for.

Table 4-7 shows the mean propellant use for each method for the 500 simulation Monte Carlo. The mean of relative propellant use for each method is exactly the same as the relative propellant use for the no wind case. Given the very small standard deviation, it is apparent that wind does not significantly affect propellant use.

Method	Mean of Relative Propellant Use	Standard Deviation of Relative Propellant Use
Lambda Control	1.00	10.00E-4
Linear Profile	0.90	5.21E-4
Quadratic Profile	1.11	1.36E-4
Cubic Profile	0.88	2.53E-4

Table 4-7: Propellant Use Statistics from Monte Carlo of Jump Trajectory w/ Wind

This analysis also shows how wind affects the attitude of the vehicle. The maximum rotation angle of the vehicle is recorded for each simulation run. Table 4-8 shows the mean and standard deviation of these attitude values for each method. As before, roll is not commanded or controlled. Compared to the no wind results shown in Table 4-5, the attitude values of the wind Monte Carlo are between about two and four times bigger than their no wind counterparts. The disturbance force introduced by adding wind likely requires the vehicle to rotate more to overcome course deviations.

Method	Mean of Max +/- Attitude (degrees)		Standard Deviation of Max +/- Phi (degrees)		
	Pitch	Yaw	Roll	Pitch	Yaw
Lambda Control	8.31	9.88	2.42	3.46	3.44
Linear Profile	3.55	5.40	0.77	0.27	0.31
Quadratic Profile	8.35	8.65	2.63	0.74	0.68
Cubic Profile	8.79	9.26	4.79	1.03	0.46

Table 4-8: Vehicle Attitude Statistics from Monte Carlo of Jump Trajectory w/ Wind

Table 4-9 shows the mean and standard deviation of the maximum slosh displacement for each method. As expected, wind causes higher slosh displacements for all methods, as the randomly changing wind force disturbs the spring mass damper modeled propellant slosh. Where as in the no wind case the cubic acceleration profile

provides the lowest maximum propellant slosh, now the lambda control method provides the least maximum slosh.

Method	Mean of Max Slosh - Any Direction (m)	Standard Deviation of Max Slosh - Any Direction (m)
Lambda Control	0.074	1.98E-2
Linear Profile	0.140	0.81E-2
Quadratic Profile	0.277	1.72E-2
Cubic Profile	0.105	1.10E-2

Table 4-9: Propellant Slosh Statistics from Monte Carlo of Jump Trajectory w/ Wind

4.2.3 High Fidelity Trajectory

The next trajectory examined a larger, more aggressive jump trajectory that demonstrates the capability of guidance to command the vehicle along a specific path with 'high fidelity.' A nominal trajectory is generated a priori, and the vehicle must follow the trajectory as closely as possible in the same amount of time.

As with the smaller jump trajectory, the Apollo approach lambda control guidance is used. In addition, real time guidance methods according to those discussed in Section 3.3.2.1 are used. For real time guidance, cubic, quadratic, and linear acceleration profiles are examined.

For the smaller jump trajectory (consisting of three discrete waypoints), guidance calculates a trajectory corresponding to the shortest time of flight to get from waypoint to waypoint. For high fidelity trajectory following, guidance must compute commands to follow a specific path in a specific amount of time. To do this, real time guidance does not use a search method to find time of flight t_{go}. Instead, t_{go} is set as the time until the next guidance calculation. Boundary conditions for guidance calculations are the current

state, and a point on the trajectory corresponding to the time of the next guidance calculation.

The nominal trajectory is shown in Figure 4-23. Compared to the smaller jump trajectory, the vehicle travels farther and faster, with faster change of direction. In this sense, the trajectory is more 'aggressive' as it requires higher body and gimbal rotations of the vehicle. The propellant tanks are about two-thirds full for this trajectory.

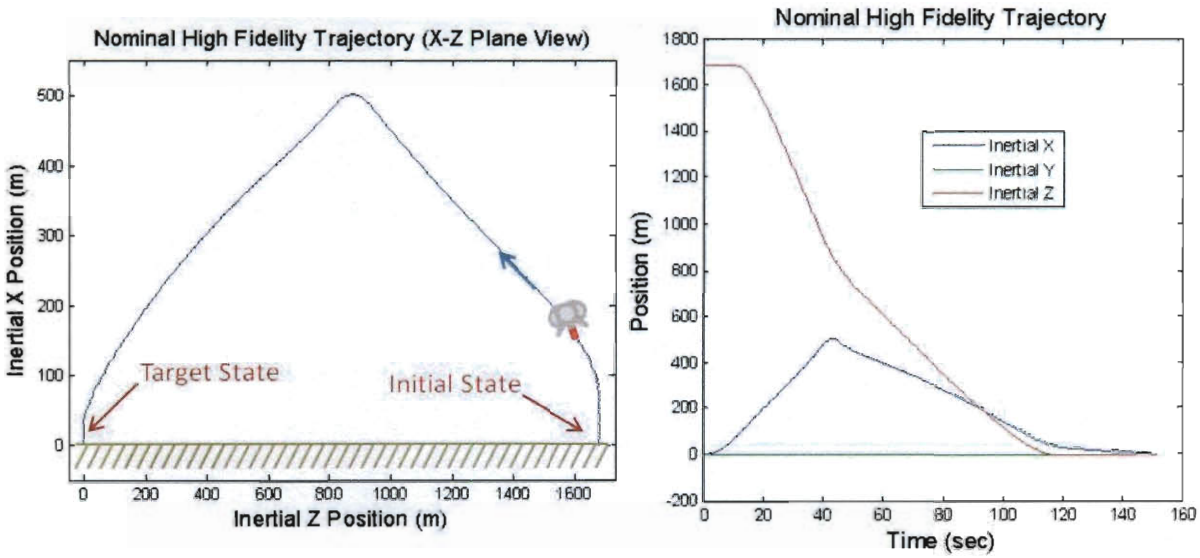

Figure 4-23: Nominal High Fidelity Trajectory

Figure 4-24 shows the trajectory error from nominal of the vehicle in the inertial frame. The real time methods exhibit similar behavior to each other, though some are slightly more accurate than others. The lambda control method exhibits smoother behavior, which is indicative of the fact that it is slower to respond to deviations from the nominal trajectory.

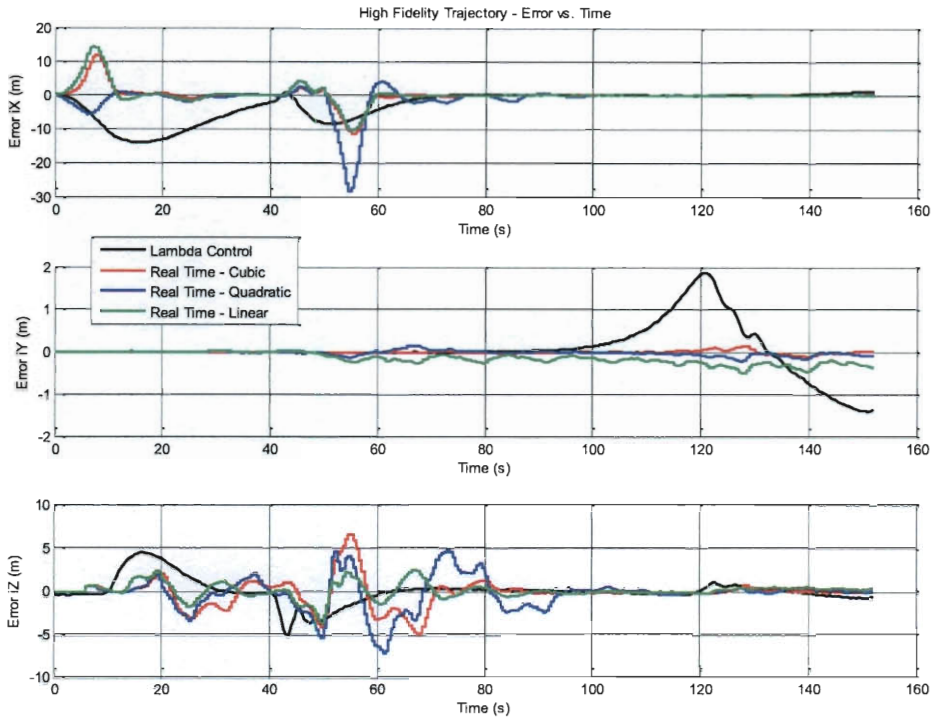

Figure 4-24: High Fidelity Trajectory Error vs. Time Plot

There are corresponding nominal velocity and acceleration plots for the nominal position trajectory. If the vehicle follows the nominal acceleration exactly, then it will follow the nominal velocity and position exactly as well. However, as position, velocity, and acceleration error accumulates, guidance must command a deviation from the nominal acceleration in order to get the vehicle back on trajectory. Figure 4-25 shows the acceleration deviation from nominal in each direction vs. time. The lambda control method and real time guidance cubic method follow the nominal acceleration more consistently that the real time quadratic and linear methods. Lambda control follows the nominal acceleration very closely because it is essentially using the nominal acceleration

as the lambda command, plus a small correction factor. The cubic acceleration profile is also be better than quadratic and linear profiles because it uses a target acceleration boundary condition, so it can aim for an acceleration target as well as velocity and position targets.

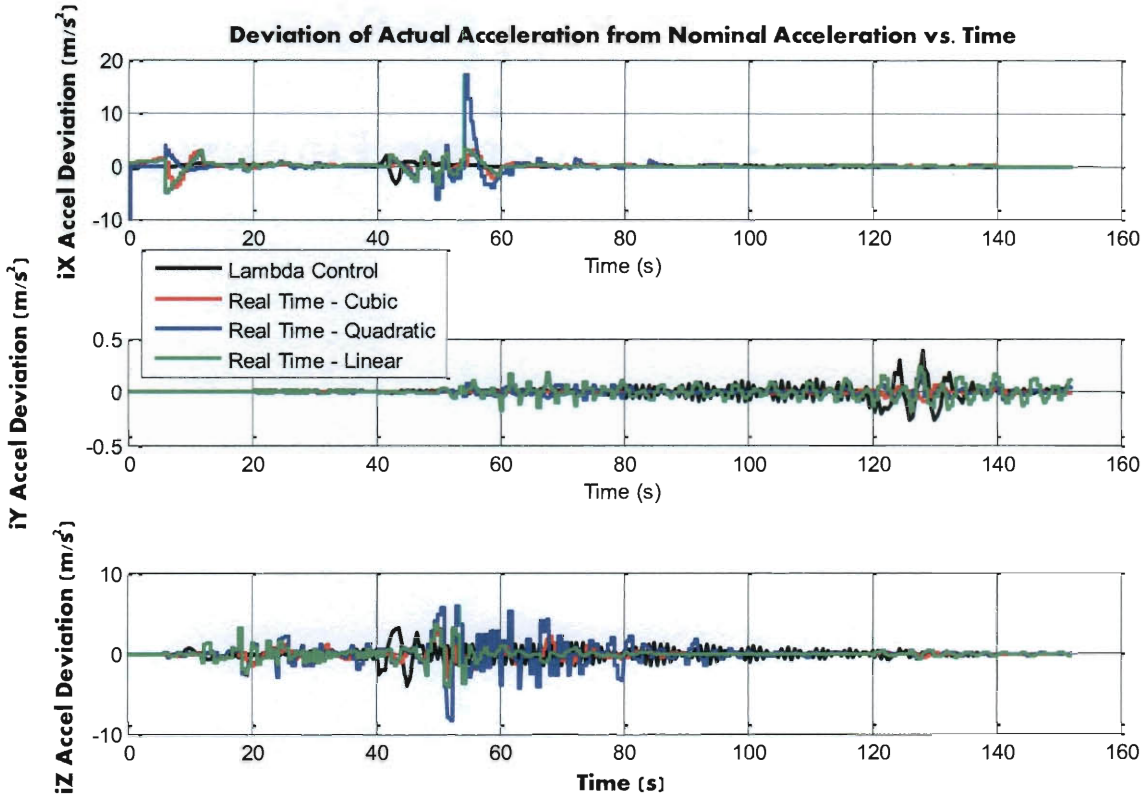

Figure 4-25: Deviation of Accel. from Nominal Accel. for High Fidelity Trajectory

The rotation (attitude) of the vehicle with respect to inertial provides a good indicator stable the vehicle is flying a trajectory. Ideally rotation from an upright orientation is minimal, in order to prevent propellant from sloshing or the vehicle from tipping over. The more the vehicle tips, the more the thrust is required or the more the

engine must gimbal in order to bring the vehicle to upright orientation. Figure 4-26 shows the pitch, roll, and yaw rotation of the vehicle vs. time. There is no command or control on roll, so it is free to deviate over time as aresult of the coupling movement of the engine gimbal in the $b\hat{y}$ and $b\hat{z}$ planes. The roll deviation is not significant, as it does not affect flight of the vehicle or safe landing.

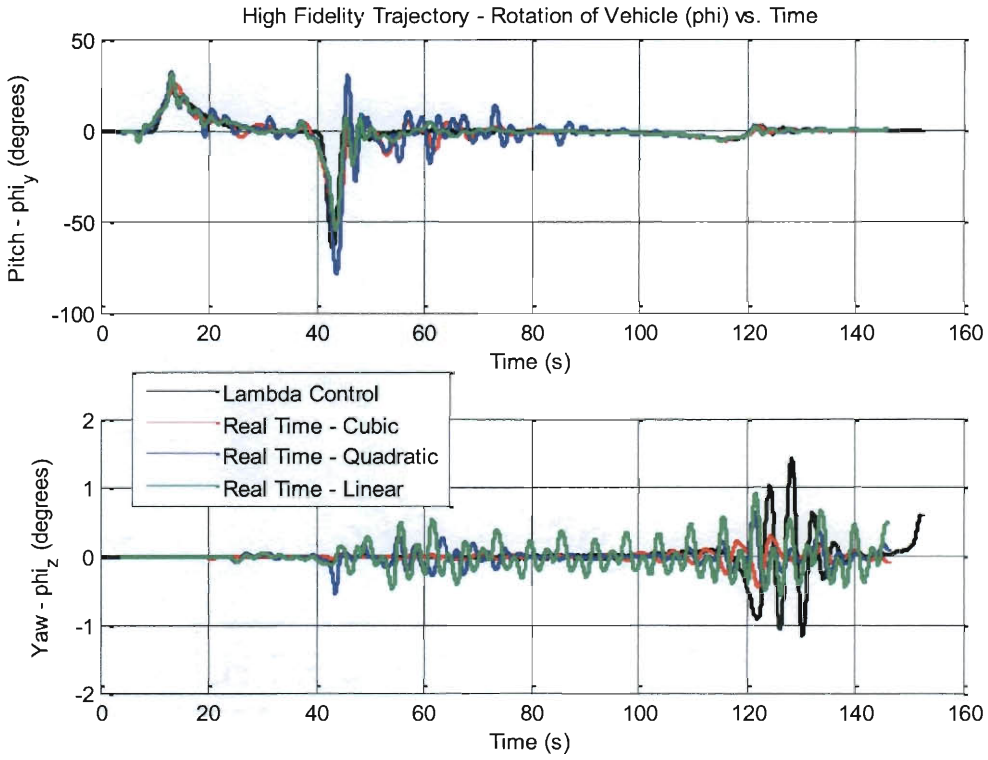

Figure 4-26: High Fidelity Trajectory Rotation of Vehicle vs. Time Plot

Yaw of the vehicle generated by the real time polynomial acceleration methods is oscillatory in nature, due to oscillations of acceleration in $i\hat{Y}$ shown in Figure 4-25. Though they are small for this case, such rotational oscillations are not ideal for the vehicle as they tend to increase propellant slosh (as a result of the associated angular

velocity and acceleration), which in turn exacerbates the problem. This can eventually lead to instability of the vehicle.

Table 4-10 shows relevant parameters from the trajectory simulation of each method. More than anything, it shows how similar the methods are to one another and that they all follow the nominal trajectory very closely.

Method	**Time of Flight (s)**	**Relative Propellant Use**	**Landing Error Magnitude (m)**
Lambda Control	152.0	1.00	1.935
Linear Accel. Profile	152.0	1.00	0.540
Quadratic Accel Profile	152.0	1.01	0.116
Cubic Accel. Profile	152.0	1.00	0.056

Table 4-10: Summary of Flight Characteristics for High Fidelity Trajectory

Table 4-11 shows the maximum vehicle rotation, trajectory deviation, and slosh for each method. The cubic acceleration profile deviates from nominal the least, while reaching the lowest rotation angles. Again, this is likely due to the fact that a cubic acceleration profile can match a target acceleration, velocity, and position, while quadratic and linear profiles only have velocity and position targets.

Method	Max +/- Attitude (degrees)		Max Trajectory Deviation (m)			Max Slosh - Any Direction (m)
	Pitch	Yaw	$i\hat{X}$	$i\hat{Y}$	$i\hat{Z}$	
Lambda Control	64.09	1.42	14.15	1.86	5.24	0.065
Linear Profile	54.40	0.90	14.36	0.49	3.67	0.071
Quadratic Profile	78.31	0.64	28.20	0.21	7.05	0.212
Cubic Profile	52.70	0.44	11.69	0.16	6.51	0.050

Table 4-11: Maximum Flight Characteristics for High Fidelity Trajectory

4.2.4 Lunar Descent Trajectory Using Planet Centered Gravity

The last trajectory examined is a lunar descent trajectory using planet centered gravity. For this, the origin of the inertial frame is at the center of the planet. The force of gravity vector points from the vehicle to the inertial frame, with a magnitude proportional to the distance of the vehicle to the inertial frame (see Section 2.8.2). For this trajectory, it is assumed that the vehicle has already de-orbited and begun powered descent. The propellant tanks are about two-thirds full for this trajectory. Figure 4-27 shows the phases of the lunar descent trajectory that is examined.

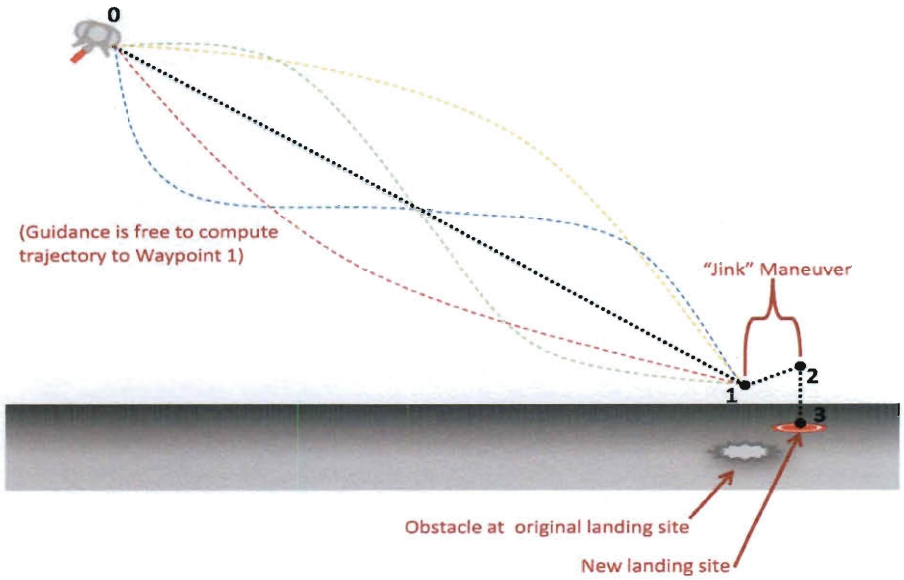

Figure 4-27: Lunar Descent Trajectory

There are three phases to the trajectory: an approach phase, a 'jink' maneuver, and a landing phase. The approach phase begins with the vehicle at an initial altitude of five kilometers, and requires it to travel two kilometers down range while simultaneously descending in altitude towards a point 50 meters directly over landing site. Unlike the high fidelity trajectory, guidance is free to compute a trajectory and time of flight to get from Point 0 to Point 1. The second phase is a simulated 'jink' maneuver, where the vehicle must maneuver from a point above the originally intending landing site to a point 200 meters cross range over to a new landing site. Once again, guidance is free to compute the trajectory and time of flight for this. Finally, the landing phase consists of a vertical descent to ground level.

This trajectory is useful to examine for several reasons. It has the largest time of flight of any trajectory examined, as well as the largest vertical and down range

translation. There is also an abrupt change in direction at the jink maneuver. These aspects stress the ability of each acceleration profile/implementation method to keep the vehicle on course and get the vehicle to the landing site as quickly as possible.

Once again the Apollo lambda control and real time guidance polynomial acceleration implementation methods are examined. The position plots for each method are shown below:

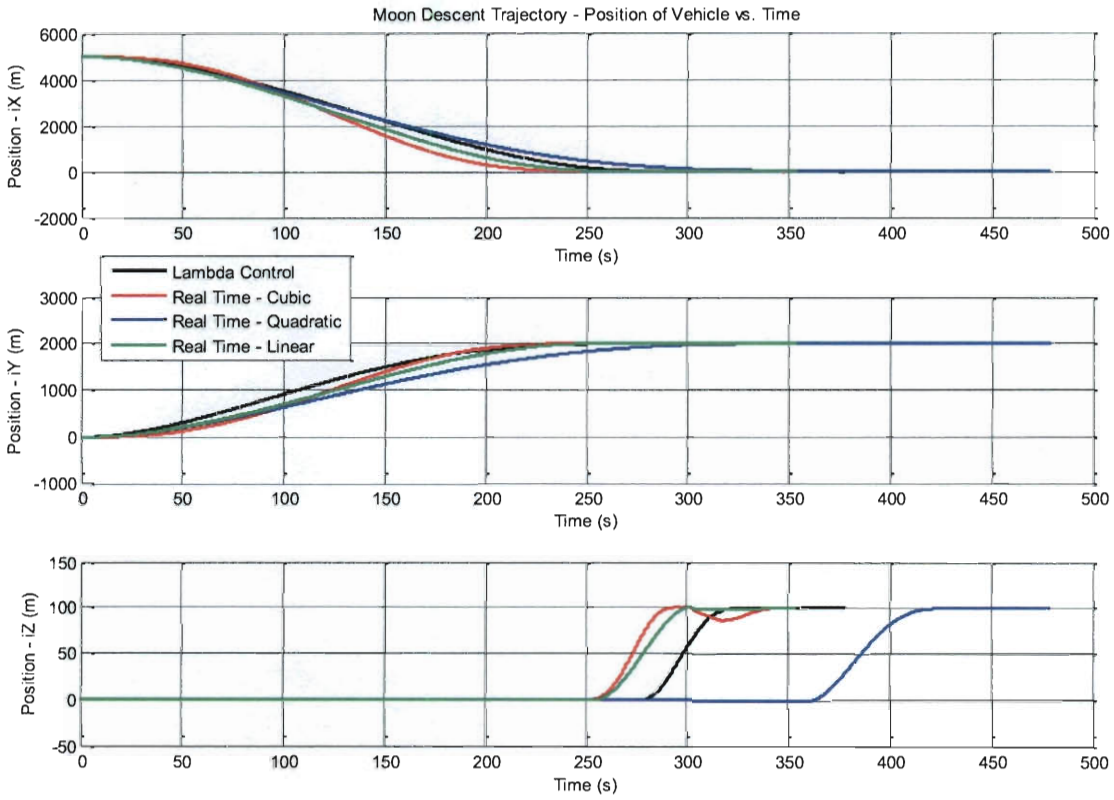

Figure 4-28: Moon Descent Trajectory Position of Vehicle vs. Time

A summary of the results for this trajectory are shown below:

Method	Time of Flight (s)	Relative Propellant Use	Landing Error Magnitude (m)
Lambda Control	397.5	1.00	0.23
Linear Accel. Profile	354	0.90	0.03
Quadratic Accel Profile	479	1.17	0.03
Cubic Accel. Profile	348	0.89	0.13

Table 4-12: Summary of Flight Characteristics for Moon Descent Trajectory

Like the other trajectories examined, real time guidance with a cubic acceleration profile has the best fuel use, though the linear acceleration profile yields very similar results. The quadratic acceleration profile once again has the slowest time of flight and worst fuel use.

Figure 4-29 shows a representative plot of the time of flight t_{go} until the vehicle gets to the next waypoint, as calculated by guidance. For this simulation, guidance recalculates t_{go} every five seconds. There is a lower limit for the calculated value of t_{go}, because as t_{go} gets smaller, the coefficients for the acceleration profile get larger to a point that the commanded acceleration trajectory cannot be followed. This fact is consistent with the findings of [29]. Figure 4-30 shows a position plot for the vehicle using real time guidance (cubic acceleration profile) with minimum t_{go} of two seconds. As can be seen, the vehicle does not fly as well as with a five second minimum t_{go} (Figure 4-28).

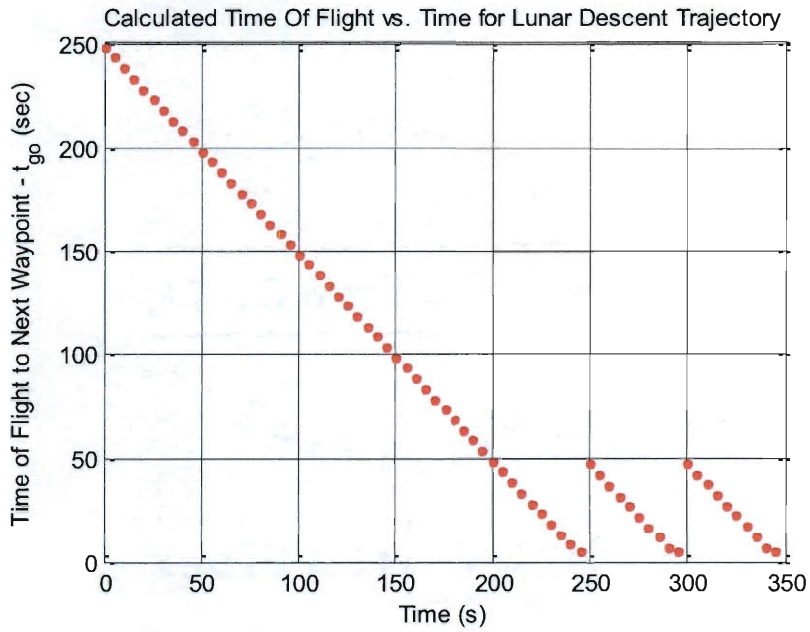

Figure 4-29: Example Calculated t_{go} to Next Waypoint vs. Time for Lunar Descent

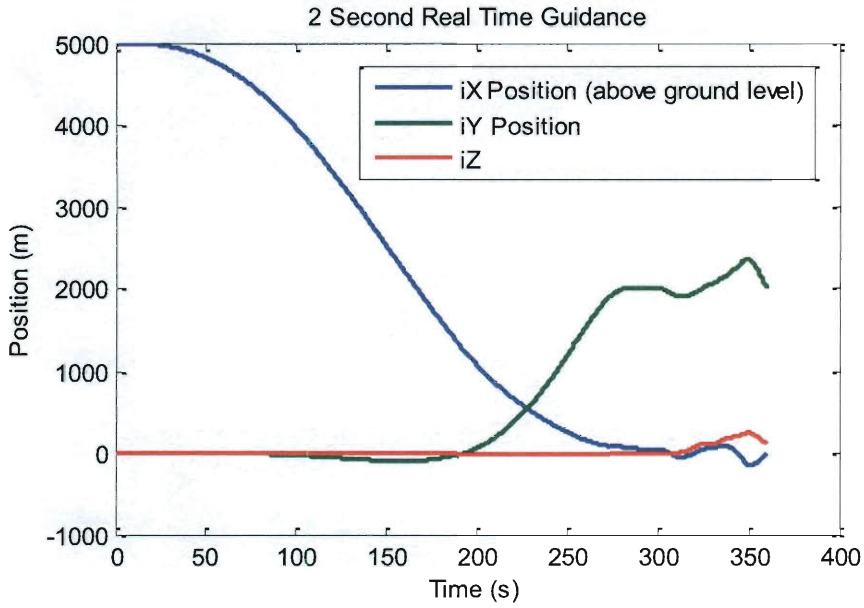

Figure 4-30: Position Plot for Two Second Real Time Guidance

As before, it is useful to examine the rotation of the vehicle with respect inertial in order to understand how aggressively the vehicle is turning in order to reach its destination. The pitch, roll, and yaw rotation of the vehicle is shown in Figure 4-31.

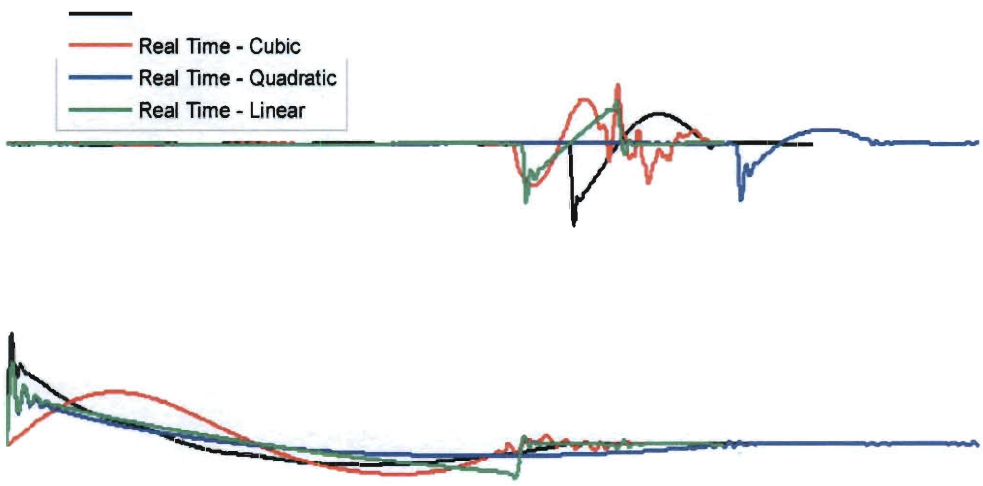

Figure 4-31: Lunar Descent Trajectory Rotation of Vehicle vs. Time Plot

For this analysis, the roll of the vehicle is irrelevant as there is no roll command or control on the vehicle, and the roll orientation does not matter with regards to landing safely. Pitch and yaw, however, should be zero at touchdown so the vehicle does not land tilted. This is the case for all methods tested.

Table 4-13 shows the max values of vehicle rotation for each method. The lambda control method tilts the vehicle the most and yields the greatest propellant sloshing. The real time polynomial acceleration methods yield similar rotations and propellant slosh, though cubic acceleration has a small edge over the other two in yaw and propellant slosh.

Method	Max +/- Attitude (degrees)		Max Slosh - Any Direction (m)
	Pitch	Yaw	
Lambda Control	19.14	19.50	3.3E-2
Linear Profile	13.45	14.67	2.4E-2
Quadratic Profile	13.19	14.59	2.5E-2
Cubic Profile	13.48	9.23	2.0E-2

Table 4-13: Maximum Flight Characteristics for Lunar Descent Trajectory

4.3 Footprint Analysis

The method for estimating the footprint of a vehicle developed in Section 3.5 is implemented and used to examine how changing initial velocity and propellant mass affects the landing footprint of a vehicle.

The method implemented is guidance based: a kinematic guidance algorithm using a cubic acceleration profile is iteratively run to compute trajectories with increasing down range and/or cross range targets. For each iteration, propellant use for the calculated trajectory is estimated and compared to the propellant available. The trajectory for which the estimated propellant equals the propellant available becomes the boundary of the vehicle's footprint. This process is repeated for various directions down range and/or cross range to develop a complete footprint.

The first test case is for zero initial velocity. The footprints for three different initial propellant masses are shown in Figure 4-32. As expected, the shape of each footprint is circular, meaning the vehicle can travel equally far in all directions.

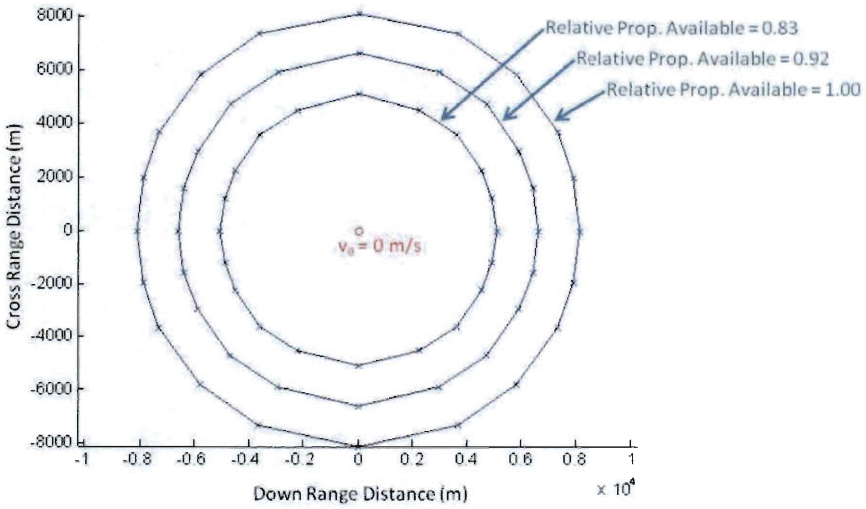

Figure 4-32: Footprints for Vehicle with Zero Initial Velocity

Figure 4-33 shows the footprints for a vehicle with an initial velocity of 25 m/s down range, and Figure 4-34 shows the footprints for the same vehicle with an initial velocity of 50 m/s down range. These results show that the footprint of a vehicle becomes more elliptical as initial velocity increases, as expected. In addition, the footprint ellipse is shifted so that the initial position of the vehicle is located increasingly closer to the periapsis of the ellipse. This is representative of the fact that the vehicle becomes less and less able to double back on its path of flight as initial velocity increases.

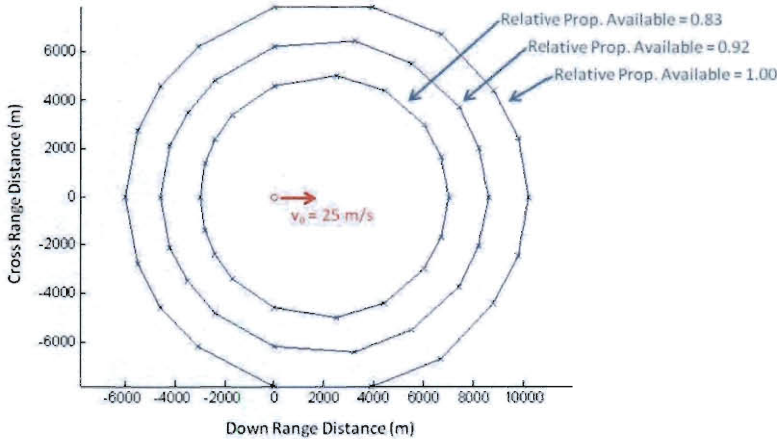

Figure 4-33: Footprints for Vehicle with Initial Velocity 25 m/s Down Range

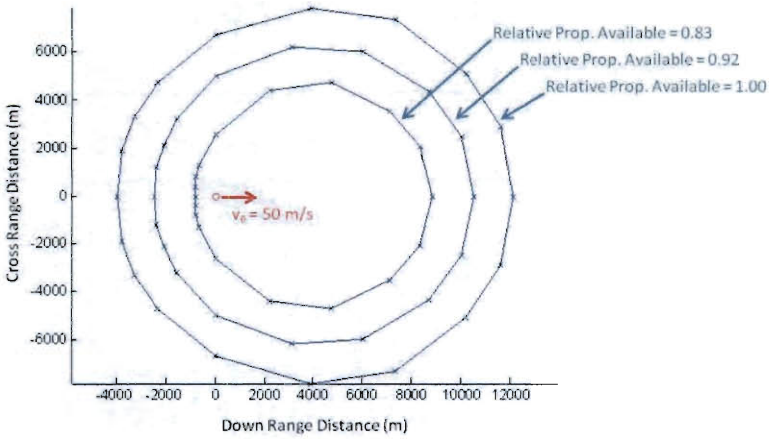

Figure 4-34: Footprints for Vehicle with Initial Velocity 50 m/s Down Range

It has been observed that other vehicle and simulation parameters affect the shape and size of landing footprints. The time of flight bounds for which potential trajectories can be calculated (see Step 3 in Figure 3-13) greatly affect how far the vehicle can go, as well as engine parameters such as max thrust or specific impulse.

For this analysis, it has been observed that the initial altitude of the vehicle does not affect the size of the footprint. This is because the same amount of propellant is expended keeping the vehicle in hover as is slowing its descent, given the same time of flight.

5 Closure

The goal of guidance is to develop a trajectory that takes a vehicle from its current (initial) state to a desired final (target) state in a certain time of flight. This thesis examines several methods for implementing guidance for an autonomous vehicle. The methods examined include: Apollo lunar descent guidance, modified Apollo guidance, as well as time and fuel efficient methods.

To facilitate the assessment of each guidance method, the dynamics for an arbitrary but representative planetary lander vehicle is developed. The significant characteristics of this lander are propellant tanks with propellant capable of sloshing, and a gimbaled engine used for thrust vector control (TVC). The dynamics is examined with respect to six degrees of freedom (6DOF): three each for translational and rotational degrees of freedom. Equations of motion for translational acceleration of the vehicle, angular acceleration of the vehicle, angular acceleration of the engine, and propellant slosh displacement acceleration are derived. Methods for modeling changing mass parameters, planet centered gravity, and atmospheric drag and wind are also developed.

Guidance is first examined as a two point boundary value problem (BVP), with the boundaries being the initial and target state of the vehicle. The solution to the BVP is an acceleration profile that if followed takes the vehicle to the target state in a certain time of flight. Linear, quadrtatic, and cubic polynomial acceleration profiles are developed. Furthermore, a fuel efficient method for commanding constant acceleration is developed, using kinematics to determine the time of flight for each acceleration level. A time efficient method for applying physical constraints on the vehicle (such as a

maximum thrust constraint) and determining the time of flight for a polynomial acceleration profile is discussed.

Two methods for implementing guidance in a legacy GNC architecture are developed. The first is an Apollo approach method, where guidance acceleration commands are calculated a priori and are referenced open loop during flight. A closed loop controller is used to correct the open loop acceleration command so that the vehicle better follows the desired trajectory. The second method is real time guidance, in which guidance calculates a new acceleration profile for the vehicle in real time. For this method, no controller is necessary on the acceleration command, as the recomputed acceleration profile takes into account trajectory deviations. A major advantage of real time guidance is that new waypoints (new target states for the BVP) can be uploaded to the vehicle during its flight, and guidance will automatically re-compute trajectories to get to the desired state.

The polynomial acceleration profiles are first compared with the constant thrust acceleration profile using two simple trajectories. The first is a vertical descent. Each acceleration profile is used to determine a trajectory kinematically (without vehicle dynamics), and in simulation with 6DOF vehicle dynamics. Time of flight and fuel use is observed: the constant acceleration profile has the lowest fuel use and time of flight, followed by cubic, linear, then quadratic. The next trajectory is a simple horizontal translation (altitude is maintained). Once again, constant acceleration has the lowest fuel use and time of flight, followed by the polynomial acceleration profiles in the same order. However, the constant acceleration profile has the highest peak velocity, pitch angle, and

slosh displacement, making it less desirable as these can all be contributing factors to vehicle instability.

The two implementation methods, Apollo approach (lambda control) and real time are compared for three different trajectories. For the real time implementation, cubic, quadratic, and linear acceleration profiles are examined. The first trajectory is a jump trajectory. The real time implementation with cubic acceleration presents the best results: lowest time of flight and fuel use, and least propellant slosh.

The jump trajectory is also examined in the presence of wind. A 500 sample Monte Carlo shows that the real time methods yield higher trajectory deviations and slosh than the lambda control method in the presence of wind. This is due to the fact that real time guidance can only correct for course deviations every five seconds as that is how often it re-computes a trajectory. Raising the rate for real time guidance yields vehicle instability, but may be possible with a different control scheme.

The second trajectory is a high fidelity trajectory, which demonstrates the vehicles ability to follow an aggressive trajectory as closely as possible. Fuel use and time of flight for all methods is essentially identical, as the vehicle is required to follow not only a specific path, but do it in a specific amount of time. For this case the lambda control method is the slowest to respond to position error, as the time of flight parameter is used to calculate lambda control gains, which are small at some times and can cannot correct immediately for large errors. The real time implementation with cubic acceleration has the lowest maximum attitude, position deviation, and slosh.

The third trajectory is a simulated moon descent. This trajectory covers the longest range, allowing each guidance method to operate over a longer period of time.

Once again the real time guidance with cubic acceleration has the best time of flight, fuel use, lowest body rotation, and lowest propellant slosh. Real time guidance is a close second for these parameters, and has slightly better landing error. For this trajectory, the lambda control method has the highest landing error, highest body rotation, and highest slosh.

In general, the real time methods (except for the quadratic acceleration profile) perform better than the Apollo approach lambda control method, except in the presence of wind. The real time methods, at the guidance rate used, do not respond as quickly to the random disturbance force of wind. Developing a way to run real time guidance faster is one potential solution to this problem. Another option is to combine real time guidance with a lambda controller. A potential issue with this, though, is that there are too many correction factors, which cause the vehicle to consistently overshoot its desired course.

Of the real time methods, the cubic acceleration profile yields the most desirable flight characteristics. One reason for this may be that it requires initial and final boundary conditions on acceleration, which means that there are never instantaneous changes in acceleration. However, the linear acceleration profile, which has no boundary conditions for acceleration, performs almost as well as the cubic profile. The quadratic profile consistently yields the least desirable flight characteristics, and should be avoided for use.

Finally, the guidance based approach for estimating a vehicle's footprint has been validated. The method yields rough estimates of landing ellipses based on initial vehicle conditions. As expected, an initial vehicle velocity elongates the vehicle ellipse. Propellant available is the biggest contributing factor to the size of the ellipse. It is

observed that for the scope of this analysis, initial altitude does not matter as the vehicle expends the same amount of propellant given the same time of flight, regardless of whether it is descending from a large altitude or small altitude. The guidance based approach is computationally intensive, so a faster estimation method may be necessary to be able to implement footprint analysis in real time.

Bibliography

[1] T. Brady, E. Robertson, C. Epp, S. Paschall, and D. Zimpfer, "Hazard Detection Methods for Lunar Landing," IEEE Aerospace conference, 2009.

[2] Allan Y. Lee, Todd Ely and Ronald Sostaric and Alan Strahan and Joseph E. Riedel and Mitch Ingham and James Wincentsen, , and , Siamak Sarani, "Preliminary Design of the Guidance, Navigation, and Control System of The Altair Lunar Lander," AIAA Guidance, Navigation, and Control Conference Toronto, Ontario Canada, 2010.

[3] J. Sellers, W. Astore, R. Giffen, and W. Larson, *Understanding Space: An Introduction to Astronautics + Website*, 3rd ed. Learning Solutions, 2007.

[4] A.R. Klumpp, "Apollo lunar-descent guidance," *MIT Charles Stark Draper Laboratory R-695*, 1971.

[5] A.R. Klumpp, "Apollo Lunar Descent Guidance," *Automatica*, vol. 10, pp. 133-146, 1974.

[6] N. S. Nise, *Control Systems Engineering*, 4th ed. Wiley, 2004.

[7] Timothy Phillips, "Control Design and Stabilty Analysis of a Gimbaled Engine Vehicle," Rice University, 2011.

[8] Scott R. Ploen, A. Behcet Acikmese, , and , Aron Wolf, "A Comparison of Powered Descent Guidance Laws forMars Pinpoint Landing," presented at the AIAA/AAS Astrodynamics Specialist Conference and Exhibit, AIAA/AAS Astrodynamics Specialist Conference and Exhibit Keystone, Colorado, 2006.

[9] D. Dooling, "L+25: a quarter century after the Apollo landing," *IEEE Spectrum*, vol. 31, no. 7, pp. 16-29, 1994.

[10] J. Meditch, "On the problem of optimal thrust programming for a lunar soft landing," *Automatic Control, IEEE Transactions on*, vol. 9, no. 4, pp. 477-484, 1964.

[11] A. Miele,, "The Calculus of Variations in Applied Aerodynamics and Flight Mechanics," in *Optimization Techniques*, G. Leitmann, Eds. New York: Academic Press, 1962.

[12] Federico Najson and Kenneth D. Mease, "A Computationally Non-Expensive GuidanceAlgorithm for Fuel Efficient Soft Landing," AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco, California, 2005.

[13] Jeremy R. Rea and Robert H. Bishop, "Analytical Dimensional Reduction of a Fuel Optimal Powered Descent Subproblem," AIAA Guidance, Navigation, and Control Conference, Toronto, Ontario Canada, 2010.

[14] U. Topcu, J. Casoliva, and K. D. Mease, "Minimum-Fuel Powered Descent for Mars Pinpoint Landing," *Journal of Spacecraft and Rockets*, vol. 44, no. 2, pp. 324-331, 2007.

[15] G. Gelly and P. Vernis, "Neural Networks as a Guidance Solution for Soft-Landing and Aerocapture," AIAA Guidance, Navigation, and Control Conference Chicago, Illinois, 2009.

[16] Chang-Kyung Ryoo, Hangju Cho, , and , Min-Jea Tahk, "Optimal Guidance Laws with Terminal Impact Angle Constraint," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 4, Aug. 2005.

[17] M. J. Tahk, C. K. Ryoo, and H. Cho, "Recursive time-to-go estimation for homing guidance missiles," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 38, no. 1, p. 13–24, 2002.

[18] U. Topcu, J. Casolivay, , and , K.D. Measey, "Fuel Effcient Powered Descent Guidance for MarsLanding," AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco, California, Aug. 15-18, 2005.

[19] Paul Mitiguy, *Advanced Dynamics and Motion Simulation*. Stanford University, 2009.

[20] Paul A. C. Mason and Scott R. Starin, *Propellant Slosh Analysis for the Solar Dynamics Observatory*. Goddard Space Flight Center.

[21] K.W. London, *AFully Coupled Multi-Rigid-Body Fuel Slosh Dynamics Model Applied to the Triana Stack*. Swales Aerospace.

[22] D. T. Greenwood, *Principles of Dynamics*, 1st ed. Prentice Hall, 1987.

[23] G. P. Sutton and O. Biblarz, *Rocket Propulsion Elements, 7th Edition*, 7th ed. Wiley-Interscience, 2000.

[24] R. A. Ibrahim, *Liquid sloshing dynamics: theory and applications*. Cambridge University Press, 2005.

[25] D. A. Vallado, *Fundamentals of Astrodynamics and Applications*, 3rd ed. Microcosm Press/Springer, 2007.

[26] J. H. Blakelock, *Automatic Control of Aircraft and Missiles*, 2nd ed. Wiley-Interscience, 1991.

[27] Nobuhiro Yamanish, Toshiya Kimura and Masahiro Takahashi and Koichi Okita and Hideyo Negishi, , and , Masahiro Atsumi, "Transient Analysis of the LE-7A

Rocket Engine Using the Rocket Engine Dynamic Simulator (REDS)," 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit Fort Lauderdale, FL, 2004.

[28] E. Wong, James Masciarelli and and and Gurkirpal Singh"Autonomous Guidance and Control Design for Hazard Avoidance and Safe Landing on Mars," AIAA Atmospheric Flight Mechanics Conference and Exhibit, Monterey, California, 2002,

[29] Ronald Sostaric and Jeremy Rea, "Powered Descent Guidance Methods For The Moon and Mars," AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco, California, 2005.

[30] L. Blackmore, B. Acikmese, and D. P. Scharf, "Minimum-Landing-Error Powered-Descent Guidance for Mars Landing Using Convex Optimization," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 4, pp. 1161-1171, 2010.

[31] A. Saraf, J. A. Leavitt, K. D. Mease, and M. Ferch, "Landing Footprint Computation for Entry Vehicles," presented at the AIAA Guidance, Navigation, and Control Conference and Exhibit, Providence, Rhode Island, 2004.

[32] P. Lu and S. Xue, "Rapid Generation of Accurate Entry Landing Footprints," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 3, pp. 756-767, 2010.

[33] S. R. Ploen, H. Seraji, and C. E. Kinney, "Determination of spacecraft landing footprint for safe planetary landing," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 1, p. 3–16, 2009.

[34] R. Humble, G. Henry, and W. Larson, *Space Propulsion Analysis and Design with Website*, 1st ed. Learning Solutions, 2007.