Spring 5-2015

Euler-Lagrange Optimal Control of Indirect Fire Symmetric Projectiles

Austin L. Nash
Rose-Hulman Institute of Technology

Euler-Lagrange Optimal Control of Indirect Fire Symmetric Projectiles

A Thesis

Submitted to the Faculty

of

Rose-Hulman Institute of Technology

by

Austin L. Nash

In Partial Fulfillment of the Requirements of the Degree

of

Master of Science in Mechanical Engineering

May 2015

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Final Examination Report

Austin Nash	Mechanical Engineering
Name	Graduate Major

Thesis Title Euler-Lagrange Optimal Control of Indirect Fire Symmetric Projectiles

DATE OF EXAM: April 23, 2015

EXAMINATION COMMITTEE:

	Thesis Advisory Committee	Department
Thesis Advisor:	Bradley Burchett	ME
	Daniel Kawano	ME
	Ronald Artigue	CHE

PASSED X **FAILED** ____

Abstract

Nash, Austin L.

M.S.M.E.

Rose-Hulman Institute of Technology

May 2015

Euler-Lagrange Optimal Control of Indirect Fire Symmetric Projectiles

Thesis Advisor: Dr. Bradley T. Burchett

An important aspect of controls engineering is the dynamic modeling and flight control of smart weapons. One division of this area involves the guidance, navigation, and control of smart projectiles. In recent decades, methods for controlling projectiles have become much more sophisticated.

In this thesis, principles of optimal control are used to develop a controller for indirect fire symmetric projectiles, or high-launch projectiles. A plant model is created to simulate the flight of a 2.75-inch Hydra-70 rocket. Two pairs of forward-mounted controllable canards are used as actuators to modify the flight toward a downrange target. A linear optimal regulator is used to compute control inputs which minimize a cost function.

Results are demonstrated through impact point dispersion plots which show both the effectiveness and robustness of the controller. Additionally, defining characteristics of the control method are explored and optimized.

TO

To my parents, who have been outstanding role models and have provided unwavering support throughout my life. To my sister, who has always been a best friend to me.

To the entire faculty and staff of the Rose-Hulman Mechanical Engineering department. You have created an environment which enables students to maximize their potential.

And to Caitlin, who has handled the last six years with grace and understanding. Your support means everything to me.

Thank you.

Acknowledgments

I would like to acknowledge the following people for their help in the creation, editing, and formatting of this thesis and all of its contents.

Dr. Bradley Burchett

Dr. Daniel Kawano

Dr. Ronald Artigue

Dr. Caroline Carvill

Darryl Mouck

Table of Contents

List of Figures

List of Tables

Nomenclature

$\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{R}$	Optimal control weighting matrices
$\boldsymbol{A}, \boldsymbol{B}$	Linear control state-space matrices
C_{NA}	Normal force aerodynamic coefficient
C_{MA}	Pitch moment due to angle of attack aerodynamic coefficient
C_{X0}, C_{X2}	Axial force aerodynamic coefficients
C_{LP}	Spin damping moment aerodynamic coefficient
C_{DD}	Fin cant rolling moment aerodynamic coefficient
C_{MQ}	Pitch damping aerodynamic coefficient
D	Projectile reference diameter (ft)
g	Gravitational constant, 32.2 (ft/s^2)
m	Projectile mass (slug)
$\boldsymbol{I}$	Identity matrix
I_{xx}, I_{yy}, I_{zz}	Inertia components in projectile reference frame (slug-ft^2)
m	Projectile mass (slug)
p, q, r	Angular velocity vector components expressed in fixed-plane reference frame (rad/s)
SL_{cg}	Station line of projectile c.g. location (ft)
SL_{cp}	Station line of projectile c.p. location (ft)
s	Downrange distance (calibers)
u, v, w	Translational velocity components of projectile center of mass in fixed-plane referene frame (ft/s)
V	Magnitude of mass center velocity (ft/s)
x, y, z	Position vector components of projectile mass center in inertial reference frame (ft)
ψ, θ, ϕ	Euler yaw, pitch, and roll angles (rad)

$\boldsymbol{u}$	Optimal control input (rad)
$\boldsymbol{\eta}, \xi, \dot{w}$	State vector terms for linear controller which make up state vector $\boldsymbol{x}$
$\boldsymbol{\Lambda}, \boldsymbol{\Sigma}, \boldsymbol{\Phi}, \boldsymbol{\Xi}, \boldsymbol{\Gamma}$	Terms in control state dynamics matrix
X, Y, Z	Total force components on projectile (lbf)
L, M, N	Total moment components on projectile (lbf-ft)
ρ	Air density (slug/ft^3)
Ma	Mach number
c	Speed of sound (ft/s)
α_C	Canard angle of attack (rad)
δ_C	Canard pitch angle (rad)
C_{Y0}, C_{Z0}	Aerodynamic trim force coefficients
M_1, M_2	Vacuum trajectory coefficients
h	Arc length (calibers)
$\bar{\theta}$	Pitch angle(s) predicted by vacuum model for respective segment
δ_θ	Theta perturbation state for altitude control
$C_{L\alpha}$	Lift coefficient to convert controller output to dimensional form (rad^{-1})
$\boldsymbol{N}$	Solution to matrix Riccati differential equation
R_{CAX}, R_{MCP}	Distance from stationline c.g. to stationline c.p. (ft)
$\boldsymbol{F}, \boldsymbol{Z}$	Matrix Hamiltonian and back-propagation algorithm terms for controller
S	Projectile reference area (ft^2)
S_C	Canard reference area (ft^2)

subscript

c	Denotes term is representing canards, rather than projectile itself

1. Introduction

In recent decades, the design and control of autonomous weapons has become an important aspect of controls engineering. One division of this area involves the guidance, navigation, and control of missiles and projectiles. The term guidance refers to the determination of an optimal trajectory for a system, while navigation concerns tracking the current state of the system, and control refers to the use of some mechanism to alter the trajectory such that it reaches a desired location [1].

In the pre-World War era, missiles were commonly used as area weapons. Launches were made with the intent of targeting a general area. Since the Second World War, guided missiles have become prevalent weapons due to advancements in science and technology. Autonomous control is impactful in many regions of missile use such as underwater homing torpedoes, intercontinental ballistic missiles, and guided projectiles [2].

This work will focus on the control of guided projectiles. Feedback controllers can be implemented such that the trajectory of a projectile is altered throughout its flight and its impact is guided to a very precise location. Methods of controlling missiles and projectiles range from the use of pulse jets to fin control. In the case of pulse jet control, sets of pulse jets are often installed near the nose of the rocket. The pulses of the jets are then controlled in order to cause the rocket to exhibit a desired behavior, such as minimizing yaw rate or pitch rate [3]. With fin control, pairs of controllable fins, or canards, are located near the nose of the rocket and controlled such that their orientation causes certain aerodynamic reactions to be imparted onto the rocket, which then alters its flight.

In this thesis, principles of optimal control theory will be used in order to manipulate two pairs of forward-mounted canards. This work seeks to expand on previous efforts in controlling direct fire projectiles in part by incorporating a series of implicit pitch angle predictions to better allow for altitude control of indirect fire launches, or launches at high initial pitch angles. The goal is to minimize impact point dispersion for a wide range of initial pitch and yaw launch angles. In this chapter, a history of aerodynamic modeling of ballistic missiles will be presented. Additionally, recent work on projectile control will be discussed.

1.1 History of Ballistic Missile Dynamic Modeling

The following section is adapted from information via the work of Hainz and Costello [4] and the work of McCoy [5]. Prior to the 16th century, it was widely accepted that projectiles flew in a straight line. Galileo Galilei recognized that the flight was parabolic in nature. Galilei modeled a projectile as a point mass with gravity being the only acting external force. In the late 1600s and early 1700s, Isaac Newton established the framework for modern classical mechanics. Newton devoted time to the idea of a projectile in a resisting medium, effectively advancing the idea of aerodynamic drag which had been established by Johann Bernoulli. As centuries have passed, aerodynamic models have become much more sophisticated.

In the 19th century, scientists throughout the western world undertook experimental firings in order to improve the accuracy of drag measurements. Ballisticians from England, France, Germany, Russia, and the United States each conducted tests for measuring drag coefficients as a function of projectile velocity. Additionally, advancements were made in optimizing the shapes of projectiles with respect to drag, with the discovery that the use of long, slender projectile shapes could cut drag significantly at high speeds.

Prior to the 20th century, researchers always assumed that long projectiles flew with small yaw. That is to say, it was assumed that projectiles typically had enough spin such that they would not fly off course in an out-of-plane direction. In the first decades of the 20th century, this assumption was relaxed and researchers began developing ballistic flight models which accounted for yawing motion and gyroscopic stability and are still in use today.

In 1919, English researchers Fowler, Gallop, Lock, and Richmond introduced the modern six-degree-of-freedom (6DOF) rigid aerodynamic force-moment system for spinning projectiles. They also presented several key simplifications to the model which allowed the first approximate analytical solution to be formed. The set of resulting equations became known as projectile linear theory. Small adjustments were made to the 6DOF model over the following years and in 1943 Canadian researchers introduced the complete model in use today.

With the advent of projectile linear theory (PLT), it became possible to design linear feedback controllers which could help in guiding and controlling projectiles to precise target locations. Over recent decades, methods of projectile control have become much more prevalent and sophisticated.

1.2 Common Methods of Projectile Control

In recent years, projectile control has commonly been implemented using model predictive control (MPC) or similar methods. Strategies have included impact point prediction, conversion of plant dynamics to discrete-time systems, and providing reference trajectories to the target.

Ollerenshaw and Costello demonstrated the use of MPC in a 2005 publication [6]. Their method involved minimizing a quadratic cost function defined by a comparison of predicted states and a predetermined reference trajectory. The desired reference trajectory must be loaded into the onboard computer prior to the projectile's launch. It is necessary to convert the dynamics into a discrete-time system and thus a discretization of the solution is required.

Additionally, a 2002 paper by Burchett and Costello made use of a predictive flight control system via MPC [3]. More specifically, an impact point predictor was utilized. Projectile linear theory was used to transform the task of controlling the projectile over the trajectory into one of controlling the impact point in the target plane. Control was based on a comparison of the commanded target location and the predicted impact point in crossrange and altitude. In general, most of the common methods of control require trajectory discretization and pre-loaded trajectories.

1.3 Modified Projectile Linear Theory

Projectile linear theory gives a set of linear ordinary differential equations (ODEs) which can reasonably model the full 6DOF nonlinear set of equations associated with projectile flight. However, a key assumption in PLT involves small Euler pitch angles. This simplification means that the analytical solution to the linear set of equations will deviate from that of the 6DOF set of equations at higher launch angles and longer ranges. Because of this, controlling flight at high pitch angles is more difficult.

In 2005, Hainz and Costello introduced a modified set of assumptions to the 6DOF equations that would produce a set of quasi-linear equations which would better approximate a solution at higher launch angles [4]. The largest difference was the relaxation of the assumption of small Euler pitch angle. It was shown that the modified projectile linear theory equations closely matched the solution to the full non-linear simulation at both low and high launch angles. Consequently, the modified linear equations can be used in a feedback controller to better guide indirect fire launches.

1.4 Introduction to this Work

Recently, Burchett and Nash presented a method for using optimal control to guide the flight of a symmetric projectile [7]. The benefit of this technique is that in part by treating gravity as an uncontrollable mode, a control problem can be formulated without the need for reference trajectories or discrete-time conversions. The result is a continuous-time finite horizon Euler-Lagrange optimal controller. Performance was demonstrated through the use of a time-varying linear optimal regulator in which roll rate and total velocity become time-varying parameters in a piecewise solution.

In this thesis, the work of Burchett and Nash will be extended to incorporate control of indirect fire projectiles via a piecewise time-varying linear optimal regulator. Since optimal control is being utilized, there is still no need for a reference trajectory. This means that control can happen in real time on board the projectile. Rather than implementing control using the traditional projectile linear theory equations, a feedback controller is designed using the modified projectile linear theory equations. The pitch angle will be treated as a time-varying parameter and recursively predicted from current position to target at each time in a control sampling period. A perturbation state will be designed and used as a means to track projectile altitude and control it to a desired value. The combination of these factors allows for improved trajectory control at high launch angles. The results will be demonstrated through impact point dispersion plots and an examination of the controller's defining characteristics.

2. Introduction to Optimal Control

Optimal control is a fundamental control technique which attempts to act such that a system in question behaves in an optimized fashion by getting from a current state to a desired end state with the least effort or cost. A basic criterion, or performance index, is generated and the control law is developed such that the control input given to the system either maximizes or minimizes the chosen performance index. In this chapter, the proper control law for a linear piecewise time-varying (LPTV) optimal regulator will be derived from basic principles of optimal control.

2.1 A Brief History of Optimal Control

The information in this section is adapted from Sussmann and Willems [8]. Conventional belief holds that today's theory of optimal control was born in the Soviet Union in the 1950s. However, the basic principles upon which optimal control was founded were first conceived in the late 17th century in the Netherlands during scientists' attempts to solve the brachystochrone problem which posed a general question about the quickest and most optimal way of getting an object from a start position to a desired end position. Various solutions were presented to the problem in 1697, making it the first known problem to ever deal with dynamical behavior while asking explicitly for an optimal path.

Through the next three centuries, today's familiar principles of optimal control and the calculus of variations were developed by world class scientists such as Euler and Lagrange.

Today, optimal control lies at the heart of control theory and has important influences on the control of a wide variety of objects including guided missiles and projectiles.

2.2 Basic Fundamental Principles of Optimal Control

The following information and derivation is adapted from [9] and [10]. In order to develop a control law to govern a specific system, the process starts with a dynamic system of the most general form:

$$\begin{gathered}\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \\ \boldsymbol{x}(t_i) = \boldsymbol{x}_o\end{gathered} \tag{2.1}$$

where $\boldsymbol{x}(t)$ is the state vector as a function of time t, $\boldsymbol{u}(t)$ is the input to the dynamic system, and $\boldsymbol{x}_o$ is the initial state. It is sought to find the input which will take the system from its initial state to a desired end state $\boldsymbol{x}(t_f)$ in the most optimal way.

To execute this task, a performance criterion, or cost function, must be developed. The performance criterion will be the basis by which the input is judged with respect to its corresponding output. That is to say, the goal will be to minimize the performance index. In its simplest form, this scalar cost function can be written as

$$J = \phi\big(\boldsymbol{x}(t_f), t_f\big) + \int_{t_i}^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, dt \tag{2.2}$$

where ϕ is a scalar function of the dynamic states and time, and L is a scalar function of the states, the input, and time.

A Lagrange multiplier approach [11] is used to append the state dynamics (2.1) as a constraint. The result is given as

$$J = \phi\big(\boldsymbol{x}(t_f), t_f\big) + \int_{t_i}^{t_f} \big[L\big(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), t\big) + \boldsymbol{\lambda}^T(t)\{\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) - \dot{\boldsymbol{x}}\}\big] dt \qquad (2.3)$$

where $\boldsymbol{\lambda}$ is termed the co-state and will aid in the optimization process. Next, a scalar function termed the Hamiltonian is defined as

$$H(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = L(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \boldsymbol{\lambda}^T(t)\boldsymbol{f}(\boldsymbol{x}(\boldsymbol{t}), \boldsymbol{u}(\boldsymbol{t}), t) \qquad (2.4)$$

With the definition of the Hamiltonian in (2.4), the cost function can be written as

$$J = \phi\big(\boldsymbol{x}(t_f), t_f\big) + \int_{t_i}^{t_f} [H(\boldsymbol{x}(t), \boldsymbol{u}(t), t) - \boldsymbol{\lambda}^T(t)\dot{\boldsymbol{x}}]\, dt \qquad (2.5)$$

In order to make (2.5) easier to evaluate, an integration by parts is performed on the $\boldsymbol{\lambda}^T(t)\dot{\boldsymbol{x}}$ term in the integral. Thus, the equation becomes

$$\begin{aligned} J = {} & \phi\big(\boldsymbol{x}(t_f), t_f\big) - \boldsymbol{\lambda}^T\big(t_f\big)\boldsymbol{x}(t_f) + \boldsymbol{\lambda}^T(t_i)\boldsymbol{x}(t_i) \\ & + \int_{t_i}^{t_f} H(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \dot{\boldsymbol{\lambda}}^T(t)\boldsymbol{x}(t)\, dt \end{aligned} \qquad (2.6)$$

To optimize the system, variations in *J* must be considered with respect to variation in input $\boldsymbol{u}$. This is done through a calculus of variations approach. The result is seen in (2.7).

$$\delta J = \left[\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}} - \boldsymbol{\lambda}^T\right]_{t_f} \delta\boldsymbol{x} + [\boldsymbol{\lambda}^T \delta\boldsymbol{x}]_{t_i} + \int_{t_i}^{t_f} \left\{\left[\frac{\partial H}{\partial \boldsymbol{x}} + \dot{\boldsymbol{\lambda}}^T\right]\delta\boldsymbol{x} + \frac{\partial H}{\partial \boldsymbol{u}}\delta\boldsymbol{u}\right\} dt \qquad (2.7)$$

The assumption that J becomes optimized implies that an extremum has been reached. For an optimal performance criterion, $\delta J = 0$. This implies that each term in (2.7) will vanish at the optimal condition. Using this information, the following criteria are developed:

$$\boldsymbol{\lambda}^T(t_f) = \left.\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}}\right|_{t_f} \tag{2.8}$$

$$\dot{\boldsymbol{\lambda}}^T = -\frac{\partial H}{\partial \boldsymbol{x}} \tag{2.9}$$

$$\frac{\partial H}{\partial \boldsymbol{u}} = \boldsymbol{0} \tag{2.10}$$

$$\boldsymbol{\lambda}^T(t_i) = \boldsymbol{0} \tag{2.11}$$

Equations (2.8) - (2.11) are commonly known as the Euler-Lagrange equations in the calculus of variations. From these equations, constraints and conditions for optimality can be formed, and when solved with the system dynamics, the optimal input can be found.

First, (2.9) is expanded and transposed using the definition of the Hamiltonian in (2.4) to obtain

$$\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^T \boldsymbol{\lambda} - \left(\frac{\partial L}{\partial \boldsymbol{x}}\right)^T \tag{2.12}$$

Similarly, (2.10) can be expanded and transposed along with the use of (2.4), resulting in

$$\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^T \boldsymbol{\lambda} + \left(\frac{\partial L}{\partial \boldsymbol{u}}\right)^T = \boldsymbol{0} \tag{2.13}$$

Thus, the problem ultimately entails solving the differential equations given by the state and co-state dynamics:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{2.14}$$

$$\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^T \boldsymbol{\lambda} - \left(\frac{\partial L}{\partial \boldsymbol{x}}\right)^T \tag{2.15}$$

with $\boldsymbol{u}(t)$ determined by evaluating

$$\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^T \boldsymbol{\lambda} + \left(\frac{\partial L}{\partial \boldsymbol{u}}\right)^T = \boldsymbol{0} \tag{2.16}$$

with boundary conditions defined by

$$\boldsymbol{\lambda}(t_f) = \left(\left.\frac{\partial \phi}{\partial \boldsymbol{x}}\right|_{t_f}\right)^T ;\ \boldsymbol{x}(t_i) = \boldsymbol{x_o} \tag{2.17}$$

The preceding process is the basis for optimal control. The process will sufficiently yield a control input $\boldsymbol{u}(t)$ which produces an extremum of a defined performance index J, thus leading the dynamic states $\boldsymbol{x}(t)$ to their desired end conditions.

2.3 Control Law for a Linear Piecewise Time-Varying Optimal Regulator

For the work in this thesis, it is sought to develop a linear optimal regulator that varies with time. This will be done in a piecewise fashion [10], with the trajectory of the projectile being broken into segments from current position to target at each time step in the sampling period. As such, it is desired to make the system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ time-varying. Consider a general dynamic system of linear ODEs with state vector $\boldsymbol{x}$, input $\boldsymbol{u}$ and initial state $\boldsymbol{x_o}$.

$$\begin{gathered}\dot{\boldsymbol{x}}(s) = \boldsymbol{A}(s)\boldsymbol{x}(s) + \boldsymbol{B}(s)\boldsymbol{u}(s) \\ \boldsymbol{x}(s=0) = \boldsymbol{x}_o\end{gathered} \tag{2.18}$$

Here, the system is not a function of time, but of downrange calibers s and $\dot{\boldsymbol{x}}(s)$ represents a caliber derivative. The process will remain the same; an independent variable change has simply been made. The reasoning for the change of variables will be further discussed in Chapter 4.

The goal is to control a linear combination of the states over a caliber interval $\{s_i, s_f\}$. Thus, let matrix transformation $\boldsymbol{Z}$ be defined as

$$\boldsymbol{Z}(s) = \boldsymbol{E}(s)\boldsymbol{x}(s) \tag{2.19}$$

The performance index for this system can be temporarily written as

$$J = \frac{1}{2}\boldsymbol{x}^T(s_f)\boldsymbol{S}\boldsymbol{x}(s_f) + \int_{s_i}^{s_f} \frac{1}{2}(\boldsymbol{Z}^T(s)\boldsymbol{T}\boldsymbol{Z}(s) + \boldsymbol{u}^T(s)\boldsymbol{R}\boldsymbol{u}(s))ds \tag{2.20}$$

where $\boldsymbol{S}, \boldsymbol{R}$ and $\boldsymbol{T}$ are weighting matrices and are required only to be positive, semi-definite matrices. By defining another weighting matrix $\boldsymbol{Q}(s) = \boldsymbol{E}^T(s)\boldsymbol{T}\boldsymbol{E}(s)$, the performance index can be re-written as

$$J = \frac{1}{2}\boldsymbol{x}^T(s_f)\boldsymbol{S}\boldsymbol{x}(s_f) + \int_{s_i}^{s_f} \frac{1}{2}(\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u})ds \tag{2.21}$$

By matching terms with (2.2), it can be said that

$$\phi = \frac{1}{2}\boldsymbol{x}^T(s_f)\boldsymbol{S}\boldsymbol{x}(s_f) \tag{2.22}$$

and

$$L = \frac{1}{2}(\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}) \tag{2.23}$$

In accordance with (2.4), the scalar Hamiltonian H is defined as

$$H = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}) \tag{2.24}$$

By utilizing the Euler-Lagrange equations for optimal control, the derivative of the co-state can be developed as

$$\dot{\boldsymbol{\lambda}} = -\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{\lambda} \tag{2.25}$$

The control $\boldsymbol{u}$ can be found by taking the partial derivative of H with respect to $\boldsymbol{u}$ as seen in (2.26).

$$\frac{\partial}{\partial \boldsymbol{u}}\left(\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}\right) + \frac{\partial}{\partial \boldsymbol{u}}(\boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u})) = \boldsymbol{0} \tag{2.26}$$

Evaluating (2.26) with $\boldsymbol{Q} = \boldsymbol{0}$ yields the correct control law for this problem.

$$\boldsymbol{u}^T\boldsymbol{R} + \boldsymbol{\lambda}^T\boldsymbol{B} = \boldsymbol{0} \tag{2.27}$$

A simple rearrangement makes it possible to solve directly for the control input as a function of the system's control matrix $\boldsymbol{B}$ and the co-state $\boldsymbol{\lambda}$:

$$\boldsymbol{u} = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\boldsymbol{T}}\boldsymbol{\lambda} \tag{2.28}$$

Additionally, the set of differential equations necessary to evaluate the control effort can be written in matrix form as

$$\begin{Bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{\lambda}} \end{Bmatrix} = \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\boldsymbol{T}} \\ -\boldsymbol{Q} & -\boldsymbol{A}^{\boldsymbol{T}} \end{bmatrix} \begin{Bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{Bmatrix} \tag{2.29}$$

Through solving the sets of ODEs given in (2.29) and evaluating (2.28), the control input which optimizes the system can be found at each step in the control sampling period. In order to accomplish this, a way of solving for the time-varying co-state is needed.

2.4 Development of LPTV Control Law via Matrix Riccati Differential Equation

In order to account for the time-varying nature of the system matrices desired in this problem, an alternative form of (2.28) is considered. From the boundary condition given in (2.8) and the definition of ϕ in (2.22), it is known that at the final state, the co-state $\boldsymbol{\lambda}$ is related to the system states $\boldsymbol{x}$ by a gain matrix $\boldsymbol{S}$. It can then be inferred that the co-state is related to the state vector by a time-varying mapping matrix $\boldsymbol{N}$, as seen in (2.30).

$$\boldsymbol{\lambda}(s_f) = \boldsymbol{S}\boldsymbol{x}(s_f) \;\; \Rightarrow \;\; \boldsymbol{\lambda}(s) = \boldsymbol{N}(\mathrm{s})\boldsymbol{x}(s) \tag{2.30}$$

Using this information and (2.28), the equation for the control input can be re-written as

$$\boldsymbol{u}(s) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\boldsymbol{T}}(s)\boldsymbol{N}(s)\boldsymbol{x}(s) \tag{2.31}$$

In order to solve for the optimal control input, a solution to $\boldsymbol{N}$ must be found. It is worth noting that the matrices in the following derivation are functions of arc-length *s* which has units of calibers. For convenience, the *s* will be left out of the next few equations. Because both $\boldsymbol{N}$ and $\boldsymbol{x}$ are time-varying, the derivative of (2.30) can be written as

$$\dot{\boldsymbol{\lambda}} = \dot{\boldsymbol{N}}\boldsymbol{x} + \boldsymbol{N}\dot{\boldsymbol{x}} \tag{2.32}$$

Substituting (2.32) into the definition of the co-state derivative given in (2.25), it is found that

$$\dot{\boldsymbol{N}}\boldsymbol{x} + \boldsymbol{N}\dot{\boldsymbol{x}} = -\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{\lambda} \tag{2.33}$$

Making use of (2.29), this can be further developed as

$$\boldsymbol{N}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{B}\boldsymbol{R}^{-\mathbf{1}}\boldsymbol{B}^{\boldsymbol{T}}\boldsymbol{\lambda}) + \dot{\boldsymbol{N}}\boldsymbol{x} = -\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{\lambda} \tag{2.34}$$

Equation (2.34) can be expanded and rearranged. First, the substitution $\boldsymbol{\lambda} = \boldsymbol{N}\boldsymbol{x}$ is made, which provides an equation in which $\boldsymbol{x}$ is present in every term. This implies that the equation is valid for any $\boldsymbol{x}$. Thus, equation (2.34) can be written as

$$\dot{\boldsymbol{N}}(s) = -\boldsymbol{N}(s)\boldsymbol{A}(s) - \boldsymbol{A}^T(s)\boldsymbol{N}(s) + \boldsymbol{N}(s)\boldsymbol{B}(s)\boldsymbol{R}^{-1}\boldsymbol{B}^{\boldsymbol{T}}(s)\boldsymbol{N}(s) - \boldsymbol{Q} \tag{2.35}$$

Equation (2.35) is known as the matrix Riccati differential equation. It is one of the most fundamental equations in control theory. In this thesis, it will be used to solve for the control input $\boldsymbol{u}$ which will help guide the projectile to the desired target location. After a solution to (2.35) is obtained, the control input $\boldsymbol{u}$ can be backed out via (2.31).

There are multiple techniques used to solve for the gain matrix $\boldsymbol{N}.$ The technique used for solving the Riccati equation in this thesis involves the use of block pulse functions and requires the evaluation of a matrix Hamiltonian with a reverse integration process to back propagate for the correct solution [12]. This will be covered in a later chapter about the development of the controller.

3. Projectile Dynamic System

This chapter introduces the dynamic equations needed to run a full simulation of projectile flight. The dynamic description of the projectile involves 12 highly nonlinear ODEs which describe its location and attitude in an inertial reference frame. Initially, the forces and moments acting on the projectile will be discussed and presented. The dynamic equations of the projectile itself will then be presented, followed by a description of the control mechanism, the forward-mounted controllable canards. Finally, the dynamic equations for modeling canard behavior will also be presented.

3.1 Aerodynamic Forces and Moments Acting on a Projectile in Flight

The information and figures in this section have been adapted from the text of McCoy [5]. In modeling the flight of a projectile, the external forces and moments acting on the system are of great importance. These forces and moments alter the flight dynamics from launch to target, and must be precisely accounted for. Aerodynamic forces and moments are characterized by coefficients which are obtained experimentally as functions of local Mach number. As the projectile's mass center velocity changes, so too does the Mach number. Consequently, the coefficients vary throughout the flight and should be properly incorporated.

One of the most prominent forces acting on the projectile throughout its flight is termed the drag force. The aerodynamic drag force is depicted in Figure 3.1.

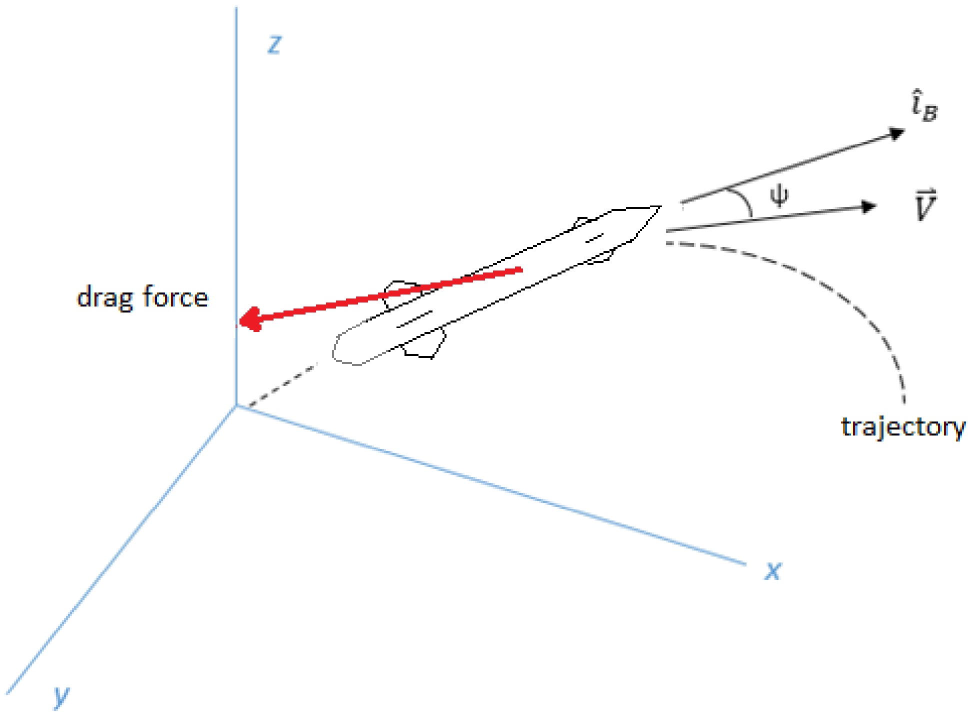


Figure 3.1: Depiction of aerodynamic drag force

Aerodynamic drag directly resists the total velocity vector of the projectile and accounts for yawing motion (out-of plane) by allowing for variation of the total yaw angle ψ. Two terms, C_{D0} and C_{D2}, are used to calculate the total drag force coefficient. In general, the total drag force coefficient is found as

$$C_D = C_{D0} + C_{D2}\psi^2 \tag{3.1}$$

and the total magnitude of the drag force would be represented by

$$F_D = \frac{1}{2}\rho V^2 S C_D \tag{3.2}$$

where ρ is the air density, V is the magnitude of the total mass center velocity of the projectile, and S is the projectile reference area with D being the projectile reference diameter.

Another primary force acting on the projectile throughout its flight is the lift force. The aerodynamic lift force is depicted in Figure 3.2.

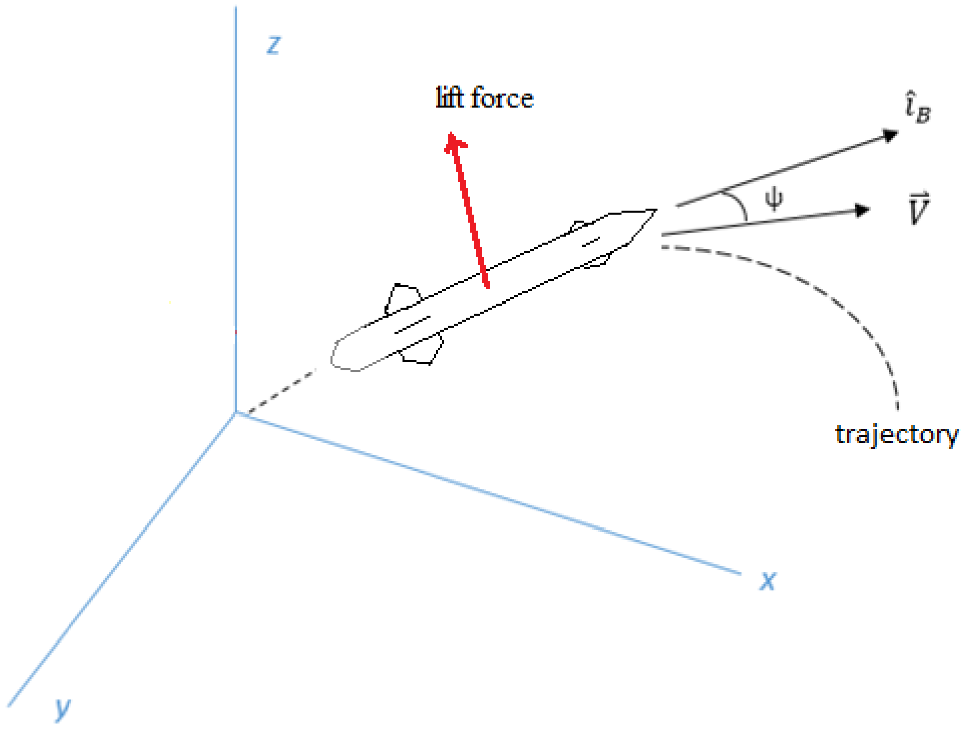


Figure 3.2: Depiction of aerodynamic lift force

The lift force acts perpendicular to the trajectory. Like the drag force coefficient, the lift coefficient also varies with yaw and is represented by

$$C_L = C_{L0} + C_{L2}\psi^2 \tag{3.3}$$

where C_{L0} and C_{L2} are Mach number dependent. The total lift force is represented by

$$F_L = \frac{1}{2}\rho V^2 S C_L \sin(\psi) \tag{3.4}$$

Oftentimes for ease of computation, authors will work in a body frame with axes parallel and perpendicular to the projectile's axis of symmetry. In this type of axis system, the lift and drag forces are resolved into axial (F_X) and normal (F_N) forces, seen in Figure 3.3.

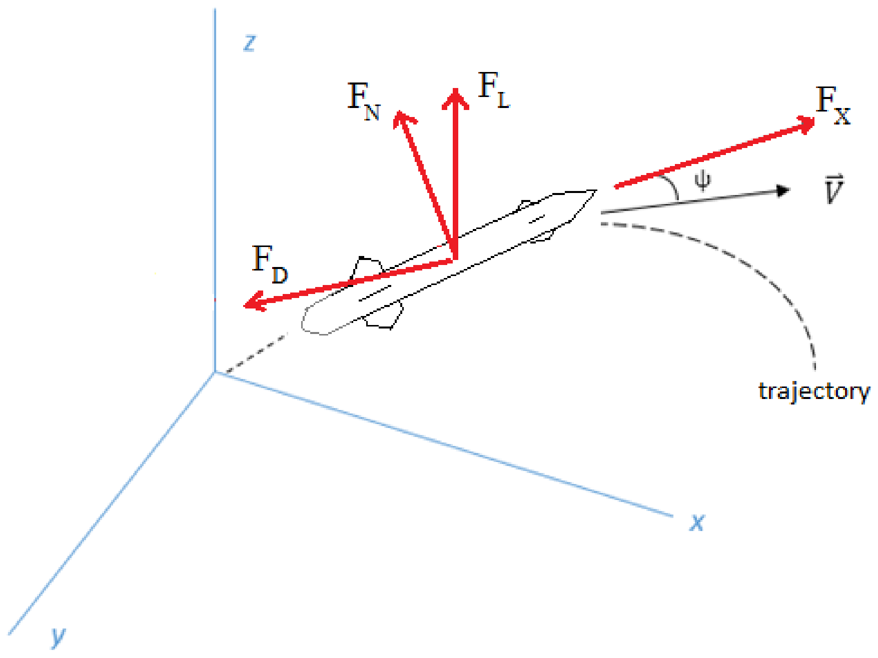


Figure 3.3: Depiction of axial and normal forces

The resulting normal force aerodynamic coefficient is termed C_{NA} and the magnitude of the normal force is calculated as

$$F_N = \frac{1}{2}\rho V^2 S C_{NA} \sin(\psi) \tag{3.5}$$

Under small yaw angles, the axial force and drag force will act in exactly opposite directions. Therefore, for small yaw,

$$C_X = -C_D \Rightarrow C_{X0} = -C_{D0}\ ;\ C_{X2} = -C_{D2} \tag{3.6}$$

and the total magnitude of the axial force is seen as

$$F_X = \frac{1}{2}\rho V^2 S C_X \tag{3.7}$$

Several moments also act on the projectile during its flight. One of these moments is termed the spin damping moment, and it always opposes the spin of the projectile and decreases axial spin, or roll. The spin damping moment is depicted in Figure 3.4.

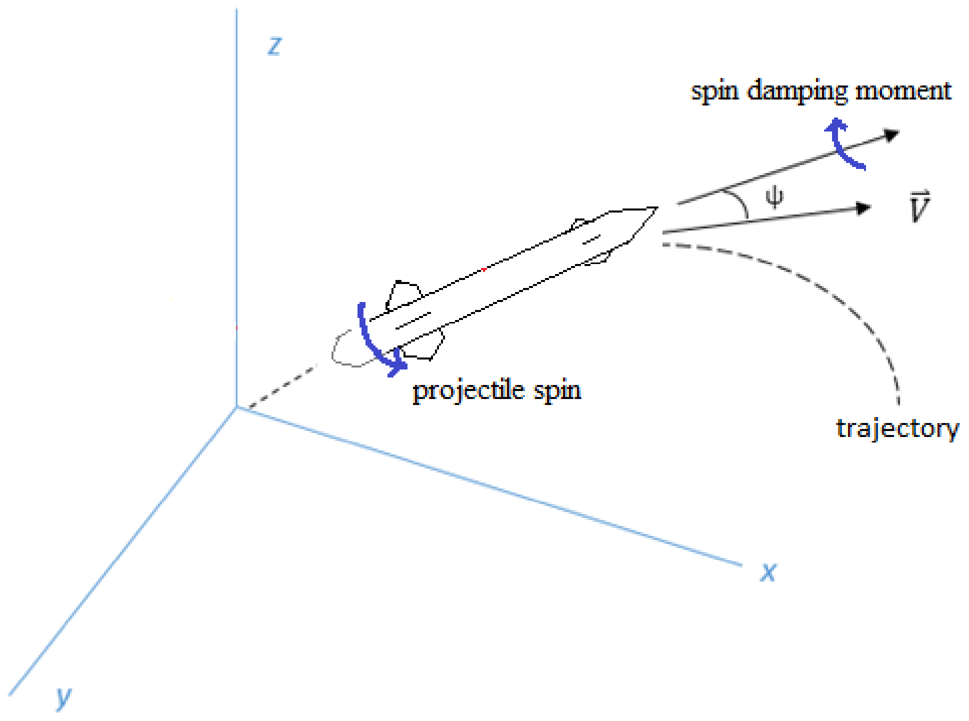


Figure 3.4: Depiction of spin damping moment

The spin damping moment coefficient is always negative and is represented by C_{LP}. The magnitude of the spin damping moment on a projectile is calculated by

$$M_{SD} = \frac{1}{2}\rho V^2 SD\left(\frac{pD}{V}\right)C_{LP} \tag{3.8}$$

where p represents the axial spin rate.

Another moment acting on the projectile during flight is a rolling moment due to fin cant. This applies only to finned missiles with differentially canted fins. This moment tends to cause increasing spin at the exact time the spin damping moment is causing decreasing spin. The two moments typically serve to cancel each other out and cause the spin rate to approach a small magnitude. The fin cant rolling moment is calculated by

$$M_{FCR} = \frac{1}{2}\rho V^2 SD\delta_F C_{DD} \qquad (3.9)$$

where δ_F is the fin cant angle and C_{DD} is the fin cant rolling moment coefficient.

A third moment acting on the projectile during flight is the pitch damping moment. This is a moment resulting from the aerodynamic lift force. The pitch damping moment is depicted in Figure 3.5.

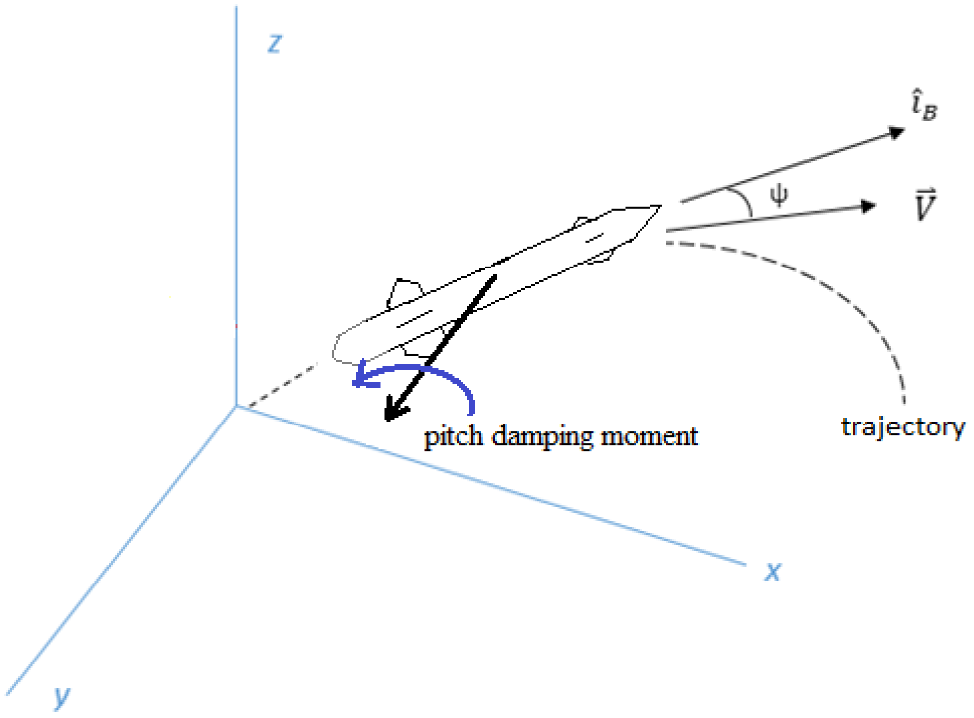


Figure 3.5: Depiction of pitch damping moment

The coefficient for the pitch damping moment is C_{MQ} and the moment is represented by

$$M_{PD} = \frac{1}{2}\rho V^2 SD\left(\frac{q_t D}{V}\right) C_{MQ} \qquad (3.10)$$

where q_t is the total transverse angular velocity.

Another force and moment pair typically acting on a projectile during flight deals with the Magnus effect. The Magnus force is due to uneven pressure forces on opposite sides of a spinning body. Due to the presence of differentially canted fins on the projectile in this thesis, a fin cant rolling moment is present and serves to oppose the spin damping moment. This causes the axial spin rate to be small enough in magnitude that the Magnus effect will be assumed to be negligible throughout the duration of this thesis.

3.2 Coordinate Systems and Reference Frames

This section is adapted from [13]. In this thesis, several reference frames and coordinate systems are used to help simulate the flight of the projectile and to aid in ensuring a computationally efficient controller. In general, the position and attitude states are derived in the inertial frame. Commonly, the inertial frame is placed such that its x-axis runs through the center of the target and the z-axis points positive downward. Another reference frame, termed the body frame, is frequently used in aerodynamic modeling. The body frame is placed such that its x-axis remains parallel to the stationline axis of the projectile. The coordinate systems are right-handed. The representation of the two frames and coordinate systems is seen in Figure 3.6.

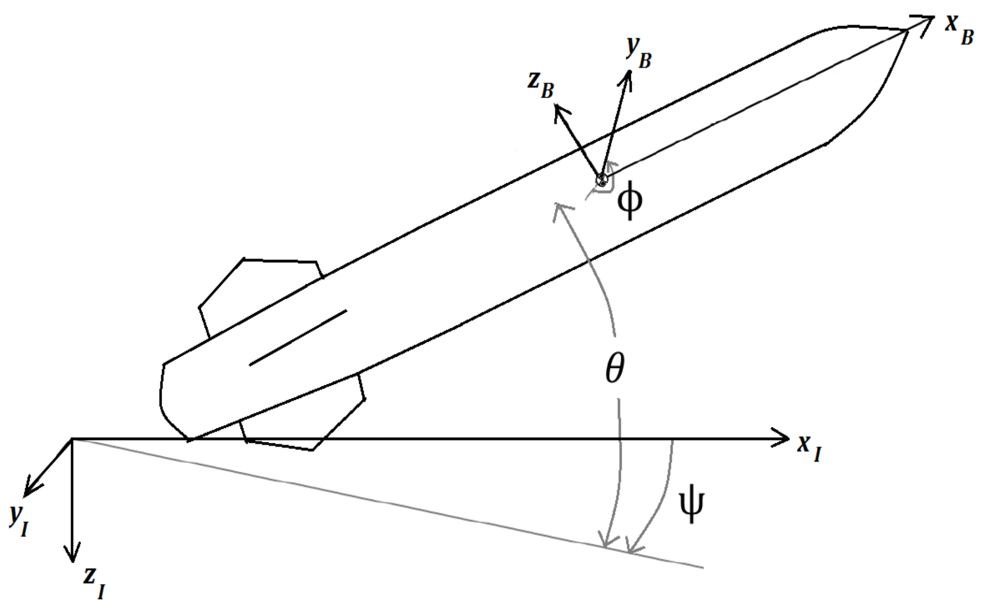


Figure 3.6: Relationship between inertial and body frames (adapted from *[13]*)

The two frames are related by the standard (Z-Y-X) aerospace rotation sequence where $\psi, \theta,$ and ϕ are the Euler angles of rotation, and the transformation is represented by

$$\boldsymbol{R}_{BI} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{3.11}$$

It is worth noting that in all equations presented, the notation for trigonometric functions is: $\sin(\alpha) \equiv s_\alpha$, $\cos(\alpha) \equiv c_\alpha$, $\tan(\alpha) \equiv t_\alpha$ for any angle α.

Another set of coordinates, fixed-plane coordinates, are often used with the body frame. The fixed-plane coordinate system is attached to the body of the projectile with the y-axis remaining parallel to the ground at all times. The relationship between the inertial and fixed-plane coordinate systems is given as

$$\boldsymbol{R}_{FPI} = \begin{bmatrix} c_\theta c_\psi & -s_\psi & s_\theta c_\psi \\ c_\theta s_\psi & c_\psi & s_\theta s_\psi \\ -s_\theta & 0 & c_\theta \end{bmatrix} \tag{3.12}$$

Additionally, the relationship between body-fixed and fixed-plane coordinates can be shown to be

$$\boldsymbol{R}_{BFP} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix} \tag{3.13}$$

and Figure 3.7 depicts the relationship graphically.

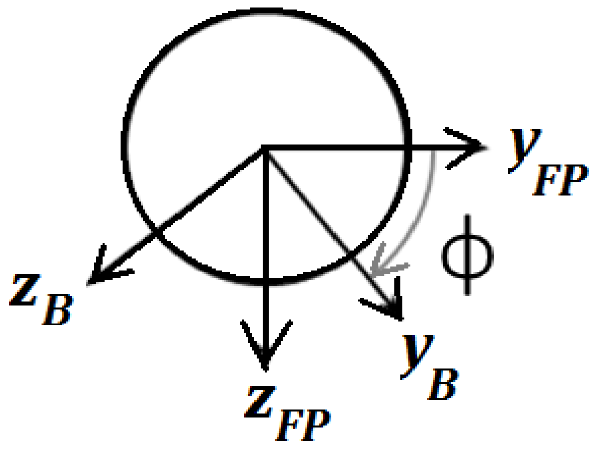


Figure 3.7: Relationship between fixed-plane and body frames (adapted from *[13]*)

The fixed-plane coordinate system offers computational advantages and thus will be used at times, especially in the development of the controller.

3.3 Six-Degree-of-Freedom Projectile Dynamic Equations

In order to model the flight dynamics of the projectile, a six-degree-of-freedom (6DOF) rigid model is utilized. The six degrees of freedom are depicted in Figures 3.8 and 3.9 and comprise the three positional coordinates of the projectile's mass center (x, y, z) and the three Euler angles describing the attitude of the projectile with respect to an inertial reference frame (ψ, θ, ϕ).

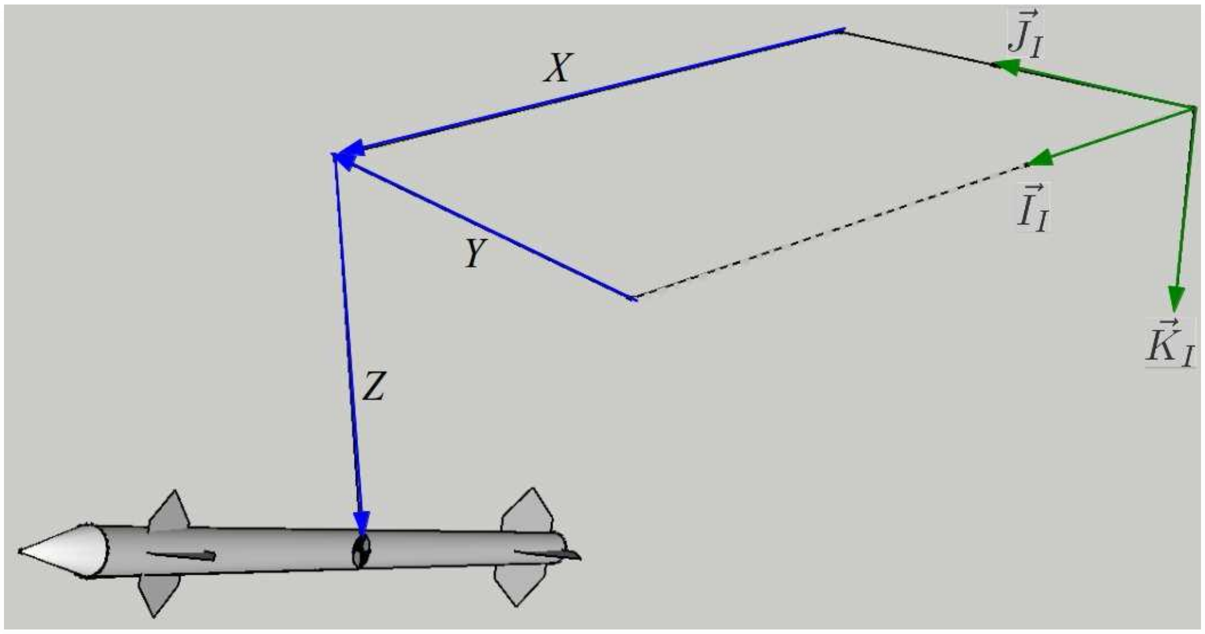


Figure 3.8: Translational degrees of freedom

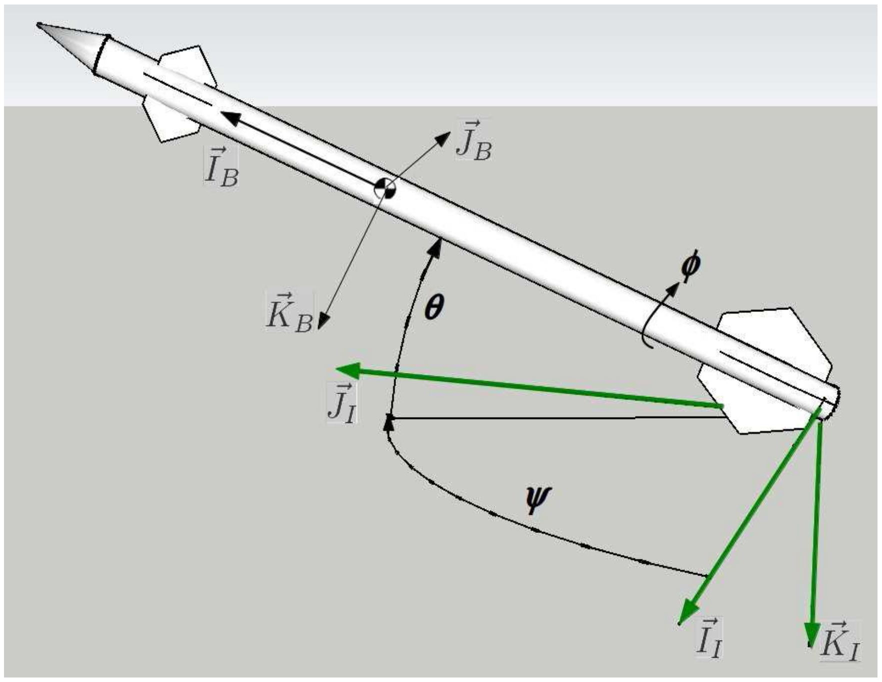


Figure 3.9: Rotational degrees of freedom

As previously stated, this thesis will work in multiple reference frames to deal with the equations of motion. The projectile plant dynamics are derived in the roll frame using body-fixed coordinates. Presented in a subsequent chapter, the modified linear projectile dynamics, which are used to implement feedback control, are derived in the no-roll frame which uses fixed-plane coordinates.

The twelve states used in the projectile dynamic model are presented in Table 3.1.

Table 3.1: Twelve states involved in the projectile dynamic model

State	Description
x	position in inertial $\vec{I}_I$ dimension; downrange
y	position in inertial $\vec{J}_I$ dimension; crossrange
z	position in inertial $\vec{K}_I$ dimension; negative of altitude
ψ	yaw angle
θ	pitch angle
ϕ	roll angle
u	$\vec{I}_B$ translational velocity component
v	$\vec{J}_B$ translational velocity component
w	$\vec{K}_B$ translational velocity component
p	roll rate
q	pitch rate
r	yaw rate

The nonlinear dynamic equations representing these states are given as follows, with m being projectile mass, $\boldsymbol{I}$ being the inertia matrix, and X, Y, Z and L, M, N denoting force and moment components on the projectile mass center:

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{3.14}$$

$$\begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} \tag{3.15}$$

$$\begin{Bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{Bmatrix} = \frac{1}{m}\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} - \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{3.16}$$

$$\begin{Bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{Bmatrix} = [\boldsymbol{I}]^{-1}\left[\begin{Bmatrix} L \\ M \\ N \end{Bmatrix} - \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}[\boldsymbol{I}]\begin{Bmatrix} p \\ q \\ r \end{Bmatrix}\right] \tag{3.17}$$

As seen in (3.14) – (3.17), several intermediate variables must be calculated in order to compute the set of state derivatives at any point in time. These intermediate variables are due to the aerodynamic forces and moments discussed earlier and due to the gravitational weight force of the projectile.

In order to obtain the total force acting on the projectile, contributions from weight (W) and body aerodynamics (A) must be calculated. Thus, the total force is given by

$$\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} = \begin{Bmatrix} X_w \\ Y_w \\ Z_w \end{Bmatrix} + \begin{Bmatrix} X_A \\ Y_A \\ Z_A \end{Bmatrix} \tag{3.18}$$

Since the dynamic equations are being expressed in a body-fixed reference frame, the forces acting on the body are represented in a rocket reference frame. The components of the projectile's weight are

$$\begin{Bmatrix} X_w \\ Y_w \\ Z_w \end{Bmatrix} = mg\begin{Bmatrix} -s_\theta \\ s_\phi c_\theta \\ c_\phi c_\theta \end{Bmatrix} \tag{3.19}$$

The total aerodynamic force on the rocket acts at the aerodynamic center of pressure and its components are

$$\begin{Bmatrix} X_A \\ Y_A \\ Z_A \end{Bmatrix} = -\frac{\pi}{8}\rho V^2 D^2 \begin{Bmatrix} C_{X0} + C_{X2}(v^2 + w^2)/V^2 \\ C_{NA} v/V \\ C_{NA} w/V \end{Bmatrix} \tag{3.20}$$

Obtaining the total moment acting on the projectile involves modeling contributions from both steady aerodynamics (SA) and unsteady aerodynamics (UA). The total moment is given by

$$\begin{Bmatrix} L \\ M \\ N \end{Bmatrix} = \begin{Bmatrix} L_{SA} \\ M_{SA} \\ N_{SA} \end{Bmatrix} + \begin{Bmatrix} L_{UA} \\ M_{UA} \\ N_{UA} \end{Bmatrix} \tag{3.21}$$

The moment contribution due to steady aerodynamics is a cross-product between a vector from the center of gravity to the projectile center of pressure and the aerodynamic force. Using a skew-symmetric matrix approach, this cross product is written as

$$\begin{Bmatrix} L_{SA} \\ M_{SA} \\ N_{SA} \end{Bmatrix} = \begin{bmatrix} 0 & -R_{CAZ} & R_{CAY} \\ R_{CAZ} & 0 & -R_{CAX} \\ -R_{CAY} & R_{CAX} & 0 \end{bmatrix} \begin{Bmatrix} X_A \\ Y_A \\ Z_A \end{Bmatrix} \tag{3.22}$$

It is worth noting that in simulations used in this thesis, R_{CAZ} and R_{CAY} are both assumed to be zero due to symmetry. The value of R_{CAX} varies with Mach number due to the variation in the center of pressure. The UA moment acting on the projectile is given by

$$\begin{Bmatrix} L_{UA} \\ M_{UA} \\ N_{UA} \end{Bmatrix} = \frac{\pi}{8}\rho V^2 D^3 \begin{Bmatrix} C_{DD} + \dfrac{pDC_{LP}}{2V} \\ \dfrac{qDC_{MQ}}{2V} \\ \dfrac{rDC_{MQ}}{2V} \end{Bmatrix} \tag{3.23}$$

In the previous dynamic equations, *V* is the magnitude of the total velocity of the projectile mass center and is calculated with a root-sum-square of each of the three translational velocity components. Each of the aerodynamic coefficients is local Mach number dependent, while air density ρ (slug/ft^3) and speed of sound c (ft/s) are altitude dependent. At each step in the simulation, the density and speed of sound must be calculated according to

$$\rho(a) = \begin{cases} 0.00238(1 - 6.88(10^{-6})a)^{4.258}, & a < 35332 \text{ ft} \\ 0.000727e^{-0.0000478(a-35332)}, & a \geq 35332 \text{ ft} \end{cases}$$
$$c(a) = \begin{cases} 49.0124\sqrt{518.4 - 0.003566a}, & a < 35332 \text{ ft} \\ 970.90, & a \geq 35332 \text{ ft} \end{cases} \tag{3.24}$$

where *a* represents the current altitude and is the negative of state variable *z* due to the assumption that *z* is positive down in the model. With the proper values of the total speed and the speed of sound, the Mach number can correctly be computed as

$$Ma = \frac{V}{c} \tag{3.25}$$

Then, the values of the aerodynamic coefficients are linearly interpolated from the current Mach number. The table values for projectile aerodynamic coefficients are displayed in Appendix B.

In order to simulate these dynamic equations, a numerical integration scheme is utilized. The scheme used in this thesis is a 4th order, variable-step Runge-Kutta algorithm implemented using the *ode45* function in MATLAB. The basic process of unguided dynamic simulation can be summarized in a six step process.

1.) Compute the magnitude of the velocity of the projectile mass center at the current time
2.) Compute the values of ρ, c, Ma based on current altitude and total velocity
3.) Using linear interpolation, find the aerodynamic coefficients for the current step
4.) Compute the total force and moment acting on the projectile

5.) Compute the state derivatives

6.) Iterate in time, repeating steps 1-5 until final state or terminal condition is reached

The previous set of equations will model unguided projectile dynamic flight. In order to control the flight, nonlinear actuators shall be installed and modeled.

3.4 Introducing Canards as Nonlinear Actuators

The mechanism for control in this thesis consists of two pairs of forward-mounted controllable canards. The canards are mounted toward the nose of the missile, as seen in Figure 3.10. They are then oriented by the control input throughout flight. It is possible to use the techniques of optimal control presented in Chapter 2 such that control can orient the canards in a way that optimizes the flight path of the projectile.

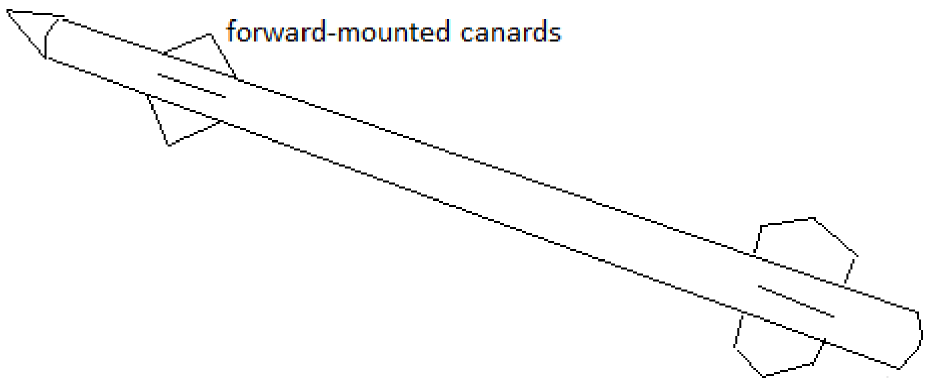


Figure 3.10: Depiction of canard actuators

Taking a closer look at the canards themselves, several angles and properties must be defined. The angles include a sweep angle (γ_c), pitch angle (δ_c), and a roll angle (ϕ_c), and are depicted in Figure 3.11. The angles represent the standard *Z-Y-X* rotations and help describe the canard orientation in a canard frame with respect to the body frame. The k-axis of the canard frame is aligned with the k-axis of the body frame, and therefore the values of γ_c are zero for each canard.

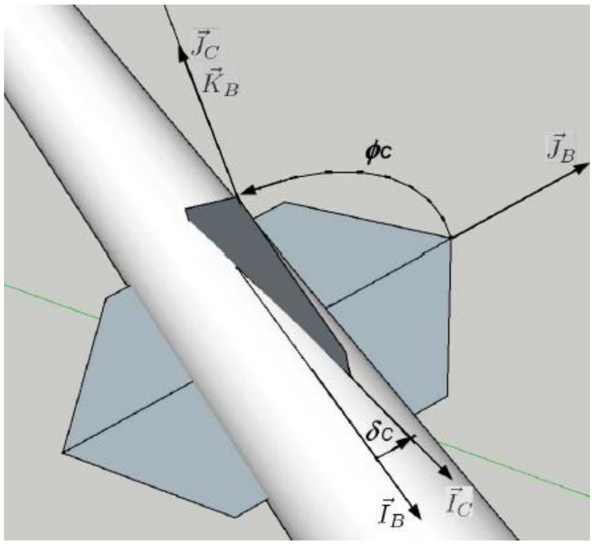


Figure 3.11: Canard angles and their orientations

Additionally, the canards have reference areas represented by S_c and vector centers of pressure represented by $\{SL_c\ BL_c\ WL_c\}$. Canard angles and properties are presented in Appendix D.

From Chapter 2, it was shown that a control input $\boldsymbol{u}$ would be computed in order to minimize the cost function and lead the projectile on its desired path. That control input corresponds to the canard pitch angle, δ_C. The canard pitch angles will be the output of the feedback control system. The canards will then be oriented according to this output, causing a change in the aerodynamic forces and moments on the system and thus altering the flight. Therefore, the overall force and moment exerted on the projectile by the canards must be determined.

There are two pairs of canards, giving a total of four canards. The pitch angles of canards one and three are the output of the controller. However, the angles will be output in the no-roll frame (NR). In order to match with the plant dynamics, they must be converted into the roll frame (R). This is accomplished by the transformation

$$\begin{Bmatrix} \delta_1 \\ \delta_3 \end{Bmatrix}_R = \begin{bmatrix} c_\phi & s_\phi \\ -s_\phi & c_\phi \end{bmatrix} \begin{Bmatrix} \delta_1 \\ \delta_3 \end{Bmatrix}_{NR} \tag{3.26}$$

Symmetric deflections are assumed such that

$$\begin{aligned} \delta_2 &= -\delta_1 \\ \delta_4 &= -\delta_3 \end{aligned} \tag{3.27}$$

The process begins by computing the position vector components from the mass center (CG) of the projectile to the center of pressure of the *i*th canard (C) according to

$$\begin{aligned} r_{x,i} &= SL_i - SL_{cg} \\ r_{y,i} &= BL_i - BL_{cg} \\ r_{z,i} &= WL_i - WL_{cg} \end{aligned} \tag{3.28}$$

The vector components are then used to compute the velocity of the canard itself:

$$\vec{V}_C = \vec{V}_{CG} + \omega \times \vec{r}_{C/CG} \tag{3.29}$$

Using a skew-symmetric matrix operation and resolving the velocity vector into three translational components, this cross product can be written as

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix}_{\tilde{C},i} = \begin{Bmatrix} u \\ v \\ w \end{Bmatrix}_{CG} + \begin{bmatrix} 0 & r_{z,i} & -r_{y,i} \\ -r_{z,i} & 0 & r_{x,i} \\ r_{y,i} & -r_{x,i} & 0 \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} \tag{3.30}$$

The velocities then need to be rotated into the canard frame. This is done through a ϕ_c, γ_c rotation sequence. In matrix form, this transformation is

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix}_{C,i} = \begin{bmatrix} c_{\gamma,i} & c_{\phi,i}s_{\gamma,i} & s_{\phi,i}s_{\gamma,i} \\ -s_{\gamma,i} & c_{\phi,i}c_{\gamma,i} & s_{\phi,i}c_{\gamma,i} \\ 0 & -s_{\phi,i} & c_{\phi,i} \end{bmatrix}_C \begin{Bmatrix} u \\ v \\ w \end{Bmatrix}_{\tilde{C},i} \tag{3.31}$$

The aim is to use the canard velocities to find the aerodynamic drag coefficients associated with the canard and thus the aerodynamic forces and moments associated with the canard. To do this, another variable, canard angle of attack α_C, is needed. This calculation is given by

$$\alpha_{C,i} = \delta_{C,i} + \tan^{-1}\left(\frac{w_{C,i}}{u_{C,i}}\right) \tag{3.32}$$

and the Mach number can be found using (3.33).

$$Ma_{C,i} = \frac{\sqrt{{u_{C,i}}^2 + {w_{C,i}}^2}}{\text{speed of sound}} \tag{3.33}$$

Using the Mach number of the canard, the lift and drag coefficients can be found through a linear interpolation method. The total lift and drag coefficients are functions of several different intermediate coefficients and the variable α_C. The total lift and drag coefficients of the canard are given by

$$C_L = C_{L1,C,i}\alpha_{C,i} + C_{L3,C,i}{\alpha_{C,i}}^3 + C_{L5,C,i}{\alpha_{C,i}}^5 \tag{3.34}$$

$$C_D = C_{D0,C,i} + C_{D2,C,i}{\alpha_{C,i}}^2 + C_{I,C,i}{C_{L,C,i}}^2 \tag{3.35}$$

where C_L and C_D represent the lift and drag coefficients for the ith canard and all other coefficients are table lookups based on canard Mach number and are displayed in Appendix C. Using the lift and drag coefficients, the lift and drag forces on the individual canard can be computed as

$$FL_{C,i} = \frac{1}{2}\rho\left({u_{C,i}}^2 + {w_{C,i}}^2\right)S_{C,i}C_{L,C,i} \tag{3.36}$$

$$FD_{C,i} = \frac{1}{2}\rho\left({u_{C,i}}^2 + {w_{C,i}}^2\right)S_{C,i}C_{D,C,i} \tag{3.37}$$

These forces are then resolved into the canard frame using

$$\begin{gathered} x_{C,i} = FL_{C,i}\sin\left(\alpha_{C,i} - \delta_{C,i}\right) - FD_{C,i}\cos\left(\alpha_{C,i} - \delta_{C,i}\right) \\ y_{C,i} = 0 \\ z_{C,i} = -FL_{C,i}\cos\left(\alpha_{C,i} - \delta_{C,i}\right) - FD_{C,i}\sin\left(\alpha_{C,i} - \delta_{C,i}\right) \end{gathered} \tag{3.38}$$

Finally, the force components are transformed back into the body frame via

$$\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_{C,i} = \begin{bmatrix} c_{\gamma,i} & 0 & s_{\gamma,i} \\ s_{\phi,i}s_{\gamma,i} & 0 & -s_{\phi,i}c_{\gamma,i} \\ -c_{\phi,i}s_{\gamma,i} & 0 & c_{\phi,i}c_{\gamma,i} \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{C,i} \tag{3.39}$$

In order to calculate the moment of canard i, a vector cross product is utilized similar to that in (3.29). Thus, the moment due to the canard can be computed by

$$\begin{Bmatrix} L \\ M \\ N \end{Bmatrix}_{C,i} = -\begin{bmatrix} 0 & r_{z,i} & -r_{y,i} \\ -r_{z,i} & 0 & r_{x,i} \\ r_{y,i} & -r_{x,i} & 0 \end{bmatrix} \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_{C,i} \tag{3.40}$$

This process computes the force and moment exerted on the projectile mass center by a single canard. The process is repeated four times, once for each individual canard. In the end, each of the contributions is summed, yielding a total force and a total moment given by the canards. The respective results are then added to (3.18) and (3.21).

The process of computing the force and moment contributions of the canards can be summarized in a five step process.

1.) Rotate δ_c from controller into the roll frame
2.) Initialize the force and moment reactions to zero
3.) Loop trough the four canards, computing the force/moment contribution of each
    a. Calculate vector distances from center of mass to canard center of pressure
    b. Compute total canard velocity
    c. Compute canard Mach number and determine lift/drag coefficient values
    d. Calculate lift and drag forces due to i^{th} canard
    e. Calculate force and moment on projectile due to i^{th} canard
    f. Resolve force and moment into the body frame
4.) Sum contributions from each canard to get total forces/moments from canards
5.) Add the total reactions to the force/moment equations in the plant dynamics

It should be mentioned that for all uncontrolled shots, each canard pitch angle δ_c is held constant at 0°. By incorporating the reactions due to the canard actuators, their orientation can be commanded and modeled to guide the flight. The commanded orientation given to the canards shall be discussed in Chapter 4.

4. Implementing Control

In this chapter, the process of calculating the correct control input will be developed. As previously stated, the control inputs in this thesis are the canard deflections δ_c. The controller will be given the current set of states and a desired target and will compute a set of angles that will correctly orient the fins to drive the projectile to its target location. In order to do this, a linear, or quasi-linear, set of states is desired. Projectile linear theory gives a basic linear set of states which work well in controlling direct fire launches. However, the assumption of small Euler pitch angle prevents the set of equations from accurately modeling high launch or long range shots. Thus, a modified linear set of equations shall be used to develop a controller which can accurately guide the projectile toward a downrange target at higher (20°-60°) launch angles.

In order to accomplish this task, the modified projectile linear theory (MPLT) equations must be set up such that the nonlinearities are essentially removed. A point mass vacuum trajectory model will be used to help model a few states that must be removed from the full state model due to the nonlinearities they cause and to give the projectile a theoretical way of successfully getting from its current position to the target. Furthermore, general linearization using a Taylor series expansion will be used to create an additional path between the control input and the altitude state derivative equation and to help track altitude error.

In the end, the Riccati equation and control law developed in Chapter 2 can be oriented and solved such that the optimum control input is developed for each step in a defined control sampling period.

4.1 Inherent Assumptions of Modified Projectile Linear Theory

To control indirect fire shots, the standard set of assumptions involved in projectile linear theory is modified. Most importantly, the assumption of small Euler pitch angle must be relaxed. To develop a better set of equations, Hainz and Costello have proposed an alternative set of assumptions [4]. The 6DOF equations used are in the no-roll frame as opposed to the roll frame 6DOF equations developed in Chapter 3. Using the information in section 3.2, it is possible to orient the 6DOF equations presented in (3.14) – (3.17) such that they are in the no-roll frame. Doing so, the equations become

$$\begin{Bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{Bmatrix} = \begin{bmatrix} c_\theta c_\psi & -s_\psi & s_\theta c_\psi \\ c_\theta s_\psi & c_\psi & s_\theta s_\psi \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{Bmatrix}\tilde{u}\\ \tilde{v}\\ \tilde{w}\end{Bmatrix} \tag{4.1}$$

$$\begin{Bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{Bmatrix} = \begin{bmatrix} 1 & 0 & t_\theta \\ 0 & 1 & 0 \\ 0 & 0 & 1/c_\theta \end{bmatrix} \begin{Bmatrix}\tilde{p}\\ \tilde{q}\\ \tilde{r}\end{Bmatrix} \tag{4.2}$$

$$\begin{Bmatrix}\dot{\tilde{u}}\\ \dot{\tilde{v}}\\ \dot{\tilde{w}}\end{Bmatrix} = \frac{1}{m}\begin{Bmatrix}\tilde{X}\\ \tilde{Y}\\ \tilde{Z}\end{Bmatrix} + \begin{Bmatrix} \tilde{r}\tilde{v} - \tilde{q}\tilde{w} \\ -t_\theta \tilde{r}\tilde{w} - \tilde{r}\tilde{u} \\ t_\theta \tilde{r}\tilde{v} + \tilde{q}\tilde{u} \end{Bmatrix} \tag{4.3}$$

$$\begin{Bmatrix}\dot{\tilde{p}}\\ \dot{\tilde{q}}\\ \dot{\tilde{r}}\end{Bmatrix} = [I]^{-1}\left[\begin{Bmatrix}\tilde{L}\\ \tilde{M}\\ \tilde{N}\end{Bmatrix} - \begin{bmatrix} 0 & -\tilde{r} & \tilde{q} \\ \tilde{r} & 0 & t_\theta \tilde{r} \\ -\tilde{q} & -t_\theta \tilde{r} & 0 \end{bmatrix} [I] \begin{Bmatrix}\tilde{p}\\ \tilde{q}\\ \tilde{r}\end{Bmatrix}\right] \tag{4.4}$$

Terms with a tilde represent a variable formulated in the no-roll frame. In subsequent Chapter 4 equations, the tilde will be dropped. Due to the amount of equations needed to run a 6DOF simulation, the equations needed for the force and moment terms are not presented here.

Assumptions are made to make the no-roll frame 6DOF equations quasi-linear and thus much less computationally expensive. As a basic summary, the assumptions involved in MPLT include, but are not limited to [4]:

1.) The states $\tilde{u}$, $\tilde{p}$, and ϕ are very large in relation to states $\tilde{v}$, $\tilde{w}$, ψ, $\tilde{q}$, and $\tilde{r}$. Products of small values and derivatives of small values are treated as negligible.
2.) The yaw angle is assumed to be small. This permits the following simplifications: $\sin(\psi) \approx \psi$ and $\cos(\psi) \approx 1$.
3.) The aerodynamic angles of attack have small magnitude.
4.) The projectile is symmetric about the station line, allowing the inertia matrix to become a diagonal matrix with $I_{YY} = I_{ZZ}$.
5.) The projectile is aerodynamically symmetric, allowing simplification to the amount of aerodynamic drag coefficients necessary to model the dynamics.
6.) The distances from the center of mass to the aerodynamic center of pressure are essentially zero in the y-direction and z-direction.
7.) Because $\tilde{u}$ is large in comparison to $\tilde{v}$ and $\tilde{w}$, the total speed is $V \approx \tilde{u}$ and $\dot{V} \approx \dot{\tilde{u}}$.
8.) A change of variables is used to convert the independent variable from time to dimensionless arc-length s. The arc-length is used to represent downrange travel in calibers. This is done through the relationship

$$s = \frac{1}{D}\int_0^t V\, d\tau \tag{4.5}$$

9.) The relationship between time derivatives and arc-length derivatives is then given by

$$\dot{\sigma} = \frac{V}{D}\sigma' \tag{4.6}$$

where a prime term represents an arc-length derivative and a superposed dot represents a time derivative.

Using the above assumptions, the nonlinear 6DOF set of equations in (4.1-4.4) can be reduced to a much simpler set of quasi-linear equations. The quasi-linear equations, presented in the no-roll frame with arc length *s* as the independent variable, are seen as follows:

$$x' = c_\theta D \tag{4.7}$$

$$y' = c_\theta D\psi + \frac{D}{V}v \tag{4.8}$$

$$z' = -Ds_\theta + \frac{Dc_\theta}{V}w \tag{4.9}$$

$$\psi' = \frac{D}{Vc_\theta}r \tag{4.10}$$

$$\theta' = \frac{D}{V}q \tag{4.11}$$

$$\phi' = \frac{D}{V}p \tag{4.12}$$

$$V' = -\frac{\pi\rho D^3}{8m}C_{X0}V - \frac{Dg}{V}s_\theta \tag{4.13}$$

$$v' = -\frac{\pi\rho D^3}{8m}C_{NA}v - Dr \tag{4.14}$$

$$w' = -\frac{\pi\rho D^3}{8m}C_{NA}w + Dq + \frac{Dg}{V}c_\theta \tag{4.15}$$

$$p' = \frac{\pi\rho V D^4}{8I_{xx}}C_{DD} + \frac{\pi\rho D^5}{16I_{xx}}C_{LP}p \tag{4.16}$$

$$q' = \frac{\pi\rho D^3 R_{MCP}}{8I_{YY}}C_{NA}w + \frac{\pi\rho D^5}{16I_{YY}}C_{MQ}q - \frac{I_{xx}D}{I_{YY}V}pr \tag{4.17}$$

$$r' = -\frac{\pi\rho D^3 R_{MCP}}{8I_{YY}}C_{NA}v + \frac{\pi\rho D^5}{16I_{YY}}C_{MQ}r + \frac{I_{xx}D}{I_{YY}V}pq \tag{4.18}$$

The quasi-linear set of equations will be arranged in a state-space form, as in (2.1), and manipulated such that nonlinearities are removed from the state-space matrices. Then, it is possible to implement optimal control techniques to predict and guide the projectile's flight.

4.2 Preliminary Manipulation to MPLT for Implementing Feedback Control

Upon examination of the set of equations given by (4.7) – (4.18), it can be seen that nonlinearities are caused primarily by three of the variables' presence: pitch angle *θ,* spin rate *p*, and total velocity *V*. The pitch angle causes nonlinearities in several terms due to the presence of the trigonometric functions. The total velocity causes nonlinearities in several of the equations due to its necessity in the independent variable conversion formula. Furthermore, nonlinearities are present due to spin rate p in equations (4.17) – (4.18). These terms will be omitted from the state vector in the state-space model in order to allow for the matrix-vector system formation.

Additionally, the roll angle ϕ has a minimal influence on the set of equations due to the no-roll frame used in the derivation. The roll angle only shows up in its own dynamic equation and will also be omitted from the state-space representation.

In Chapter 3.4, it was shown how the control input δ_c will act to orient the canards and cause a force and moment pair to alter projectile flight. This derivation covered nonlinear actuator modeling. However, when working with the linear controller, the process is much simpler. The variables C_{Y0} and C_{Z0} are aerodynamic trim force coefficients which are orthogonal to the projectile's station line axis and are created due to the movement of the controllable canards. Therefore, in the linear mapping, they will be treated directly as control inputs. Once the controller outputs their optimum values, they will then be transformed by a roll rotation and an inverse table lookup into the dimensional roll-frame δ_c values needed for the nonlinear actuator modeling.

The aerodynamic trim forces will serve to provide swerve forces as well as yaw and pitch moments to the missile. They will affect the dynamic modeling of the v, w, q, r equations. Accounting for the force and moment contributions due to the trim forces, equations (4.14) – (4.15) and (4.17) – (4.18) are re-written as

$$v' = -\frac{\pi\rho D^3}{8m} C_{NA} v - Dr + \frac{\rho S_C D}{2m} V C_{Y0} \tag{4.19}$$

$$w' = -\frac{\pi\rho D^3}{8m} C_{NA} w + Dq + \frac{Dg}{V} c_\theta - \frac{\rho S_C D}{2m} V C_{Z0} \tag{4.20}$$

$$q' = \frac{\pi\rho D^3 R_{MCP}}{8I_{YY}} C_{NA} w + \frac{\pi\rho D^5}{16I_{YY}} C_{MQ} q - \frac{I_{xx}D}{I_{YY}V} pr + \frac{\rho S_C D}{2I_{YY}} V(SL_C - SL_{CG})C_{Z0} \quad (4.21)$$

$$r' = -\frac{\pi\rho D^3 R_{MCP}}{8I_{YY}} C_{NA} v + \frac{\pi\rho D^5}{16I_{YY}} C_{MQ} r + \frac{I_{xx}D}{I_{YY}V} pq + \frac{\rho S_C D}{2I_{YY}} V(SL_C - SL_{CG})C_{Y0} \quad (4.22)$$

With the manipulations in (4.19) – (4.22), the system is now a function of control input $\boldsymbol{u}$. However, a few modifications will need to be made in order to enable best controller performance. Firstly, a method of incorporating the variation of *p*, *V*, and θ is sought. Although they will be omitted from the state-space model, they do vary with time and should be treated as such. A way to incorporate these variations is crucial.

4.3 Estimation of Time-Varying Parameters Using a Point Mass Vacuum Approach

To incorporate the time-varying nature of the parameters *p*, *V*, and θ into the model, their approximate future values must be predicted from current position to target each time through the controller. This can be done in part by considering the projectile as being a point mass in a vacuum. Assuming that the only force acting on the projectile is gravity, basic conservation principles can then help predict approximate future values for each of the parameters.

Assuming the projectile to be operating as a point mass in a vacuum, a basic model of the mass and its flight trajectory can be drawn as in Figure 4.1. The model depicts a projectile launching from a pre-defined origin and intersecting a desired target. It is worth noting that in this section, the *z*-axis is defined as positive upward, indicating that vacuum model *z* is actually the negative of projectile state *z*.

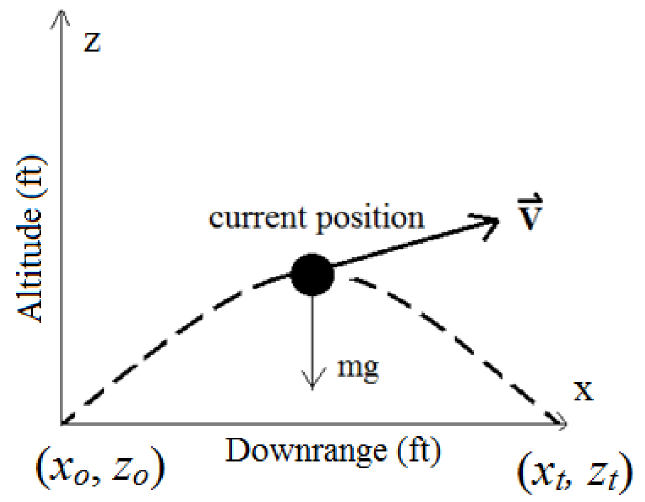


Figure 4.1: Projectile as a point mass in a vacuum

Starting with conservation of linear momentum (CoLM) in the x-direction, it can be shown that the velocity of the particle is constant in the x-direction and will be called V_{ox}.

$$m\frac{dV_x}{dt} = \sum F_x \Rightarrow \frac{dV_x}{dt} = 0 \Rightarrow V_x = \text{constant} = V_{ox} \tag{4.23}$$

Integrating the result of (4.23) yields time of flight as a function of downrange distance *x*.

$$x = V_{ox}t \Rightarrow t = \frac{x}{V_{ox}} \tag{4.24}$$

In a similar process, CoLM in the z-direction results in an expression of the vertical velocity as a function of time.

$$m\frac{dV_z}{dt} = \sum F_z \Rightarrow m\frac{dV_z}{dt} = -mg \Rightarrow \frac{dV_z}{dt} = -g \tag{4.25}$$

Defining $a = -g$ and integrating twice yields the z-position of the particle as a function of time.

$$z = z_o + V_{oz}t + \frac{1}{2}at^2 \tag{4.26}$$

Substituting (4.24) into the result, the z-position of the particle can be written as a function of downrange distance.

$$z(x) = z_o + V_{oz}\left(\frac{x}{V_{ox}}\right) + \frac{1}{2}a\left(\frac{x}{V_{ox}}\right)^2 \tag{4.27}$$

As previously stated, the vacuum model will be used to approximate the future p, V, and θ values of the projectile from current position to target position. The idea is to give the controller a theoretical way of getting from its current location to the target location via a point mass trajectory and to attempt to back out the parameter values needed to cause the projectile to closely follow this trajectory.

The projectile is launched from the (x_o, z_o) state and will theoretically intersect the target at (x_t, z_t). Because it is desired to execute the launch from a (0,0) origin, the terms x_o and z_o vanish from the equations. Thus, another point is needed to create the parabolic vacuum model. In order to begin the control sequence as early as possible, it is best to use a set of projectile downrange and altitude states very soon after launch to build the model. In order to begin control swiftly, the other point will consist of the x and $-z$ states of the projectile at the first instance in the control sampling period. A schematic of the situation can be viewed in Figure 4.2.

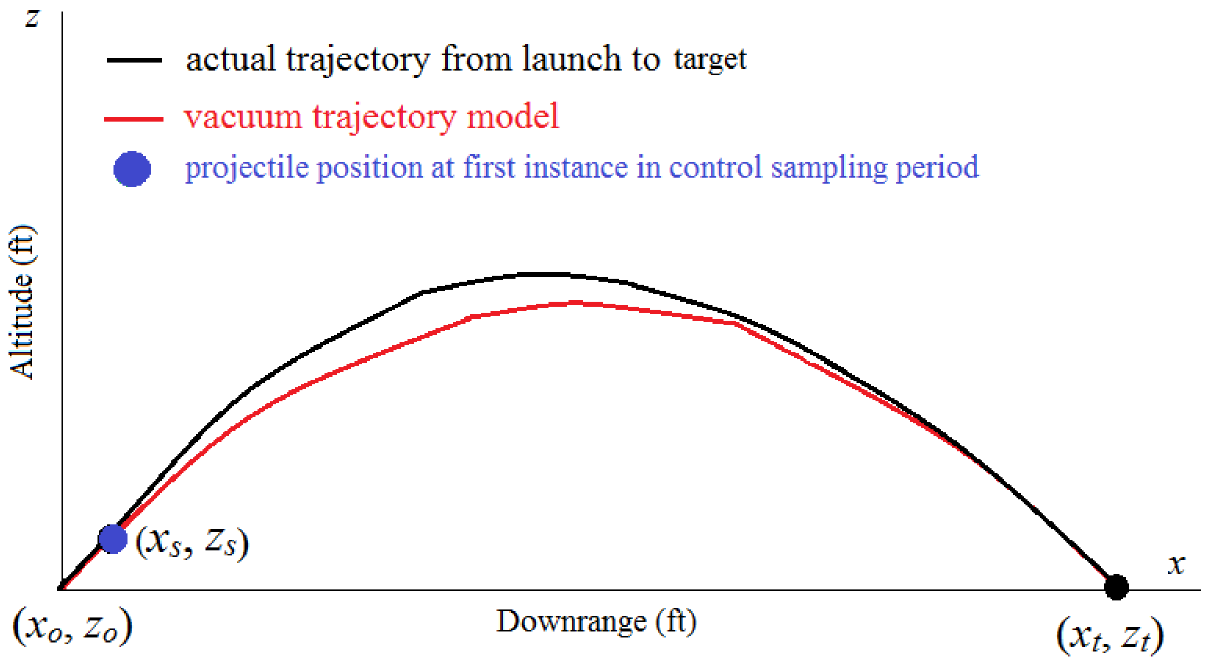

z
actual trajectory from launch to target
vacuum trajectory model
projectile position at first instance in control sampling period
Altitude (ft)
(x_s, z_s)
x
(x_o, z_o)
Downrange (ft)
(x_t, z_t)

Figure 4.2: Modeling future states with the vacuum trajectory

There are two free variables, or coefficients, which can be solved for: the initial x-velocity and initial z-velocity. Two desired ordered pairs exist for the model to intersect: (x_s, z_s) and (x_t, z_t). It is possible to write a pair of equations and solve for the coefficients that would cause the point mass trajectory to be launched from the origin and intersect both the early projectile position and the target position.

Equation (4.27) can be re-written as

$$z(x) = M_1 x + M_2 x^2 \tag{4.28}$$

where

$$M_1 = \frac{V_{oz}}{V_{ox}} \quad ; \quad M_2 = \frac{a}{2{V_{ox}}^2} \tag{4.29}$$

Plugging in the desired coordinate pairs, the equations are written in matrix-vector form:

$$\begin{bmatrix} x_s & {x_s}^2 \\ x_t & {x_t}^2 \end{bmatrix} \begin{Bmatrix} M_1 \\ M_2 \end{Bmatrix} = \begin{Bmatrix} -z_s \\ 0 \end{Bmatrix} \tag{4.30}$$

Using the equations for M_1 and $M_{2,}$ the vacuum model can then be created. It is worth noting that when the x-position of the projectile is zero, this process is invalid because it creates a matrix singularity in (4.30). Thus, the projectile is required to fly uncontrolled for the first caliber downrange regardless of the control sampling period.

Each time into the controller, the same original values of M_1 and M_2 are used to predict a path from the projectile's current downrange position to its target position. In other words, the same vacuum trajectory model is used throughout the entire duration of flight.

With the vacuum model created, basic physics can be used to make the predictions about the sine and cosine of the pitch angle at future optimal states. Inside the controller, the x-distance from current projectile position to target is broken into a pre-determined number of equal length segments N_s. For each segment, the corresponding values of sine and cosine of the pitch angle can be predicted via the vacuum model. A basic schematic of the segmentation is seen in Figure 4.3.

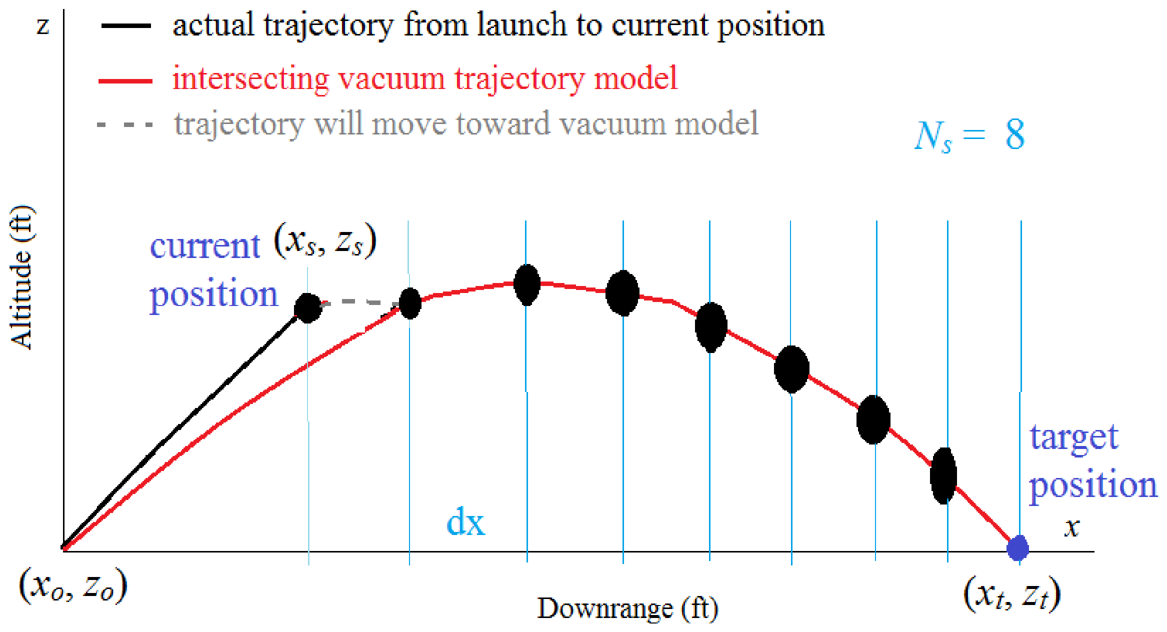

z
actual trajectory from launch to current position
intersecting vacuum trajectory model
trajectory will move toward vacuum model
$N_s = 8$
Altitude (ft)
current position (x_s, z_s)
target position
x
dx
(x_o, z_o)
Downrange (ft)
(x_t, z_t)

Figure 4.3: Segmentation of predicted trajectory

It is worth noting that, at the current state at any time *t* through the controller, the actual position values and vacuum prediction values are not identical. The hope, however, is that the vacuum predictions for altitude and pitch angle will provide a successful possible path to the target and will then cause the trajectory to mimic, or follow, that path. As the projectile gets further downrange, its actual path should closely match the path of the vacuum model.

In order to compute the angle values for each segment, a closer look is taken into a single segment, as seen in Figure 4.4.

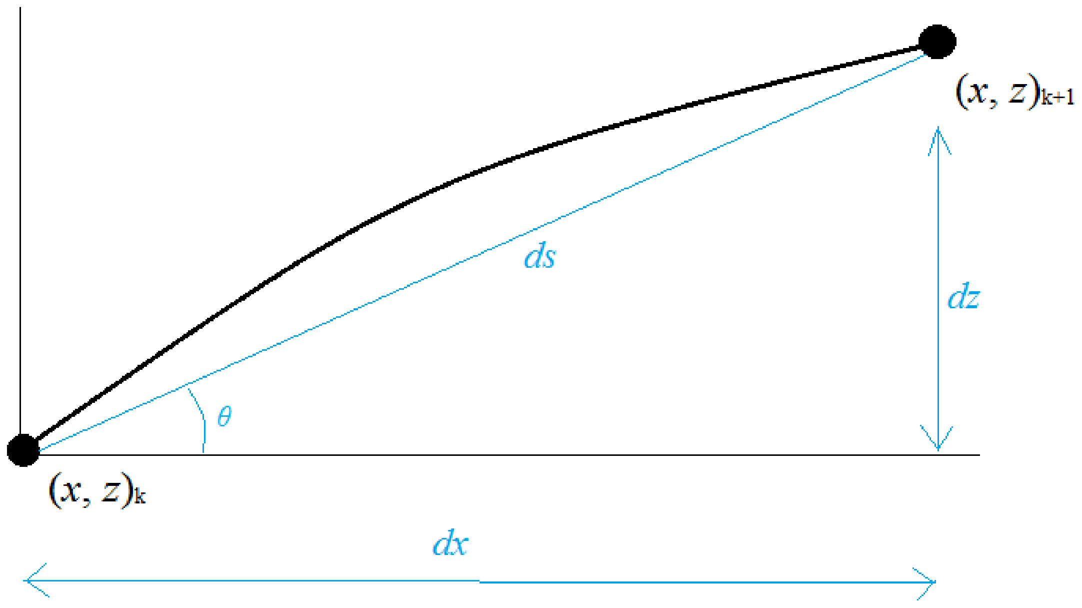


Figure 4.4: Prediction sine and cosine values

The value of *dx* is easily computed by the formula

$$dx = \frac{x_t - x_1}{N_s} \tag{4.31}$$

where x_1 is the projectile's current downrange position upon the call for control. Because the future altitude *z* has been developed as a function of downrange distance *x*, it is possible to compute the theoretical next altitude value as

$$z_{k+1} = M_1(x_{k+1}) + M_2(x_{k+1})^2 \tag{4.32}$$

from which *dz* is computed as

$$dz_k = z_{k+1} - z_k \tag{4.33}$$

Using the Pythagorean Theorem, the differential arc length from the current state to the next predicted state is

$$ds_k = \sqrt{dx^2 + {dz_k}^2} \tag{4.34}$$

It is then possible to predict the sine and cosine values based on the right triangle which has been formed:

$$c_{\theta_k} = \frac{dx}{ds_k} \tag{4.35}$$

$$s_{\theta_k} = \frac{dz_k}{ds_k} \tag{4.36}$$

Using the differential arc length calculated in (4.34), values for roll rate *p* and total speed *V* can also be predicted for each segment from current position to target. To do this, closed form solutions for these equations are used [4]. First, the value of *ds* is currently in downrange feet, and the closed-form expressions for *p* and *V* deal in calibers. In order to convert to calibers, the following conversion is performed.

$$h_k = \frac{ds_k}{D} \tag{4.37}$$

Values of *p* and *V* are then recursively predicted using analytical closed-form expressions as given in (4.38 – (4.39)

$$V(s+h) = \sqrt{\left(V^2(s) + \frac{b_v}{a_v}\right) e^{-2a_v h} - \frac{b_v}{a_v}} \tag{4.38}$$

$$p(s+h) = C_{pe1}e^{C_{pe2}h} - C_{p0} \tag{4.39}$$

where the values of the various constants are defined by the following:

$$b_v = gD\sin\theta(s) \tag{4.40}$$

$$a_v = \frac{\pi\rho D^3}{8m}C_{X0} \tag{4.41}$$

$$C_{pe2} = \frac{\pi\rho D^5}{16I_{xx}}C_{LP} \tag{4.42}$$

$$C_{pe1} = p(s) + \frac{2C_{DD}V(s)}{DC_{LP}} \tag{4.43}$$

$$C_{p0} = \frac{2C_{DD}V(s)}{DC_{LP}} \tag{4.44}$$

The values for p, V, $\sin(\theta)$, and $\cos(\theta)$ are thus able to be recursively predicted from current downrange position to target each time the controller is called. Doing this will allow for the accounting of their time-varying nature into a controller and will also give the controller a set of these parameters which represent a successful path toward the target.

The predictions of the parameters *p* and *V* given by the point mass vacuum trajectory should be better the farther downrange the projectile has traveled. When the projectile is in the early stages of its flight, the point mass model will not necessarily match the actual model at the current state. However, as the projectile flies further downrange, it should begin to track the point mass model and the predictions given become extremely close to the actual values for *p* and V. Conversely, because the predictions for $\sin(\theta)$ and $\cos(\theta)$ are based solely on the physics of the vacuum model, they do not change much throughout flight.

4.4 General Linearization via Taylor Series Expansion to Better Control Altitude

Another initial issue with the controller at high launch angles concerns how to best control the altitude state. The current modified linear equation for modeling the changing altitude is given in (4.45) with an important term highlighted in red text:

$$z' = -Ds_{\theta} + \frac{Dc_{\theta}}{V}w \tag{4.45}$$

The issue is that this term is essentially flying uncontrolled in the current form of the equation. The other term in the equation is influenced by w, which is being directly controlled by trim force C_{Z0}. Alteration to (4.45) is sought such that another direct path between the result and the control input is created, which would serve to improve altitude control. To accomplish this task, a general linearization scheme is employed using a Taylor series expansion about a trim point defined in part by the vacuum trajectory model.

To begin the process, the altitude state derivative equation is written as a basic function f of the states θ and w:

$$z' = f(\theta, w) \tag{4.46}$$

Assuming a trim point defined by $(\bar{\theta}, \bar{w})$ it is possible to approximate (4.45) as a first-order Taylor series expansion about the trim point. The resulting equation is then

$$z' = f(\bar{\theta}, \overline{w}) + \frac{\partial f}{\partial \theta}\delta_\theta + \frac{\partial f}{\partial w}\delta_w \tag{4.47}$$

Evaluating the partial derivatives in (4.47) allows for an expansion. The general linearization equation then becomes

$$z' = -Ds_{\overline{\theta}} + \frac{Dc_{\overline{\theta}}}{V}\overline{w} - \left(\frac{Ds_{\overline{\theta}}}{V}\overline{w} + Dc_{\overline{\theta}}\right)\delta_\theta + \frac{Dc_{\overline{\theta}}}{V}\delta_w \tag{4.48}$$

It is also known that $\delta_w = w - \overline{w}$, and so

$$z' = -Ds_{\overline{\theta}} + \frac{Dc_{\overline{\theta}}}{V}\overline{w} - \left(\frac{Ds_{\overline{\theta}}}{V}\overline{w} + Dc_{\overline{\theta}}\right)\delta_\theta + \frac{Dc_{\overline{\theta}}}{V}(w - \overline{w}) \tag{4.49}$$

Assuming that the trim point value $\overline{w}$ is equal to zero, the expansion is greatly simplified. The resulting equation is given by

$$z' = -Ds_{\overline{\theta}} + \frac{Dc_{\overline{\theta}}}{V}w - Dc_{\overline{\theta}}\delta_\theta \tag{4.50}$$

The pitch angle can be represented by

$$\theta = \bar{\theta} + \delta_\theta \tag{4.51}$$

Taking the derivative of each side of (4.51),

$$\dot{\theta} = \dot{\bar{\theta}} + \dot{\delta}_\theta \tag{4.52}$$

Assuming that the derivative of the trim point is essentially zero,

$$\dot{\theta} = \dot{\delta}_\theta \tag{4.53}$$

By developing an expression for the derivative of δ_θ, it is possible to add a Θ-perturbation state into the model. Rearranging (4.51), this state is defined as

$$\delta_\theta = \theta - \bar{\theta} \tag{4.54}$$

where θ is the current pitch angle state from the plant dynamics and $\bar{\theta}$ is the corresponding initial pitch angle predicted by the point mass vacuum trajectory model for the first segment which will merge the actual trajectory with the model. Because of the relationship given by (4.53), the derivative of the new state is defined as

$$\dot{\delta}_\theta = \frac{D}{V} q \tag{4.55}$$

Since the δ_θ term can now become a state in the model, there is essentially a new path between a term in the altitude state equation and the trim force C_{Z0}. The process can be illustrated by viewing and examining the red term in (4.56):

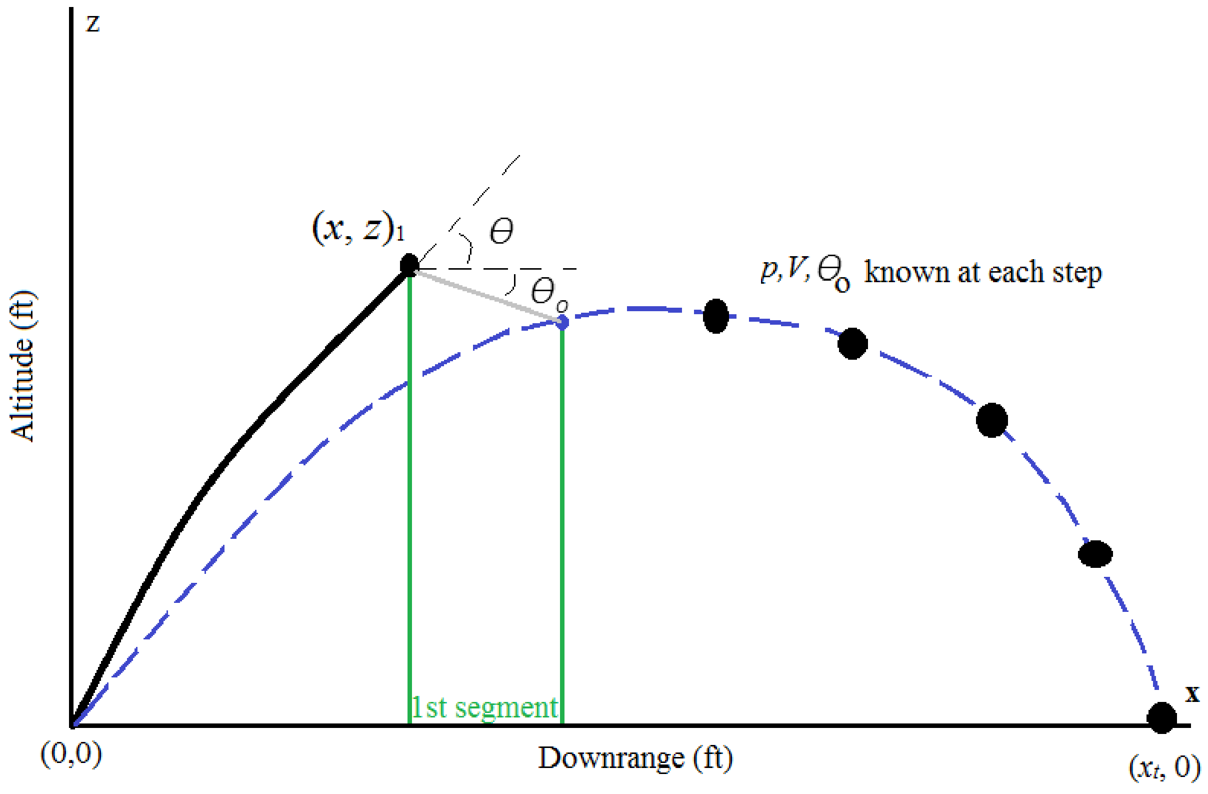
$$z' = -D s_{\bar{\theta}} + \frac{D c_{\bar{\theta}}}{V} w - D c_{\bar{\theta}} \delta_\theta \tag{4.56}$$

Now, the state δ_θ has been introduced into the model. It is known that $\delta_\theta = f(q)$. When looking back at the equation for q presented in (4.21), it is seen that q is a direct function of the control trim force. Thus, it can be said that

$$\delta_\theta = f\big(q(C_{Z0})\big) \;\; \Rightarrow \;\; \delta_\theta = f(C_{Z0}) \tag{4.57}$$

Therefore, another path between the altitude state derivative and the control input has been created. This will help to better predict the optimal control altitude for the high launch shots. Also with the addition of the δ_θ state, a state for tracking the difference between the vacuum model altitude and the actual altitude state has effectively been created. By placing a high control penalty on this state, it is possible to force the projectile to track the vacuum model, which is known to intersect the target at zero altitude.

Figure 4.5 illustrates the intent of using both the vacuum model and the Θ-perturbation state to help control altitude.



$\boldsymbol{\theta_o}$ **≡ angle needed in the 1st segment such that trajectory merges with vacuum model**

$\boldsymbol{\theta}$ **≡ current projectile Euler pitch angle**

$\boldsymbol{\delta_\theta \equiv \theta - \theta_o}$

Figure 4.5: Illustration for altitude control

Ideally, the term δ_θ will converge to a zero value, meaning that the trajectory is in fact tracking that of the vacuum model.

Incorporating both the ability to predict time-varying parameters using a vacuum trajectory and the use of a Taylor series expansion for altitude control enables better controller performance. Now, the system can be represented in state-space form and control can be imparted on the projectile.

4.5 State-Space Representation of System for Feedback Control

The projectile yaw-swerve and epicyclic pitch-yaw equations can now be collected into a nine-dimensional state-space description. In order to conform to the antecedents of optimal control, an uncontrollable state $\dot{w}$ is appended to the model with the initial condition $\dot{w}(0) = 1$. This term is used in the state-space model to treat gravity as an uncontrollable mode. In state-space form, the system has independent variable s and is represented by (4.58).

$$\begin{Bmatrix} \dot{\boldsymbol{\xi}} \\ \dot{\boldsymbol{\eta}} \\ \ddot{w} \end{Bmatrix} = \begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{\Gamma} & \boldsymbol{\Sigma} \\ \mathbf{0} & \boldsymbol{\Xi} & \boldsymbol{\Lambda} \\ \mathbf{0} & \mathbf{0} & 0 \end{bmatrix} \begin{Bmatrix} \boldsymbol{\xi} \\ \boldsymbol{\eta} \\ \dot{w} \end{Bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \\ \mathbf{0} \end{bmatrix} \begin{Bmatrix} C_{Z0} \\ C_{Y0} \end{Bmatrix} \tag{4.58}$$

which can be written more compactly as

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{4.59}$$

The terms in the state-matrix $\boldsymbol{A}$ are described by the following:

$$\mathbf{\Phi} = DC_{\bar{\theta}}\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.60}$$

$$\mathbf{\Gamma} = \frac{D}{V}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\bar{\theta}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \dfrac{1}{c_{\bar{\theta}}} \end{bmatrix} \tag{4.61}$$

$$\mathbf{\Sigma} = Ds_{\bar{\theta}}[0 \quad -1 \quad 0 \quad 0]^T \tag{4.62}$$

$$\mathbf{\Lambda} = \frac{Dg}{V}c_{\bar{\theta}}[0 \quad 1 \quad 0 \quad 0]^T \tag{4.63}$$

$$\mathbf{\Xi} = \begin{bmatrix} -\Xi_1 & 0 & 0 & -D \\ 0 & -\Xi_1 & D & 0 \\ 0 & \Xi_3 & \Xi_4 & -\Xi_5 \\ -\Xi_3 & 0 & \Xi_5 & \Xi_4 \end{bmatrix} \tag{4.64}$$

The terms in $\mathbf{\Xi}$ represent the aerodynamic force terms in the equations and are given by

$$\Xi_1 = \frac{\rho S D}{2m}C_{NA} \tag{4.65}$$

$$\Xi_3 = \frac{\rho S D}{2I_{yy}}C_{MA} \tag{4.66}$$

$$\Xi_4 = \frac{\rho S D^3}{4I_{yy}}C_{MQ} \tag{4.67}$$

$$\Xi_5 = \frac{I_{xx}D}{I_{YY}V}p \tag{4.68}$$

$$C_{MA} = (SL_{COP} - SL_{CG})C_{NA} \tag{4.69}$$

Additionally, the state vector ***x*** is represented by

$$\boldsymbol{\xi} = [y \quad z \quad \delta_\theta \quad \psi]^T \tag{4.70}$$

$$\boldsymbol{\eta} = [v \quad w \quad q \quad r]^T \tag{4.71}$$

The control matrix ***B*** contains the term **b**, which is defined as

$$\mathbf{b} = \begin{bmatrix} 0 & -b_1 & b_2 & 0 \\ b_1 & 0 & 0 & b_2 \end{bmatrix}^T \tag{4.72}$$

where

$$b_1 = \frac{\rho S_C D}{2m}V \tag{4.73}$$

$$b_2 = \frac{\rho S_C D}{2I_{yy}}V(SL_C - SL_{CG}) \tag{4.74}$$

Using techniques of optimal control as developed in Chapter 2, the optimum control input can then be found. Control will be imparted onto the system at a defined sampling period.

4.6 Use of Optimal Control to Obtain Control Input

The following is adapted from [12]. From Chapter 2, the control law for this system was found to be

$$\boldsymbol{u}(s) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{T}(s)\boldsymbol{N}(s)\boldsymbol{x}(s) \tag{4.75}$$

and it was discovered that $\boldsymbol{N}$ was the solution to the matrix Riccati differential equation (2.35). In part by treating gravity as an uncontrollable mode, the need for a reference trajectory has been eliminated. Thus, the matrix term $\boldsymbol{Q}$ in (2.35) is set to a zero matrix, giving

$$\dot{\boldsymbol{N}}(s) = -\boldsymbol{N}(s)\boldsymbol{A}(s) - \boldsymbol{A}^{T}(s)\boldsymbol{N}(s) + \boldsymbol{N}(s)\boldsymbol{B}(s)\boldsymbol{R}^{-1}\boldsymbol{B}^{\boldsymbol{T}}(s)\boldsymbol{N}(s) \tag{4.76}$$

Then, (4.76) can be decomposed into two matrix differential equations:

$$\dot{\boldsymbol{W}}(s) = \boldsymbol{A}(s)\boldsymbol{W}(s) - \boldsymbol{B}(s)\boldsymbol{R}^{-\boldsymbol{1}}\boldsymbol{B}^{\boldsymbol{T}}(s)\boldsymbol{Y}(s) \tag{4.77}$$

$$\dot{\boldsymbol{Y}}(s) = -\boldsymbol{A}^{\boldsymbol{T}}(s)\boldsymbol{Y}(s) \tag{4.78}$$

Adhering to the cost function in (2.20), target conditions are chosen as

$$\begin{gathered}\boldsymbol{W}(s_t) = \boldsymbol{I}\\ \boldsymbol{Y}(s_t) = \boldsymbol{P}\end{gathered} \tag{4.79}$$

The matrix term $\boldsymbol{P}$ is a diagonal matrix of control penalty terms. In this case, it is desired to control the altitude and crossrange states to zero values and thus high control penalties are placed on these terms. Additionally, a high penalty is placed on the Θ-perturbation state in hope that the trajectory is driven toward that of the vacuum model.

The matrix Riccati solution can be found to be

$$\boldsymbol{N}(s) = \boldsymbol{Y}(s)\boldsymbol{W}^{-1}(s) \tag{4.80}$$

Equations (4.77) – (4.78) can be written in terms of a time-varying Hamiltonian as $\dot{\boldsymbol{Z}}(s) = \boldsymbol{F}(s)\boldsymbol{Z}(s)$. This is represented by

$$\begin{Bmatrix} \dot{\boldsymbol{W}}(s) \\ \dot{\boldsymbol{Y}}(s) \end{Bmatrix} = \begin{bmatrix} \boldsymbol{A}(s) & -\boldsymbol{B}(s)\boldsymbol{R}^{-1}\boldsymbol{B}^{\boldsymbol{T}}(s) \\ \boldsymbol{0} & -\boldsymbol{A}^{\boldsymbol{T}}(s) \end{bmatrix} \begin{Bmatrix} \boldsymbol{W}(s) \\ \boldsymbol{Y}(s) \end{Bmatrix} \tag{4.81}$$

In order to solve the time-varying Riccati equation, the trajectory of the projectile is discretized into N_s segments from current position to target. At each segment, the time-varying parameters *p*, *V*, and θ are estimated using the vacuum trajectory model as previously discussed. The solution is then back-propagated using equations (4.82) – (4.83).

$$\boldsymbol{Z}_{N_s} = \left(\boldsymbol{I} + \frac{h}{2}\boldsymbol{F}_{N_s}\right)^{-1} \boldsymbol{Z}(s_t) \tag{4.82}$$

$$\boldsymbol{Z}_k = \left(\boldsymbol{I} + \frac{h}{2}\boldsymbol{F}_k\right)^{-1} \left(\boldsymbol{I} - \frac{h}{2}\boldsymbol{F}_{k+1}\right)\boldsymbol{Z}_{k+1} \tag{4.83}$$

where h is the differential arc-length of the respective segment given by the vacuum trajectory model which connects the points denoted by subscripts *k* and *k+1*.

The time-varying Hamiltonian ***F*** changes for each segment from current position to target. Once the trajectory is discretized into its segments, the parameters *p*, *V*, and θ are predicted for each segment. In doing this, ***F*** can then be recursively predicted for each segment.

The matrix term $\boldsymbol{Z}$ can then be back-propagated from the target to the current position. Using $\boldsymbol{Z}_1$, the matrix Riccati solution is then found via (4.80) and thus the control can be computed using (4.75). Essentially, a time-varying model is used in conjunction with the vacuum model to predict the projectile's optimal path from current position to target. Then, the proper control input for the current state of the projectile is backed out using the process described.

A final step in the process concerns a transformation of the resulting canard deflection. In the control equations, the $\boldsymbol{B}$ matrix includes scaling due to dynamic pressure, canard area, and stationline moment arm. The control input found contains the non-dimensional canard trim force coefficients in the no-roll frame. In order to move these values into dimensional canard deflections and into the roll frame, they are converted by a table lookup into a dimensional form and then rotated into the body frame by a roll angle transformation matrix.

The canard deflections are first divided by the canard lift coefficient $C_{L\alpha}$ such that the units are then in radians. For supersonic flight, the value of $C_{L\alpha}$ for this particular rocket is always 4.135 rad^{-1}. For subsonic flight, a Mach table lookup is used and the Mach number of the canard is assumed to be equal to that of the mass center for simplicity. The result is limited from [-1,1] rad such that a saturation limit is applied to the system and then rotated into the roll frame.

4.7 Summary of Control Implementation Algorithm

The process of implementing control has been developed throughout the chapter. The basic process of control can be summarized in an 8-step process. Control is determined through the following progression:

1.) At the first time in the control sampling period, solve for and save the coefficients for creating the vacuum model.
    a. The model will launch from the origin, intersect a projectile state early in the trajectory, and hit the target.
2.) Compute the pitch angle to get from current position to the first spot on the vacuum trajectory. From the prediction of this angle, develop the current value of the δ_θ state.
3.) Recursively predict values for p, V, c_θ, s_θ, and h using the vacuum trajectory model while updating aerodynamic coefficients at each segment based on new predicted velocity and altitude.
4.) Build the corresponding matrix Hamiltonian for each segment.
5.) Integrate backwards in time using (4.82) – (4.83).
6.) Using $\boldsymbol{Z_1}$, compute the Riccati solution at the current state from (4.80).
7.) Compute the control needed at the current state using (4.75).
8.) Convert to dimensional form, limit from [-1,1] rad and rotate into roll frame.

The process is repeated each time in the control sampling period. In the event of a matrix singularity causing controller breakdown at any point in the trajectory, the previous values of $\delta_{C,NR}$ are kept until the next time in the control sampling period.

4.8 <u>Overall Flowchart of Projectile Flight and Control</u>

In order to capture the overall picture of flight simulation and control, it is helpful to organize the process into a computational flowchart. The flowchart, seen in Figure 4.6, describes the process by which control is implemented and the resulting flight is simulated.

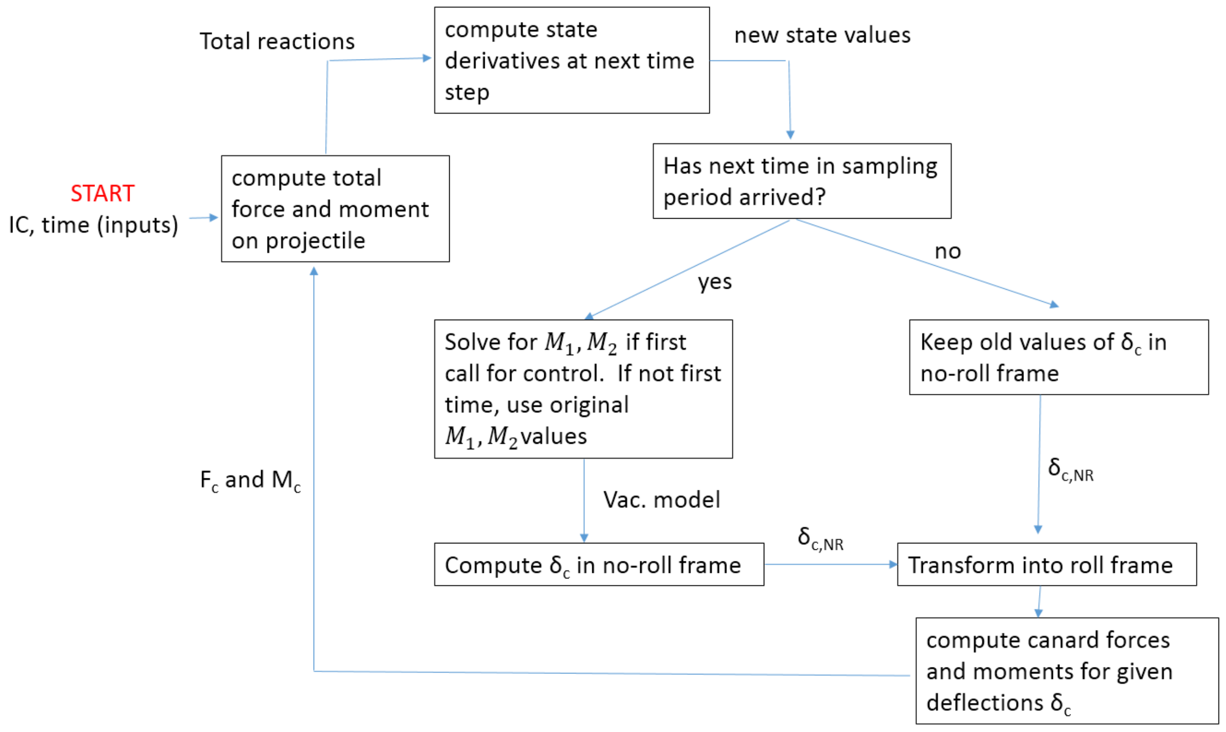

Total reactions
compute state derivatives at next time step
new state values
START
IC, time (inputs)
compute total force and moment on projectile
Has next time in sampling period arrived?
yes
no
Solve for M_1, M_2 if first call for control. If not first time, use original M_1, M_2 values
Keep old values of δ_c in no-roll frame
F_c and M_c
$\delta_{c,NR}$
Vac. model
$\delta_{c,NR}$
Compute δ_c in no-roll frame
Transform into roll frame
compute canard forces and moments for given deflections δ_c

Figure 4.6: Flowchart of simulation and control algorithm

The initial condition and time are passed into a function which computes the total forces and moments on the projectile, after which the values of the state derivatives are computed using a variable time step. If the next time value has reached a multiple in the control sampling period (e.g. 5 ms, 10 ms, 15 ms and so on for a 5 ms control sampling period), then a controller function is utilized to calculate the optimal canard deflections, which are then rotated into the roll frame and used to compute canard force and moment contributions. A new total force and moment is then calculated, which leads to the next values of the state derivatives. If the next instant in the sampling period has not been reached, the last calculated values of canard commands in the no-roll frame are used to calculate new reactions and state derivatives. The process is repeated until the projectile reaches the target.

5. Results and Discussion

In this chapter, the results of trajectory simulations will be presented and discussed. In achieving optimal results, several parameters within the controller had to be varied and tuned. Additionally, faults were found in the controller which prohibited optimum performance. Remedies to these faults were discovered and incorporated. Ultimately, peak performance was found and demonstrated with a Monte Carlo dispersion set of varying pitch and yaw angles. Dispersion sets were chosen such that uncontrolled shots impacted the ground on all four sides of the x,y target plane: top, bottom, left and right.

The projectile simulated in this thesis is from the Hydra-70 family of rockets [14]. The Hydra-70 system is a series of 2.75-inch rockets created by the U.S. Navy in the 1940s. The uses for this family of rockets include ground-to-ground firings and the rockets have been used by many branches of the U.S. military including the Marine Corps, Navy, Air Force, and Army Special Operations forces.

Several numerical properties of the projectile were first defined before the start of simulations could begin. The properties include various parameters such as projectile mass and inertia properties, positional coordinates of the projectile and canard mass centers, and acceleration due to gravity. The basic projectile properties needed to run simulations are seen in Table 5.1. Each of the four canards has a separate set of dynamical properties which can be seen in Appendix D.

Table 5.1: Projectile properties required for simulations

Property	**Symbol**	**Value**
Reference diameter	D	0.223 ft
Reference area	S	0.0391 ft^2
Mass	m	0.7143 slug
Gravitational acceleration	g	32.2 ft/s^2
x-axis inertia	I_{xx}	0.005 slug-ft^2
y-axis inertia	I_{yy}	1.4 slug-ft^2
z-axis inertia	I_{zz}	1.4 slug-ft^2
Stationline of c.g.	SL_{cg}	2.5 ft
Buttline of c.g.	BL_{cg}	0 ft
Waterline of c.g.	WL_{cg}	0 ft

The values in Table 5.1 were used throughout the duration of the simulations presented in this thesis. All shots were launched from the origin in inertial space. Initial pitch and yaw angles were variable. Other initial conditions were: $\phi = 0$ rad, $u = 2177.7\ \mathrm{ft/s}$, $v = 0\ \mathrm{ft/s}$, $w = 0\ \mathrm{ft/s}$, $p = -58.928\ \mathrm{rad/s}$, $q = 0\ \mathrm{rad/s}$, $r = -0.058\ \mathrm{rad/s}$.

There were several properties which needed to be optimized inside the controller. An optimal control sampling period must be determined. Obviously, tighter control sampling periods would serve for better performance. However, this comes at the expense of computing efficiency. It is necessary to balance improved control with computational expense. Likewise, the number of segments used in the implicit trajectory predictions provides a similar dilemma. If too few segments are used, the predictions given are poor, and the control breaks down.

5.1 Examining a Trajectory: Uncontrolled vs. Controlled States

When implementing control on the projectile, the trajectory is altered in a desired way. This causes some of the controlled states to become drastically altered when compared to their uncontrolled paths. In this thesis, control is performed such that swerve forces and yaw and pitch moments are altered. Thus, it would be expected that the states *v*, *w*, *q*, and *r* would have vastly different trends in controlled instances. Figures 5.1-5.11 show comparisons of the paths of the projectile states in controlled manners versus their uncontrolled manners. For the following figures, the projectile was shot with a 39.02° pitch angle and 0.23° yaw angle.

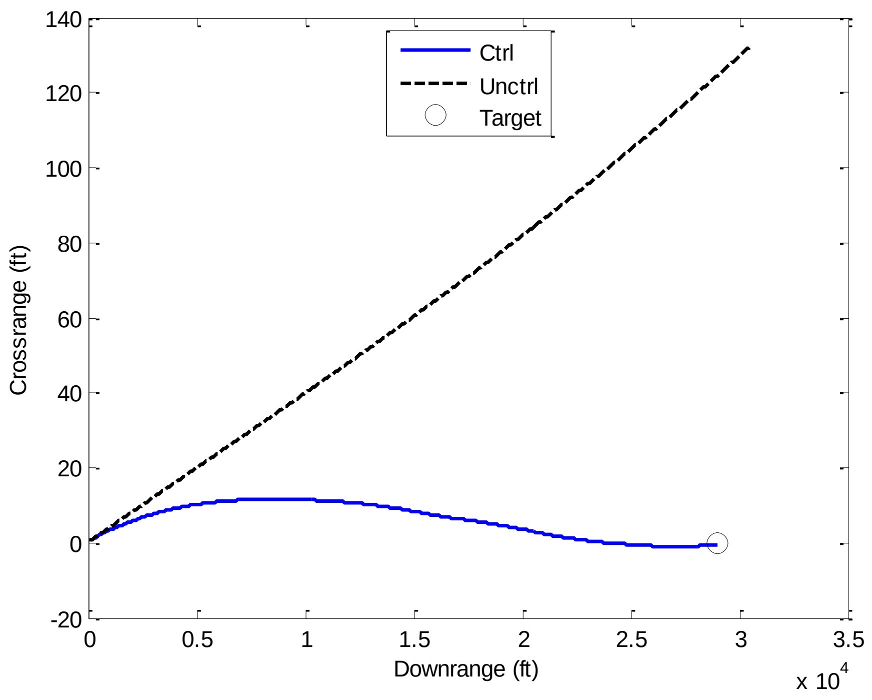


Figure 5.1: Crossrange comparison

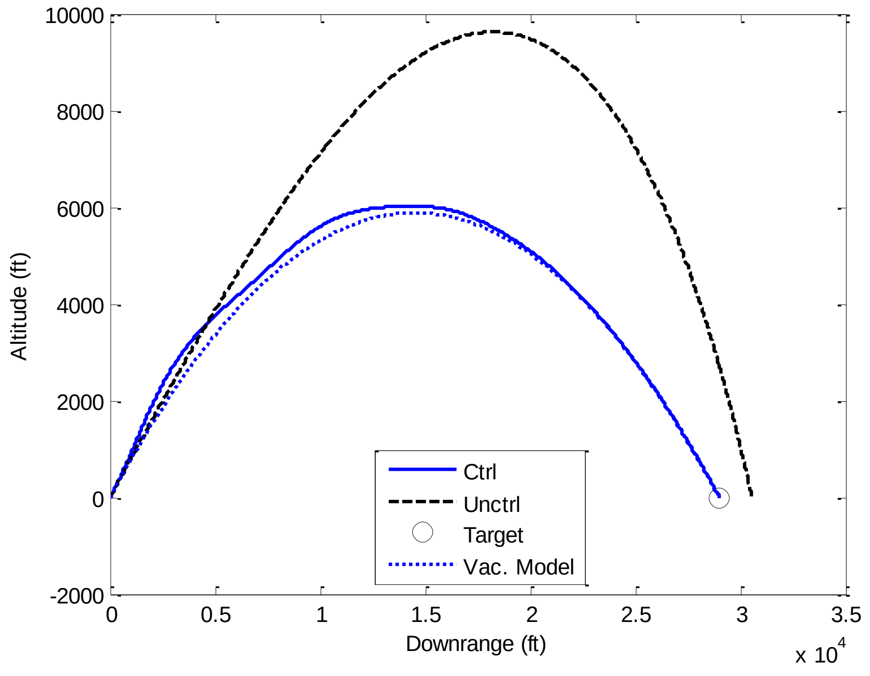


Figure 5.2: Altitude comparisons

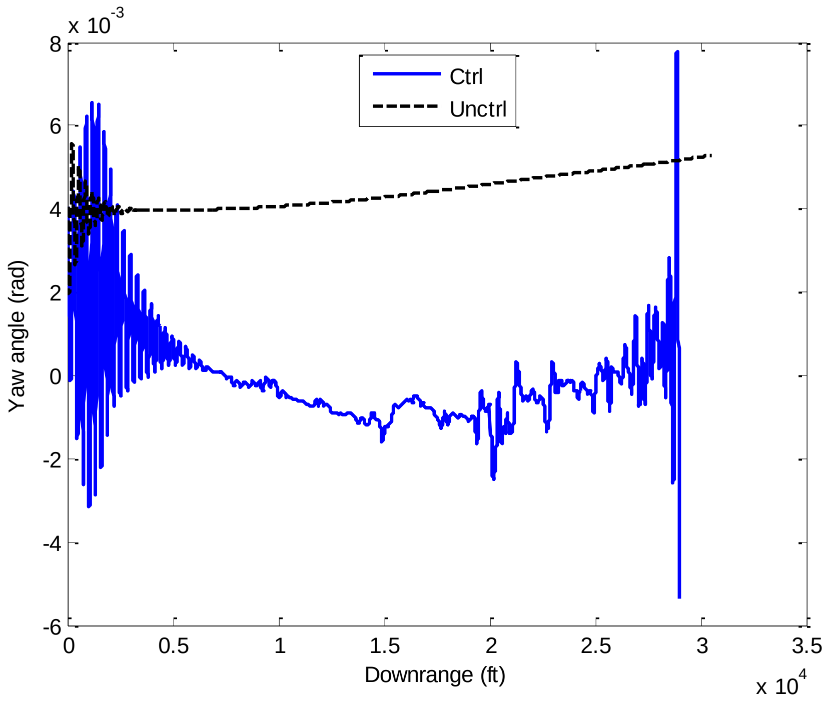


Figure 5.3: Yaw angle comparisons

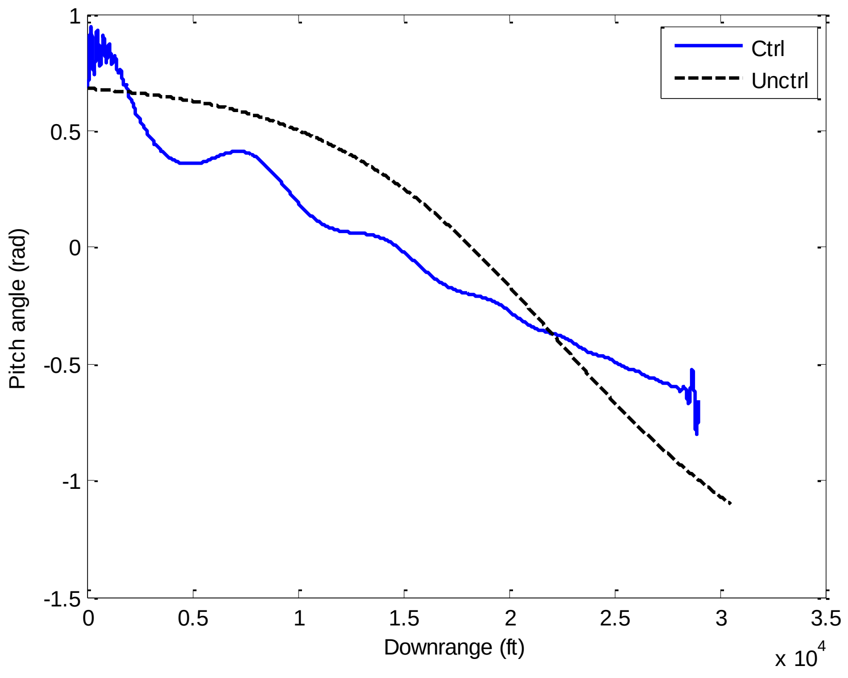


Figure 5.4: Pitch angle comparisons

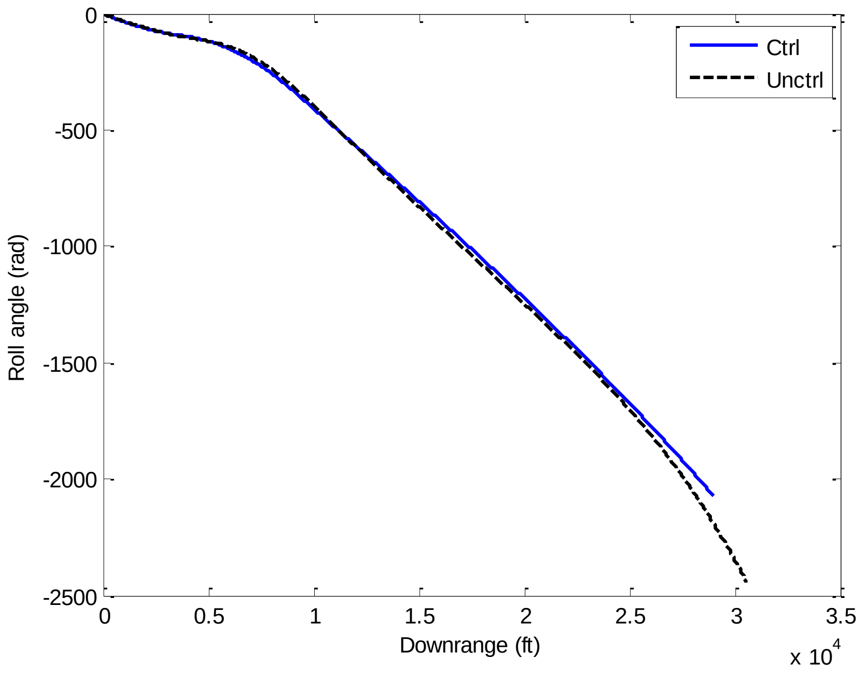


Figure 5.5: Roll angle comparisons

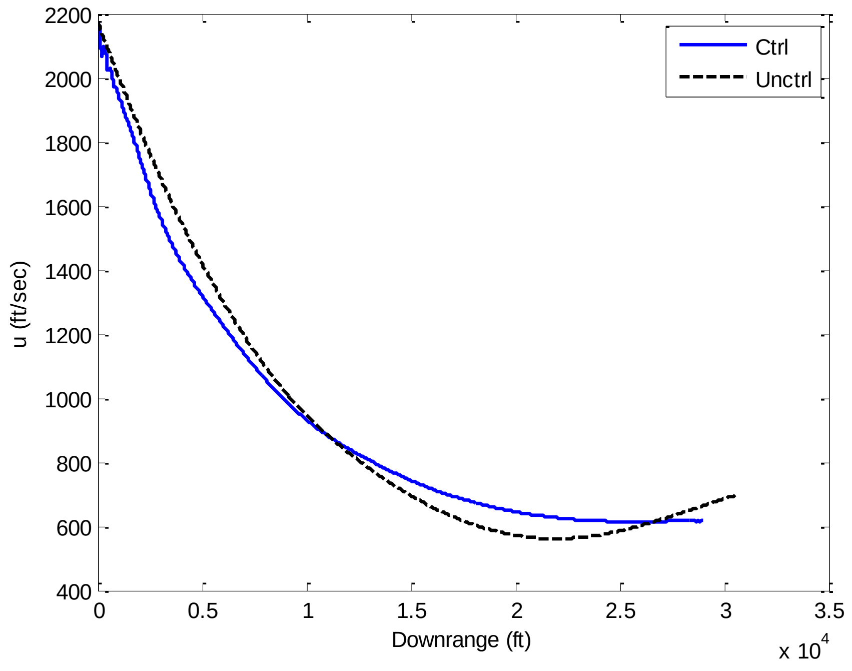


Figure 5.6: I_B velocity comparisons

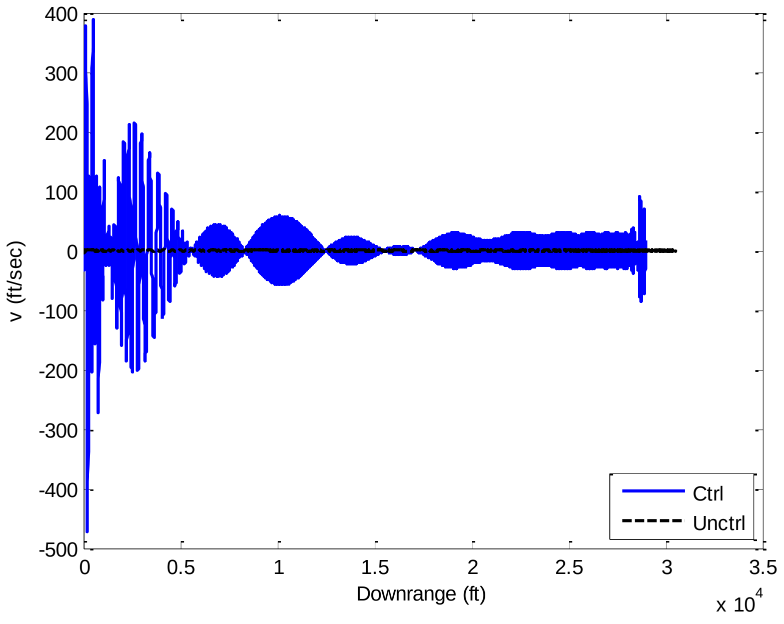


Figure 5.7: J_B velocity comparisons

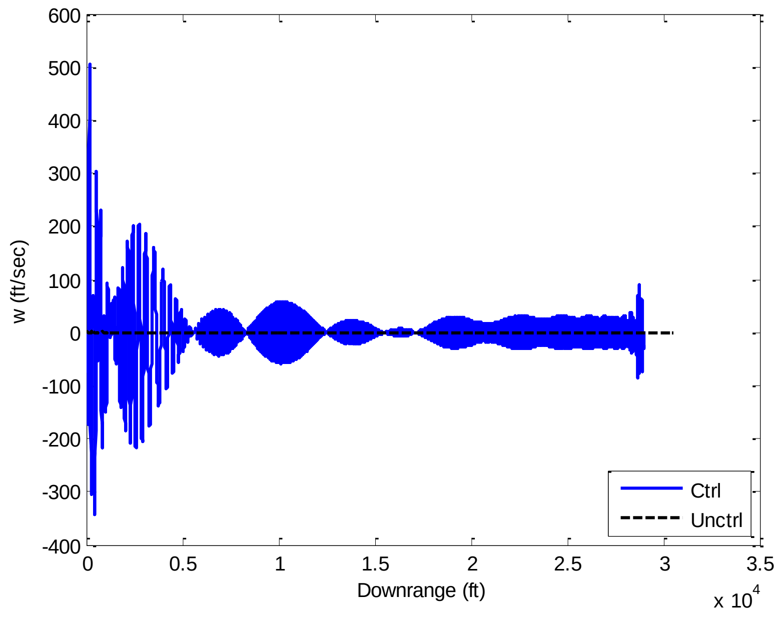


Figure 5.8: K_B velocity comparisons

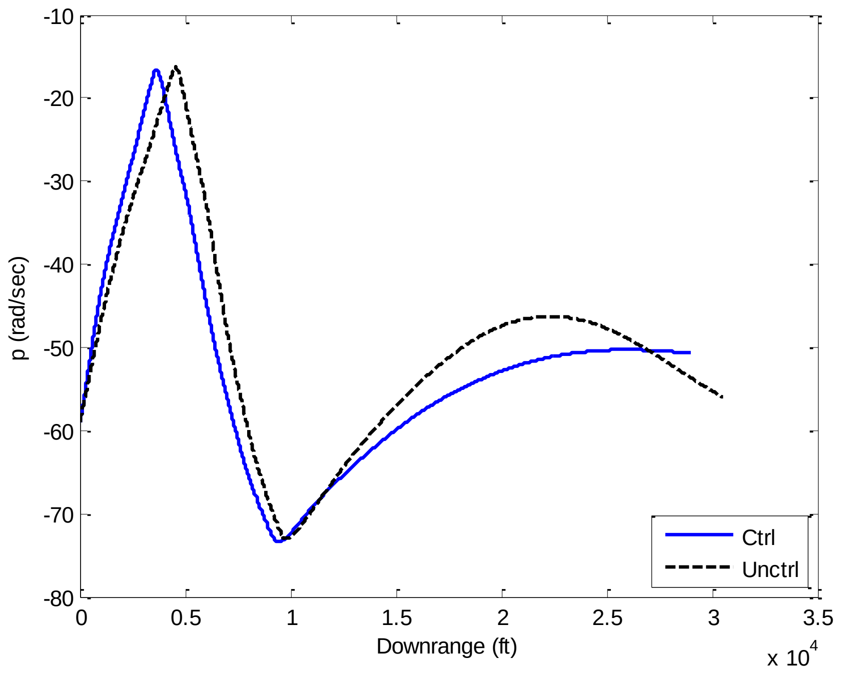


Figure 5.9: Roll rate comparisons

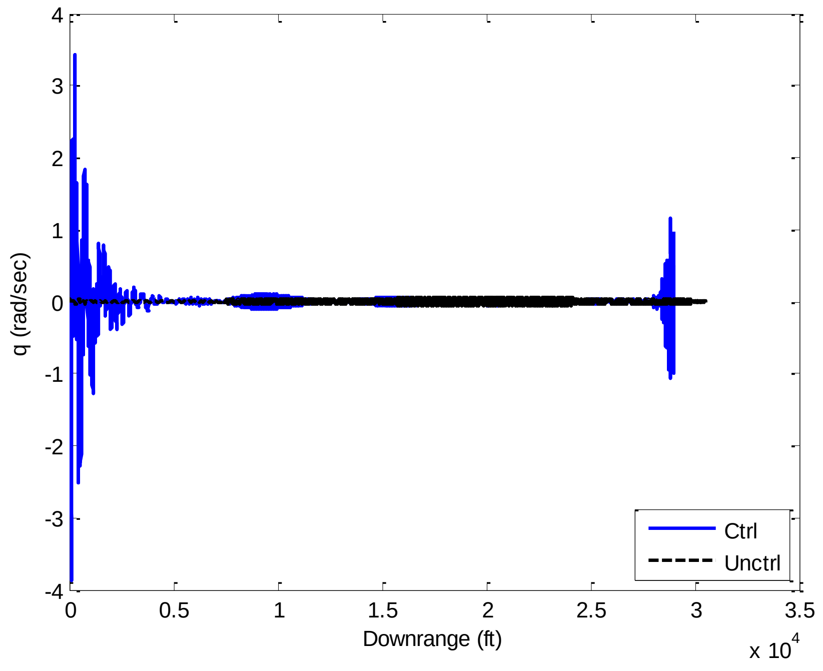


Figure 5.10: Pitch rate comparisons

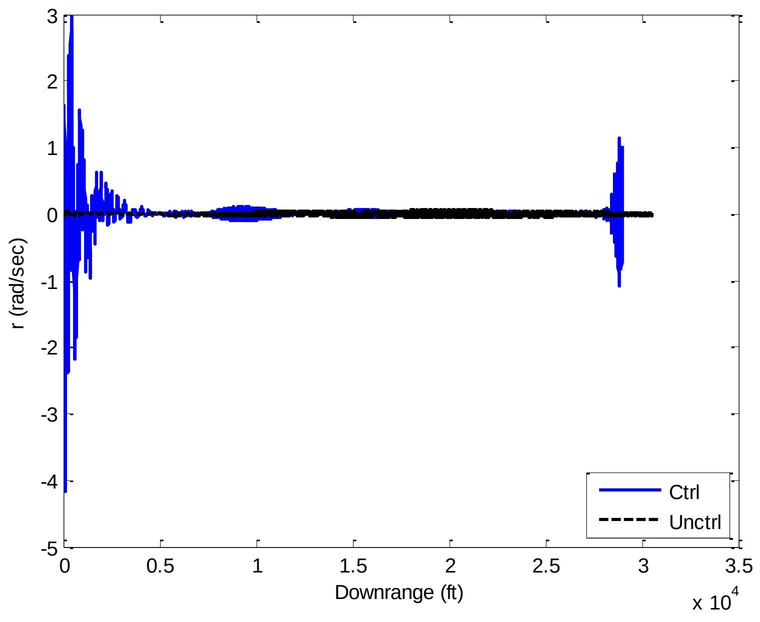


Figure 5.11: Yaw rate comparisons

When looking at Figures 5.1-5.11, it can be seen that most of the states do not differ drastically from their uncontrolled versions to their controlled versions. This is to be expected since the aim of the controller is manipulate a few of the states (*v, w, q, r*) to guide two of the states, *y* and *z*, to desired locations at the target.

The altitude and crossrange state histories serve to show the purpose and effectiveness of the controller. In the uncontrolled version, the projectile flies off course in the crossrange channel. Additionally, it drastically overshoots the target with respect to altitude. Ideally, the projectile would hit zero crossrange and zero altitude at the exact downrange distance of the target. In the controlled path history, the projectile immediately begins to steer back toward the target in both altitude and crossrange, impacting the target plane very close to the desired target point.

The yaw angle history differs in the controlled instance because the crossrange is directly tied to the yaw angle. Once the projectile begins to steer back toward the target, the yaw angle begins to vary. Similarly, the pitch angle is affected in the controlled instance. This is to be expected; the pitch angle in the controlled version is closely tied to the predicted pitch angles of the vacuum trajectory, which will be discussed later. Some of the oscillations in pitch and yaw are likely due to the coupling of control for crossrange and altitude. Conversely, the roll angle is largely unaffected by the implementation of control as the crossrange and altitude impacts are not affected by the projectile roll.

The I_B component of the projectile velocity is not affected much at all by control. However, the J_B and K_B components are affected. This is due to the desired method of implementing control. In the linear controller, the design works such that the aerodynamic trim forces exhibit control on the states *v* and *w*. Thus, once control begins, oscillations in these states become much larger in magnitude. Similarly, the pitch rate and yaw rate are affected by control. Again, their magnitudes increase drastically once control begins, whereas the roll rate remains fairly similar to its uncontrolled history.

5.2 Controller Robustness: Impact Point Dispersion Results

In order to demonstrate both the effectiveness and the robustness of the controller, a Monte Carlo dispersion set was created. The desire was to create a large set of launch conditions comprising varying yaw and pitch angles. The set of angles was designed such that the uncontrolled trajectories intersected a target plane on four sides of the target: top, bottom, left, and right. A depiction of this can be seen in Figure 5.12.

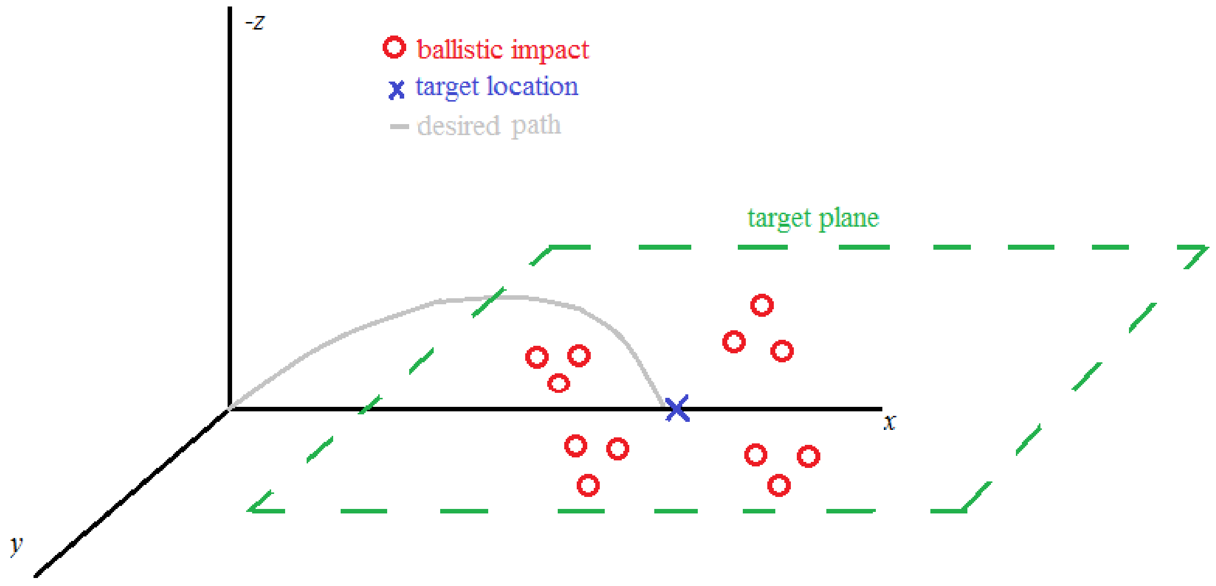


Figure 5.12: Target plane and desired Monte Carlo characteristics

The target plane is located in the *x*-*y* plane with zero altitude. The goal is that when each trajectory crosses zero altitude, it will have zero crossrange and will be exactly 29,000 feet downrange (i.e. right at the target location). The set of launch angles was created such that unguided impacts would occur all around the target plane. In other words, there would be unguided shots that both undershoot and overshoot the target and impact the target plane both to the left and the right of the *x*-axis.

A set of yaw angles was created via a random set of 125 numbers with zero mean and standard deviation of 0.0087 radians. The set of pitch angles was created in several chunks. First, it was determined to demonstrate controller performance on a range of pitch angles from 20°-60° to sufficiently show performance on a wide range of indirect fire launch angles. Thus, a downrange target location needed to be determined. Uncontrolled simulations were run to approximate the downrange distance at which various initial pitch angle shots impacted the target plane. Since these simulations were being run simply to get a rough estimate of where to place the target, the MPLT equations were used as a linear plant model for quickness. Figure 5.13 shows the results graphically.

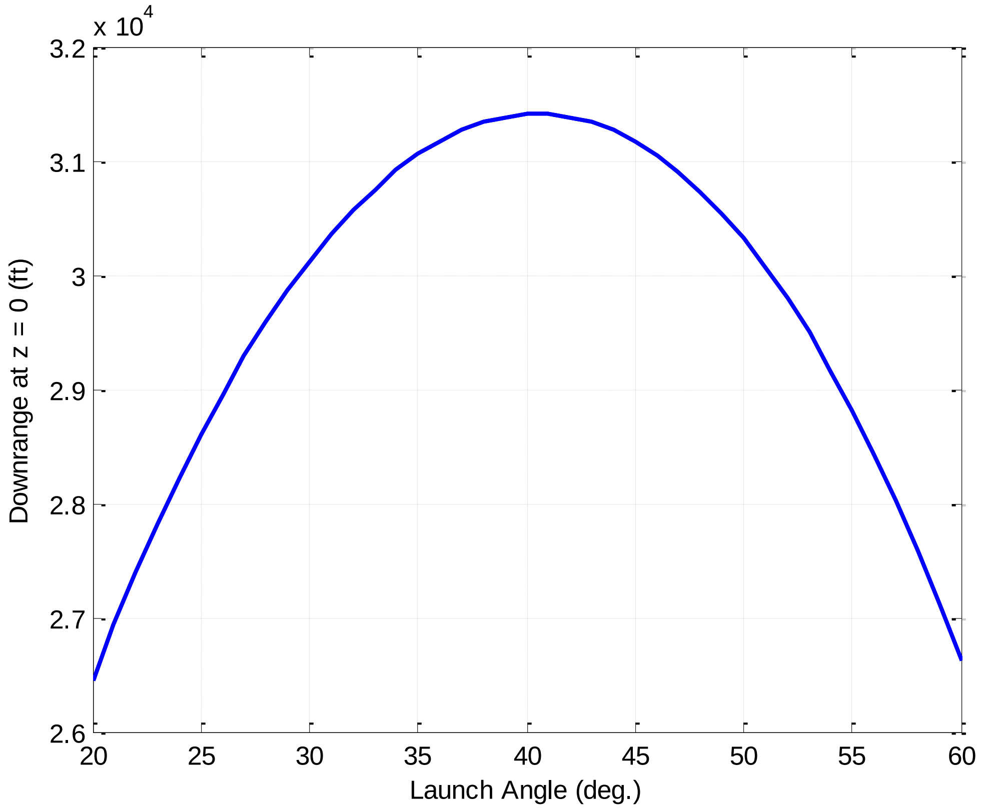


Figure 5.13: Construction of Monte Carlo pitch angles

From Figure 5.13 a target was chosen to be located at 29,000 feet downrange. This would mean that the target would be roughly equidistant from both extreme overshoots and extreme undershoots. It can be seen that launch angles of about 26° and 54° intersect the target plane at around 29,000 feet downrange. Thus, many of the points came from the ranges 20°-26° and 54°-60°. The others came from the range of 26°-54°. This ensured that many of the launches would undershoot the target and many of the launches would overshoot the target.

Thus, 25 random launches within the range 20°-26° were chosen with a 23° mean and 1.8° standard deviation. Additionally, 25 random launches from within the range 54°-60° were chosen with a 56.4° mean and a 1.4° standard deviation. Another 50 random launch angles were chosen from within the range 26°-54° with a mean of 40° and a standard deviation of 4.6°.

A final grouping of 25 pitch angles was chosen from two subgroups of 12 and 13 points respectively, with means of near 30° and 50° and small standard deviations of about 1.5°. These angles would provide uncontrolled shots which landed closer to the target and were not extreme undershoots or overshoots. This was done for completeness, such that there were not gaps in the input pitch angle distribution. The input angle pairs consisting of random yaw and pitch angles can be seen in Appendix E.

Using the set of launch angles, 125 controlled and uncontrolled shots were made with the full nonlinear plant model. The controlled trials were run with a control sampling period of 25-ms and with 50 segment predictions. A plot depicting the impact points is seen in Figure 5.14.

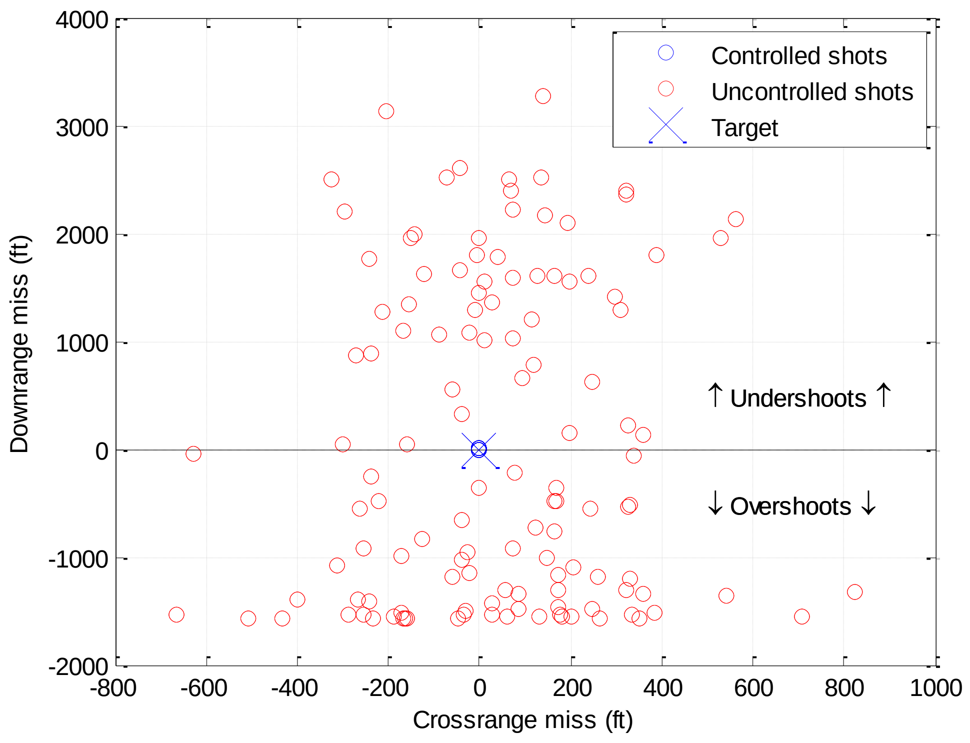


Figure 5.14: Ballistic and controlled impact points

As seen, the controlled impact point dispersion is virtually undetectable when compared to the uncontrolled impact point dispersion. In order to investigate the effectiveness of the controller, a closer look is taken into each set of impact points. To measure the performance, a CEP circle is drawn for each set. The CEP is a circle with a prescribed radius such that half of the points of a given set lie within the circle. Figure 5.15 shows the results.

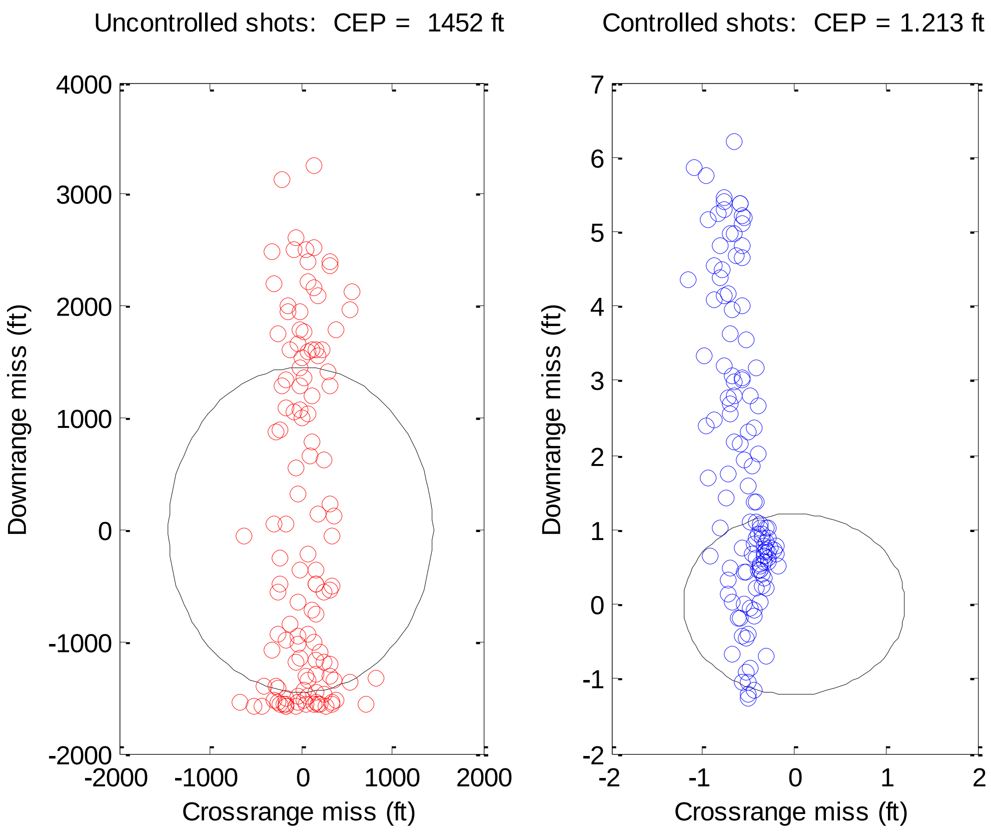


Figure 5.15: Ballistic and controlled impact points with CEP

In the uncontrolled case, the radius of the CEP is 1452 ft. In other words, the median miss distance in the uncontrolled flight simulations is over 1,450 ft. Conversely, in the controlled simulations, the radius of the CEP is brought down several orders of magnitude to just over one foot, at 1.213 ft to be exact.

5.3 Controller Trade Studies: Investigating Optimal Performance

There were several variable parameters within the controller which could have an impact on the overall performance and miss distances. These parameters were typically associated with a computational trade-off. The parameters could be set such that optimum accuracy was attained or they could be set such that computational time was kept to a desired value at the cost of performance. In this section, the variable parameters and their effects on controller performance will be discussed.

5.3.1 Impact of Segmentation on Performance

One of the traits of the controller in this thesis is the use of the vacuum model to make implicit trajectory predictions for a few of the parameters which cause nonlinearities in the MPLT projectile dynamics. The controller breaks the remaining distance into manageable segments and predicts an optimal trajectory for the projectile to follow. The success of the impact depends largely upon the amount of segments used in these implicit trajectory predictions. Generally, the more segments used, the more accurate the predictions are and the better control can be. However, adding more segments also means increased computing time and the introduction of more round-off error. Thus, the battle concerns a trade-off between accuracy and computational efficiency.

A simulation was run in order to investigate at what point diminishing returns are evident with respect to an increased number of segments in the trajectory predictions. Each of the simulations was run with a 25-ms control sampling period. The entire set of 125 yaw and pitch angles was used in the segmentation study. The values of segmentation used ranged from 30 to 50 segments. The results can be seen in Figure 5.16.

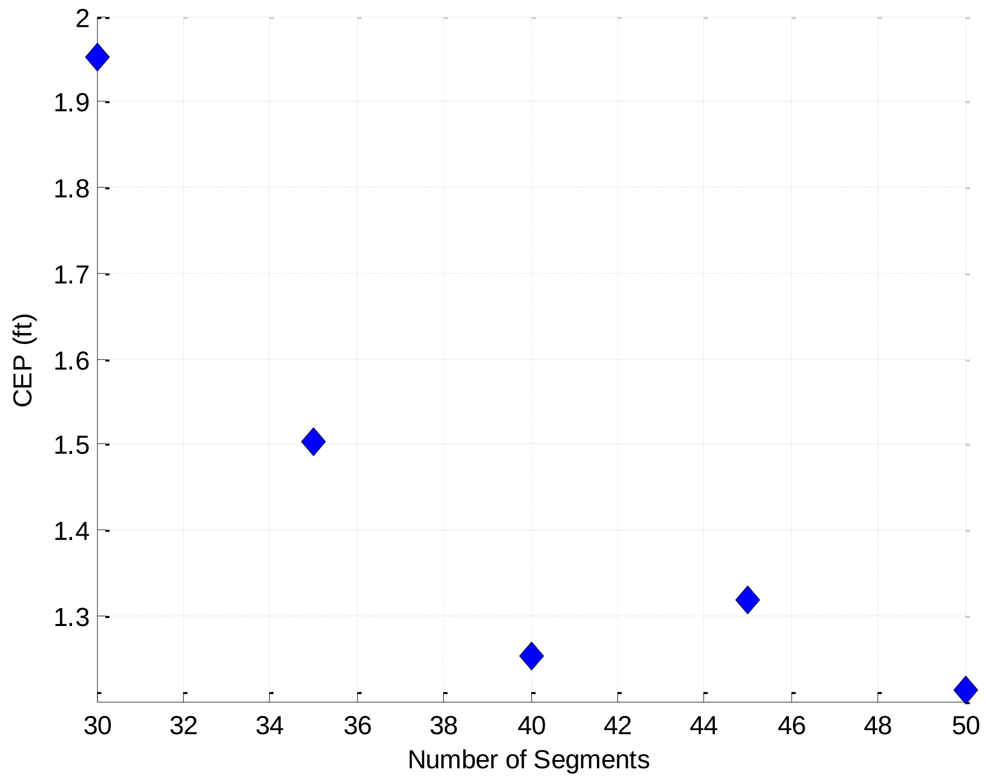


Figure 5.16: Impact of added segments on CEP

The CEP improves when going from 30 to 35 segments and from 35 to 40 segments. It jumps slightly in the 45 segment case. This could be due to excess round-off error in the simulations. The 50 segment case gives the best CEP, at 1.213 feet. Convesely, using 40 segments yields a CEP below 1.3 feet and requires much less computation. It should be noted that using segmentation levels below 30 caused poor predictions to be made and the controller would break down in many cases.

5.3.2 Impact of Sampling Period on Performance

Another aspect of the controller which has an impact on overall accuracy of the controlled shots is the control sampling period. Each time control is used, it adds computation time and space into the process. However, finer control sampling periods will generally mean improved results, or smaller miss distances at target plane impact.

In order to study the impact of varying the sampling period, the dispersion set of 125 values was used with 50 segments and with several different control sampling periods, ranging from as little as 10-ms to as large as 250-ms. In each case, a resulting CEP was calculated. The results are displayed in Figure 5.17.

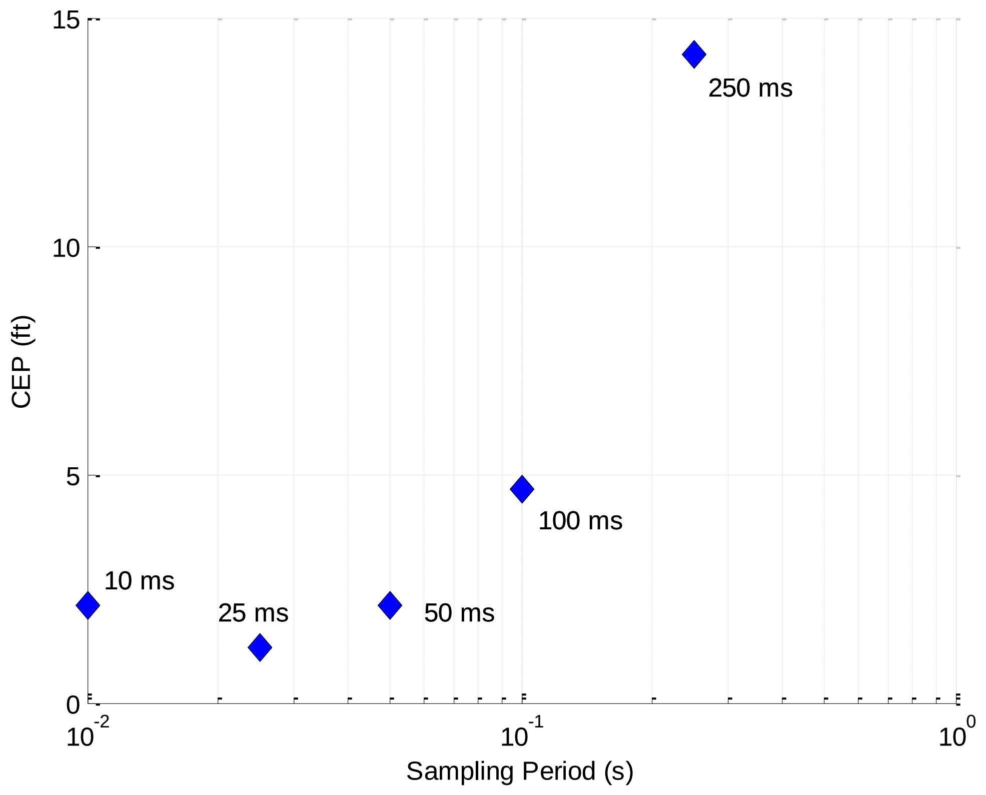


Figure 5.17: Impact of sampling period on CEP

As the sampling period decreases, the CEP is generally reduced. It appears that performance is greatly degraded for sampling periods above around 50 ms. However, there is a definite trend of diminishing return for sampling periods under 50 ms, with the CEP actually rising for the 10 ms case. This would suggest that sampling periods of around 25 ms would be ample in attaining decent accuracy while still maintaining manageable computational efficiency.

5.4 Vacuum Model Prediction Accuracy

One of the defining characteristics of the controller used in this thesis is the use of the point mass vacuum trajectory to model parameters which cause nonlinearities in the MPLT plant dynamics. As discussed in Chapter 4, the vacuum model is used to generate a theoretical trajectory from launch to target which the actual trajectory can attempt to duplicate. The intent is that, as the projectile moves downrange, its path becomes very similar to the path of the vacuum model. Figures 5.18-5.21 show the actual path histories of a successfully controlled trajectory for altitude and for the parameters which cause the nonlinearities. The figures also show overlays of the paths predicted by the vacuum model at the instant control begins and again once the projectile is halfway downrange. The black lines represent the successfully controlled full state history, while the red lines represent the predicted path given by the vacuum model immediately after launch, and the blue lines represent the predicted vacuum model path given when the projectile was halfway downrange. Fifty segments were used in each predicted path.

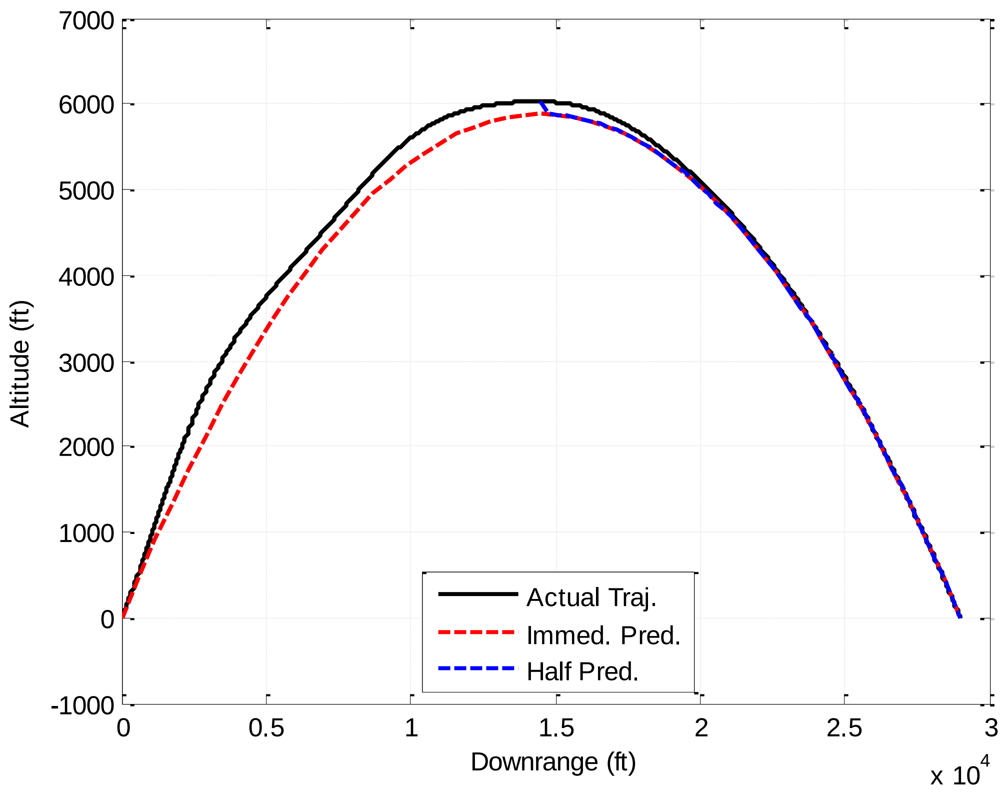


Figure 5.18: Prediction comparison for projectile altitude

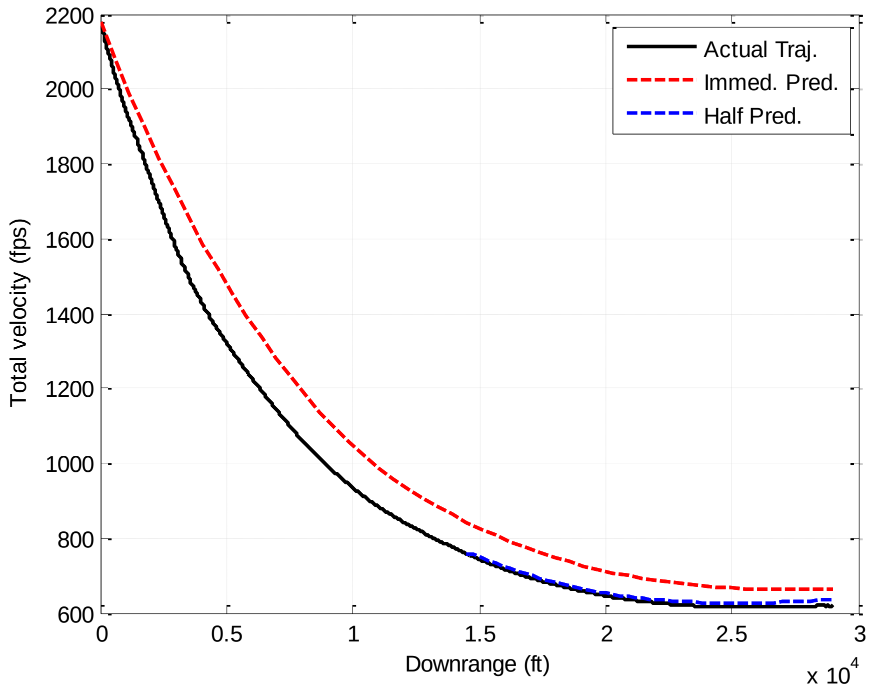


Figure 5.19: Prediction comparison for projectile velocity

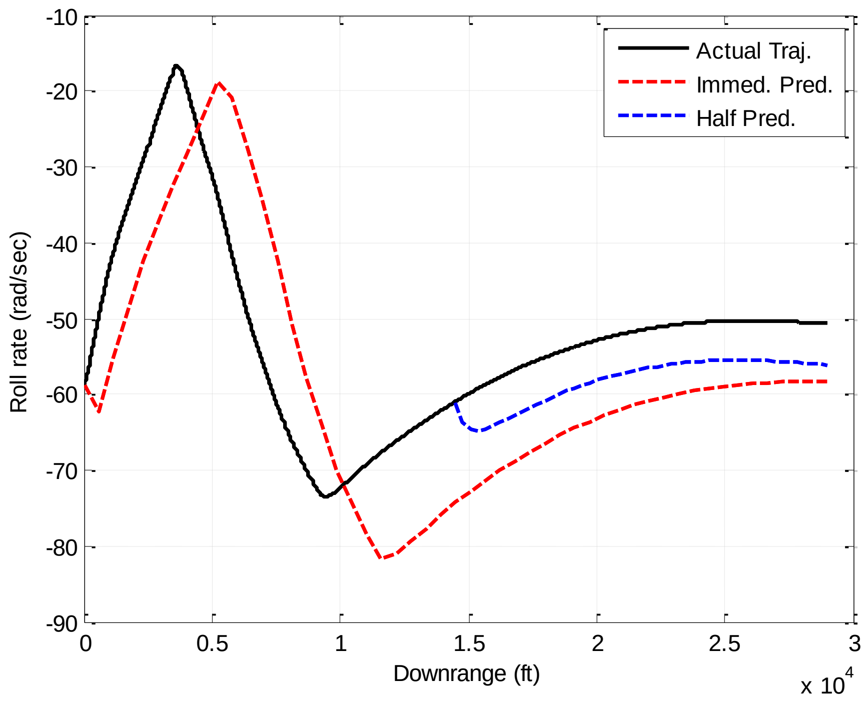


Figure 5.20: Prediction comparison for projectile roll rate

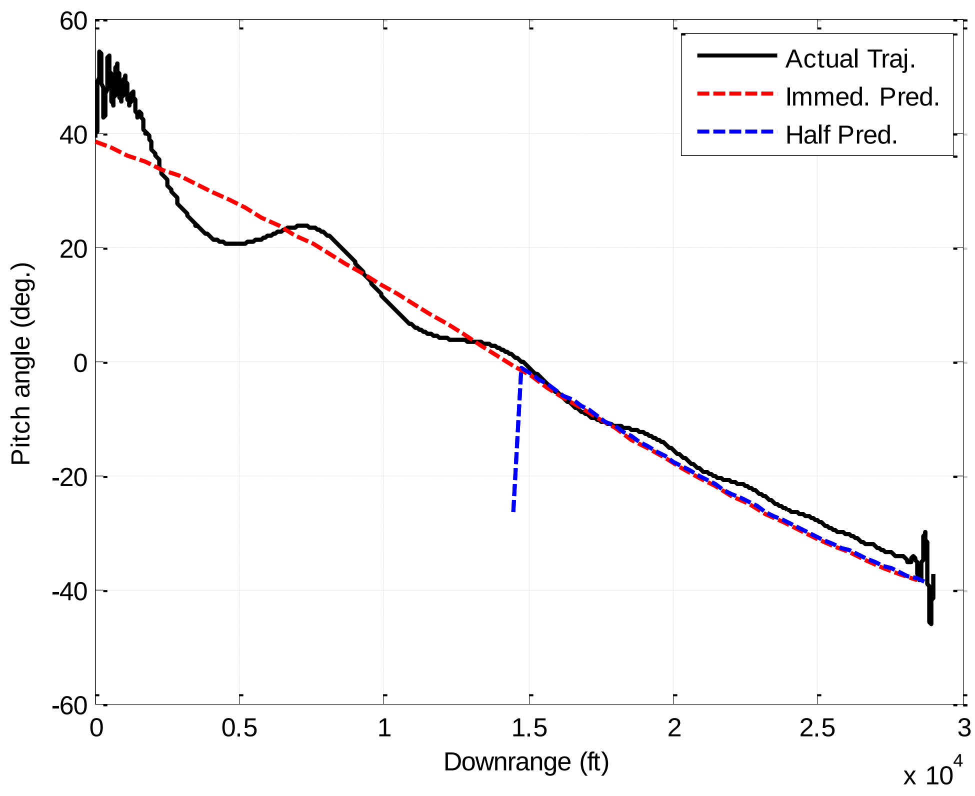


Figure 5.21: Prediction comparison for projectile pitch angle

As seen in Figures 5.18-5.21, the predictions given at the first instance in the control sampling period are very good. The figures show that the values for the states of the controlled trajectory do indeed closely follow the suggested paths given by the vacuum trajectory. Furthermore, the predicted parameters for *p* and *V* seem to become even more accurate as the projectile gets further downrange. The predictions for altitude and the pitch angles do not change much because they are solely dependent on the vacuum model, which is constant.

For the blue paths, the initial prediction spike for the pitch angle and altitude represents the first prediction made such that the trajectory assumes the path of the vacuum model. The current state of the projectile upon the call for control is not equal to that predicted by the vacuum model for that downrange location. The next predicted state lies on the vacuum model, thus the altitude and pitch angle predictions jump slightly initially in order to get the projectile back to the model trajectory.

Another view of the accuracy of the predictions can be seen when viewing Figure 5.22. The figure represents the actual trajectory tracking the vacuum model as it moves downrange. The left-hand vertical axis features the difference between the vacuum model and the actual trajectory in altitude at each step downrange. The right-hand vertical axis represents the state history of the added perturbation state. The figure shows that improving altitude control is directly correlated to the minimization of the perturbation state. This signals that the perturbation state is in fact acting to control the altitude to its desired value.

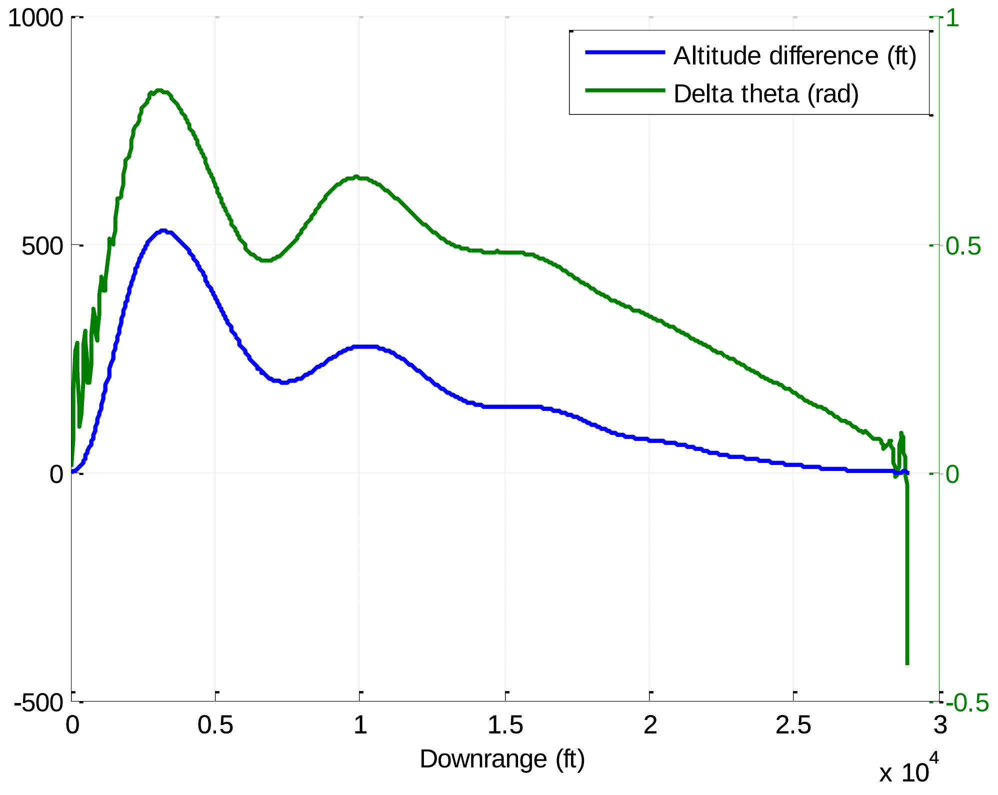


Figure 5.22: Improved altitude control via theta perturbation state

As seen in the figure, the projectile deviates from the vacuum model initially, but as it uses its control authority, the trajectory moves back to that of the model quite well, eventually leading to the desired target.

5.5 Vacuum Model Robustness

The model seems to work well for all launch angles tested. While 20° pitch angle launches will take a much more shallow path, steeper launch angles will have a higher trajectory. As shown in Figure 5.23, the vacuum model is capable of being used to guide the entire range of pitch angles to successful target impacts.

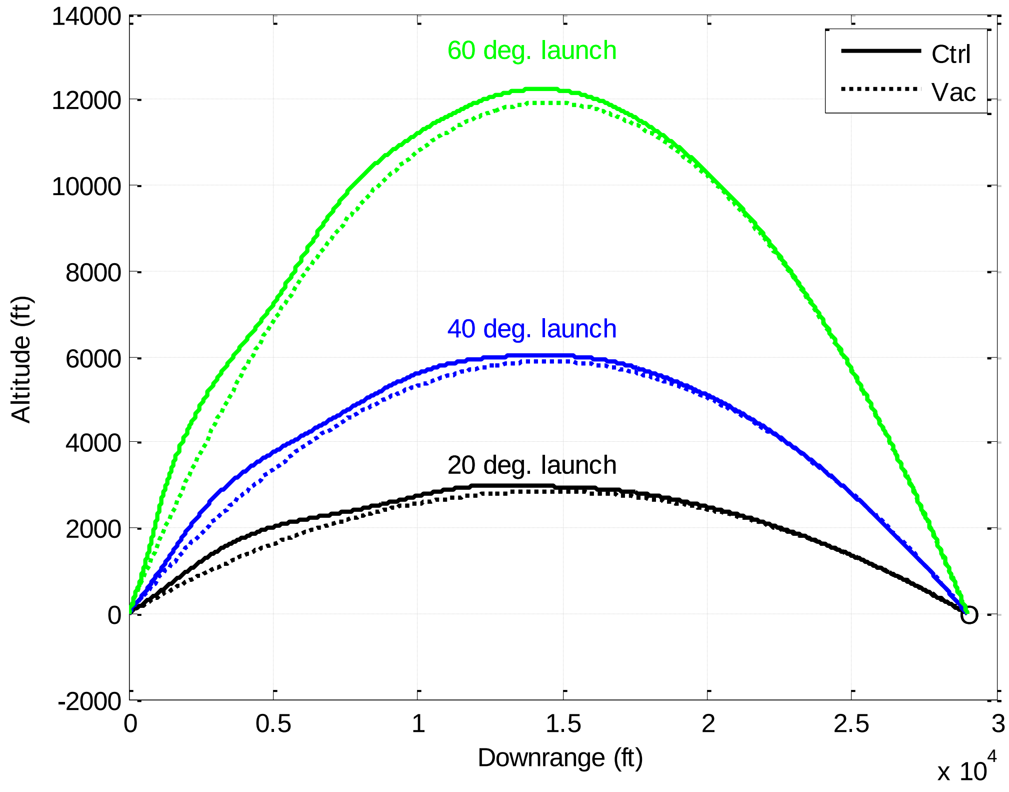


Figure 5.23: Robustness of using vacuum model for control

5.6 Tracking Miss Distances as a Function of Launch Angles

Most works in the area of missile guidance concern direct fire launches, or launches in which a direct line of sight exists between the missile nose and the target. In this thesis, however, higher launch angles were tested such that no direct line of sight exists. It was desired to investigate whether the controller worked best for a certain range of initial pitch angles. Thus, the miss distances were plotted as a function of the input pitch angles. The results are seen in Figure 5.24.

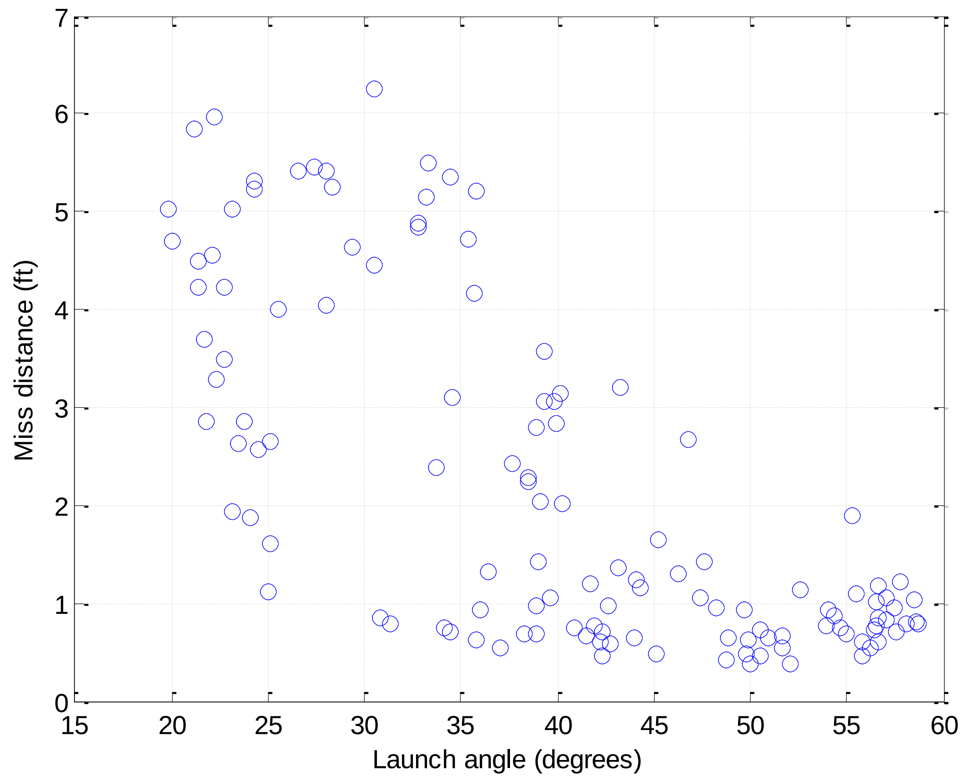


Figure 5.24: Miss Distances resulting from input pitch angle dispersion

The figure shows that the miss distances seem to be somewhat correlated with initial pitch angle. For higher launch angles, the misses are smaller at the target plane. While the largest misses are still only around 6 ft in magnitude, it is clear that most of the larger misses occur for the smaller values of input pitch angle, while most of the input pitch angles which are higher than 50° result in misses of around one foot or less.

5.7 Observing the Canard Actuator Deflections

A final aspect of control implementation involves the canard actuators and their deflections which result in the application of desired aerodynamic reactions on the projectile. Recall that the canards are to have pitch angle deflections limited to [-1, 1] radians in the no-roll frame. A sample plot of a time history of the canard deflections is seen in Figure 5.25.

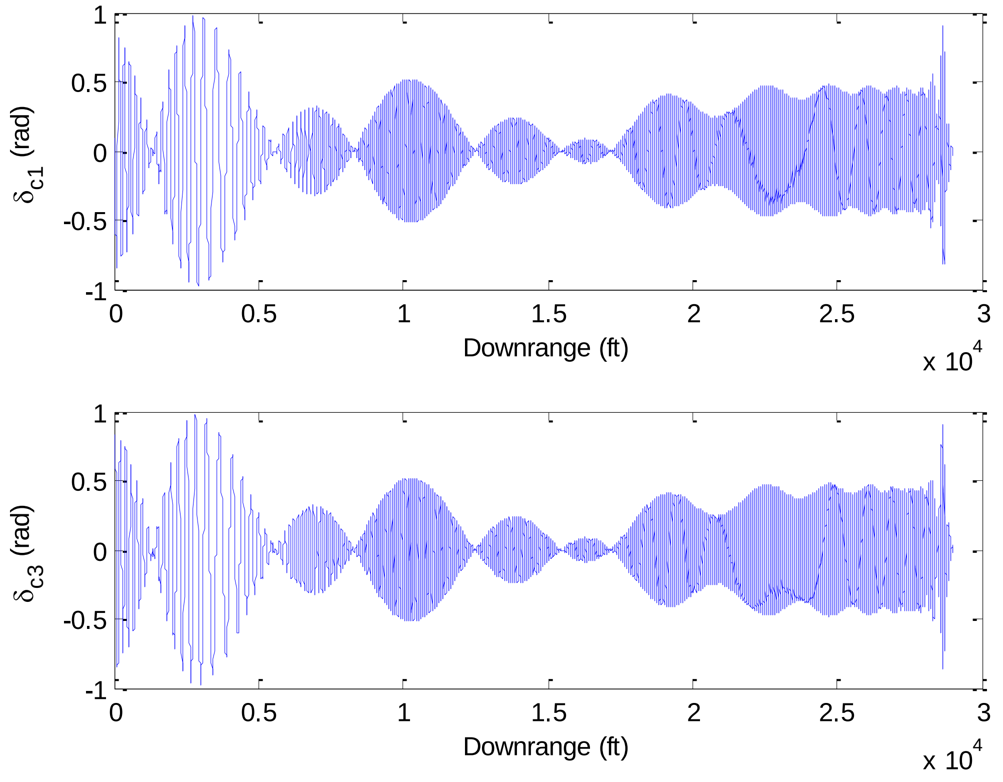


Figure 5.25: Commanded canard deflections

Immediately as control begins, the canards begin to deflect and oscillate. The magnitude of the deflections is larger for the first few thousand feet downrange as the projectile attempts to track the vacuum model. The commands then remain well below saturation until the target becomes near, upon which the deflections ramp up in a final attempt to hit the target. The deflections near the beginning and the end of some trajectories may actually reach a magnitude of more than one radian. This is because the deflections are limited to a saturation value in their local frame. When unrolling these angles, the resulting deflections may become slightly lower and slightly higher than one radian.

5.8 Summarizing the Results

Throughout the chapter, results have been presented on the findings of the work in this thesis. Dispersion plots were generated and parameters were optimized within the controller to enable optimal performance. It was found that using 50 segments in the implicit trajectory predictions seemed to work best, and a control sampling period of about 25-ms was ample for proper control.

Using the optimum parameters and conditions, the dispersions were run and the CEP was reduced from over 1,450 ft in the uncontrolled tests to just over one ft in the controlled trials. Additionally, it was demonstrated that the use of the vacuum model predictions gave the controlled trajectory a successful way of getting to the target. It was also shown that the use of the theta perturbation state effectively controlled the altitude state to its desired value at the target.

In the following chapter, the work in this thesis will be briefly summarized. Conclusions will be drawn from the work performed. Additionally, future work will be proposed.

6. Conclusions

Throughout the duration of this thesis, many aspects of modeling and controlling indirect fire projectile flight have been discussed and presented. Before the beginning of this work, controllers and models were created to simulate control of direct fire launches in [7]. That work was extended to incorporate control to indirect fire launches, or high launches, which has been documented in this thesis.

6.1 Summarizing Work Performed in this Thesis

In this thesis, a six-degree-of freedom plant model was developed and used to simulate the dynamics of a Hydra-70 rocket in flight. Two pairs of forward-mounted controllable canards were used as actuators in attempting to control the flight to a desired location. It was desired to control both the crossrange and altitude of the projectile to zero values at a downrange target located 29,000 ft from the launch spot. A target plane was created which evaluated both the downrange and crossrange positions at the time at which the projectile hit zero altitude.

A nine-dimensional state-space model was used with MPLT equations in order to create a linear controller to control the projectile to its desired target. By using basic physics and closed-form analytical expressions, a vacuum trajectory was created and predictions were made to force the projectile to follow the vacuum model. Additionally, a perturbation state was used to create a path between altitude and the control input. This added state was shown to directly improve altitude control.

A dispersion plot was generated which showed the effectiveness and robustness of the controller. Unguided shots were run for a wide range of initial launch angles and a CEP was found to have a radius of over 1,450 ft. The same shots were made with the controller in place and the CEP was reduced to just over one ft.

6.2 Future Work

The work in this thesis and its preceding work was done with a fin-stabilized projectile. Fin-stabilized projectiles are commonly used for direct fire launches and are characterized by the presence of canted fins located near the base of the rocket [15]. Stability is obtained by giving the leading edge of the fins a cant angle and imparting small spin rates on the projectile initially.

In the realm of indirect fire launches, it is customary to use spin-stabilized munitions. Spin-stabilized projectiles are used to promote flight stability in long-range shots. The spinning is created by the firing of the projectile through a rifled tube. A series of rotating bands on the projectile engages the barrel rifling, which in turns causes high spin rates to be imparted onto the munition [15].

Because the work preceding this thesis concerned direct fire launches, a fin-stabilized rocket was being used. Therefore, the model which was already in place was used. Future work would concern employing the controller used in this thesis on a spin-stabilized rocket.

Another area of future work would be to tighten the control authority and use a smaller range of input pitch angles. Control authority is usually low for typical smart projectiles. It is common for actuators on an indirect fire smart projectile to modify the flight by 200 m for a range of 20,000 m [16]. This ensures acceptable values of angle of attack and promotes flight stability.

In this thesis, the actuators were allowed to saturate at one radian, or around 60°. It was desired to use a large range of input pitch angles to demonstrate the ability of the MPLT controller to successfully alter a wide range of shots. Future work would be to limit this saturation and tune the control weighting matrices to optimize flight for a smaller range of initial pitch angles which could use much less control authority, yielding small angles of attack throughout flight.

6.3 Final Acknowledgment

Finally, I would like to express my gratitude to Dr. Bradley Burchett for his help throughout the duration of this research. The opportunity to work on this thesis and its preceding work has been very rewarding and enjoyable. I would also like to thank the faculty and staff in the Rose-Hulman Mechanical Engineering department. They have been helpful beyond words in my time as a student.

References

[1] NASA, "Guidance, Navigation, and Control Technology Assessment for Future Planetary Science Missions," Jet Propulsion Laboratory, Pasadena, CA, 2013.

[2] Federation of American Scientists: Military Analysis Network, "Fundamentals of Naval Weapons Systems," [Online]. Available: http://fas.org/man/dod-101/navy/docs/fun/index.html. [Accessed 30 January 2015].

[3] B. Burchett and M. Costello, "Model Predictive Lateral Pulse Jet Control of an Atmospheric Rocket," *Journal of Guidance, Control, and Dynamics,* vol. 25, pp. 860-867, 2002.

[4] L. C. Hainz and M. Costello, "Modified Projectile Linear Theory for Rapid Trajectory Prediction," *Journal of Guidance, Control, and Dynamics,* vol. 28, pp. 1006-1014, 2005.

[5] R. L. McCoy, Modern Exterior Ballistics, Atglen, PA: Schiffer Publishing Ltd., 1999.

[6] D. Ollerenshaw and M. Costello, "Model Predictive Control of a Direct Fire Projectile Equipped with Canards," in *AIAA Atmospheric Flight Mechanics Conference and Exhibit,* San Francisco, CA, 2005.

[7] B. T. Burchett and A. L. Nash, "Euler-Lagrange Optimal Control for Symmetric Projectiles," in *AIAA Science and Technology Forum and Exposition,* Kissimmee, FL, 2015.

[8] H. J. Sussmann and J. C. Willems, "300 Years of Optimal Control: From the Brachystochrone to the Maximum Principle," *IEEE Control Systems,* vol. 17, pp. 32-44, 1997.

[9] A. E. Bryson, Jr. and Y.-C. Ho, Applied Optimal Control, Waltham, MA: Ginn and Company, 1969.

[10] B. T. Burchett, *Advanced Control Systems, Lecture Notes,* Terre Haute, IN: Rose-Hulman Institute of Technology, 2014.

[11] M. D. Weir, J. Hass and F. R. Giordano, Thomas' Calculus: Early Trancendentals, 11th ed., Boston, MA: Pearson, 2008.

[12] L. Dou and J. Dou, "The design of optimal guidance law with multi-constraints using block pulse functions," *Aerospace Science and Technology,* vol. 21, pp. 201-205, 2012.

[13] F. Fresconi, I. Celmins and S. I. Silton, "Theory, Guidance, and Flight Control for High Maneuverability Projectiles," U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, 2014.

[14] Federation of American Scientists: Military Analysis Network, "U.S. Missiles: Hydra-70 Rocket System," [Online]. Available: http://fas.org/man/dod-101/sys/missile/hydra-70.htm. [Accessed 20 January 2015].

[15] Federation of American Scientists: Military Analysis Network, "U.S. Land Warfare Systems: Big Bullets for Beginners," [Online]. Available: http://fas.org/man/dod-101/sys/land/bullets2.htm. [Accessed 15 March 2015].

[16] F. Fresconi, G. Cooper and M. Costello, "Practical Assessment of Real-Time Impact Point Estimators for Smart Weapons," *Journal of Aerospace Engineering,* vol. 24, pp. 1-11, 2011.

Appendices

Appendix A: MATLAB Code Used in Thesis

aero_props_find.m: used to get aero coeffs for nonlinear plant

```
function coeffs = aero_props_find(Mach)
% output:  aero coeffs for nonlinear plant model
% input:  Mach number

% load the data which has been defined as a global variable

global aerodata_nl

% coeff order in vector:  CX0,CX2,CYB1,CZA1,CDD,CLP,CMQ,CNR,RCAX
% use interp1 to get correct coeffs

coeffs(1) = interp1(aerodata_nl(:,1),aerodata_nl(:,3),Mach);
coeffs(2) = interp1(aerodata_nl(:,1),aerodata_nl(:,9),Mach);
coeffs(3) = interp1(aerodata_nl(:,1),aerodata_nl(:,15),Mach);
coeffs(4) = interp1(aerodata_nl(:,1),aerodata_nl(:,16),Mach);
coeffs(5) =interp1(aerodata_nl(:,1),aerodata_nl(:,6),Mach);
coeffs(6) = interp1(aerodata_nl(:,1),aerodata_nl(:,5),Mach);
coeffs(7) = interp1(aerodata_nl(:,1),aerodata_nl(:,7),Mach);
coeffs(8) = interp1(aerodata_nl(:,1),aerodata_nl(:,11),Mach);
coeffs(9) = (2.5 -
interp1(aerodata_nl(:,1),aerodata_nl(:,2),Mach))*-1;

end
```

aero_props_lin.m: used to get aero coeffs for linear controller

```
function coeffs = aero_props_lin(Mach)
% output:  aero coeffs for linear controller
% input:  Mach number


% load the data which has been stored as a global variable

global aerodata_l

% find the appropriate coeffs using interp1

coeffs(1) = (2.5 -
interp1(aerodata_l(:,1),aerodata_l(:,2),Mach))*-1;
coeffs(2) = interp1(aerodata_l(:,1),aerodata_l(:,3),Mach);
coeffs(3) = interp1(aerodata_l(:,1),aerodata_l(:,4),Mach);
coeffs(4) = interp1(aerodata_l(:,1),aerodata_l(:,6),Mach);
coeffs(5) = interp1(aerodata_l(:,1),aerodata_l(:,5),Mach);
coeffs(6) =interp1(aerodata_l(:,1),aerodata_l(:,8),Mach);
coeffs(7) = interp1(aerodata_l(:,1),aerodata_l(:,7),Mach);

end
```

canard_aero_find.m: used to get aero props for canards

```
function coeffs = canard_aero_find(Mach)

% load the data

%aerodata_c = csvread('canard_aero_props.csv');
global aerodata_c

% coeff order in vector:  CL,CL3,CL5,CD0,CD2,Ci
% use interp1 to get correct coeffs
if Mach > .4 & Mach < 1
coeffs(1) = interp1(aerodata_c(:,1),aerodata_c(:,2),Mach);
coeffs(2) = interp1(aerodata_c(:,1),aerodata_c(:,3),Mach);
coeffs(3) = interp1(aerodata_c(:,1),aerodata_c(:,4),Mach);
coeffs(4) = interp1(aerodata_c(:,1),aerodata_c(:,5),Mach);
coeffs(5) = interp1(aerodata_c(:,1),aerodata_c(:,6),Mach);
coeffs(6) = interp1(aerodata_c(:,1),aerodata_c(:,7),Mach);

else
coeffs(1) =
interp1(aerodata_c(:,1),aerodata_c(:,2),Mach,'nearest','extrap')
;
coeffs(2) =
interp1(aerodata_c(:,1),aerodata_c(:,3),Mach,'nearest','extrap')
;
coeffs(3) =
interp1(aerodata_c(:,1),aerodata_c(:,4),Mach,'nearest','extrap')
;
coeffs(4) =
interp1(aerodata_c(:,1),aerodata_c(:,5),Mach,'nearest','extrap')
;
coeffs(5) =
interp1(aerodata_c(:,1),aerodata_c(:,6),Mach,'nearest','extrap')
;
coeffs(6) =
interp1(aerodata_c(:,1),aerodata_c(:,7),Mach,'nearest','extrap')
;

end
```

canard props.m: stores properties for canards

```
function [SL,BL,WL,GAMCAN,PHICAN,DELTACAN,SCAN] = canard_props()

% this function has no inputs: it is just storing the properties 
for
% each canard:  load them when needed

  SL(1) = 3.8;
  BL(1) = 0.14167;
  WL(1) = 0;
  GAMCAN(1) = 0;
  PHICAN(1) = 0;
  DELTACAN(1) = 0;
  SCAN(1)   = 0.00433516492;

  SL(2) = 3.8;
  BL(2) = -0.14167;
  WL(2) = 0;
  GAMCAN(2) = 0;
  PHICAN(2) = pi;
  DELTACAN(2) = 0;
  SCAN(2)   = 0.00433516492;

  SL(3) = 3.8;
  BL(3) = 0;
  WL(3) = 0.14167;
  GAMCAN(3) = 0;
  PHICAN(3) = pi/2;
  DELTACAN(3) = 0;
  SCAN(3)   = 0.00433516492;

  SL(4) = 3.8;
  BL(4) = 0;
  WL(4) = -0.14167;
  GAMCAN(4) = 0;
  PHICAN(4) = -pi/2;
  DELTACAN(4) = 0;
  SCAN(4)   = 0.00433516492;

end
```

canard_reactions.m: used to compute force/moments contributions of canards

```
function [Xc,Yc,Zc,Lc,Mc,Nc] = 
canard_reactions(states,rho,c,t,coeffs,xt,dt,flag)
```

```
% output:  total force and moment contributions of canards
% inputs:  current states, air density, speed of sound, current
time, aero
%          coeffs, target location, sampling period for ctrl,
flag to
%          signal for control

% grab constants and parameters needed; redefine some inputs

[D,m,g,Ixx,Iyy,Izz,S,Scan,a] = params(); RHO = rho;
x = states(1); y = states(2); z = states(3); psi = states(4);
theta = states(5); phi = states(6); q_til = states(11);
r_til = states(12);
u_til = states(7); v_til = states(8); w_til = states(9);
p_til = states(10);
[SL,BL,WL,GAMCAN,PHICAN,DELTACAN,SCAN] = canard_props();

% define canard defl. in no-roll frame as a global variable

global zetanr
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% hold off on control initially; singularity issue
if x < 1

zetanr(1,1) = 0;
zetanr(2,1) = 0;

else
    % if time for ctrl call has arrived, go to controller
function;
    % if not, hold on to previous no-roll canard commands

    if flag == 1
        controls = control_func(states,50,t,xt);
        if controls(1) <=1 & controls(2) <=1
        zetanr(1,1) = controls(1);
        zetanr(2,1) = controls(2);
        end
    else
        dummy = 1;
    end

end


% transform canards commands to proper frame
```

```
transmat = [cos(phi),sin(phi);-sin(phi),cos(phi)];
zetar = transmat*zetanr;
DELTACAN(1) = zetar(1); DELTACAN(3) = zetar(2);
DELTACAN(2) = -DELTACAN(1);
DELTACAN(4) = -DELTACAN(3);

% initialize reactions to zero

XCANFORCE = 0;
YCANFORCE = 0;
ZCANFORCE = 0;
LCANMOMENT = 0;
MCANMOMENT = 0;
NCANMOMENT = 0;

% start process for canard reaction contributions:  4 canards,
sum the
% total

SLCG = 2.5; BLCG = 0; WLCG = 0;

    for i = 1:4

        a = 1;
        rxcan = (SL(i) - SLCG)*a;
        rycan = (BL(i) - BLCG)*a;
        rzcan = (WL(i) - WLCG)*a;

        A1 = [0,rzcan,-rycan;-rzcan,0,rxcan;rycan,-rxcan,0];
        A2 = [cos(GAMCAN(i)),cos(PHICAN(i))*sin(GAMCAN(i)),...
            sin(PHICAN(i))*sin(GAMCAN(i));-sin(GAMCAN(i)),...

cos(PHICAN(i))*cos(GAMCAN(i)),sin(PHICAN(i))*cos(GAMCAN(i));...
            0,-sin(PHICAN(i)),cos(PHICAN(i))];

        uvw_pt =  [u_til,v_til,w_til]' + A1*[p_til;q_til;r_til];
        uvw_can = A2*uvw_pt;


        alfacan = DELTACAN(i) + atan2(uvw_can(3),uvw_can(1));
        machcan = sqrt(uvw_can(1)^2 + uvw_can(3)^2)/c;

        coeffs1 = canard_aero_find(machcan);
        cl1can = coeffs1(1); cl3can = coeffs1(2); cl5can =
coeffs1(3);
```

```
        cd0can = coeffs1(4); cd2can = coeffs1(5); cican =
coeffs1(6);

        clcan = cl1can*alfacan + cl3can*alfacan^3 +
cl5can*alfacan^5;
        cdcan = cd0can + cd2can*alfacan^2 + cican*clcan^2;

        liftcan = RHO*(uvw_can(1)^2 +
uvw_can(3)^2)*SCAN(i)*clcan/2;
        dragcan = RHO*(uvw_can(1)^2 +
uvw_can(3)^2)*SCAN(i)*cdcan/2;

        sangle = sin(alfacan - DELTACAN(i));
        cangle = cos(alfacan - DELTACAN(i));
        xlcan = liftcan*sangle - dragcan*cangle;
        ylcan = 0;
        zlcan = -liftcan*cangle - dragcan*sangle;

        xforcec = cos(GAMCAN(i))*xlcan + sin(GAMCAN(i))*zlcan;
        yforcec = sin(PHICAN(i))*sin(GAMCAN(i))*xlcan...
            -sin(PHICAN(i))*cos(GAMCAN(i))*zlcan;
        zforcec = -cos(PHICAN(i))*sin(GAMCAN(i))*xlcan...
            + cos(PHICAN(i))*cos(GAMCAN(i))*zlcan;

        lmomentc = rycan*zforcec - rzcan*yforcec;
        mmomentc = rzcan*xforcec - rxcan*zforcec;
        nmomentc = rxcan*yforcec - rycan*xforcec;

        XCANFORCE = XCANFORCE + xforcec;
        YCANFORCE = YCANFORCE + yforcec;
        ZCANFORCE = ZCANFORCE + zforcec;
        LCANMOMENT = LCANMOMENT + lmomentc;
        MCANMOMENT = MCANMOMENT + mmomentc;
        NCANMOMENT = NCANMOMENT + nmomentc;

    end

    Xc = XCANFORCE; Yc = YCANFORCE; Zc = ZCANFORCE;
    Lc = LCANMOMENT; Mc = MCANMOMENT; Nc = NCANMOMENT;

end
```

control_func.m: used to compute no-roll control input

```
function u = control_func(states,Ns,t,xt)

% define necessary global variables
```

```
global count2
global M
global N2

% preallocate for segments calculations
x = zeros(1,Ns+1); yp = x; p = x; V = x; h = zeros(1,Ns); cth =
h; sth = h;

% load parameters; define states
[D,m,g,Ixx,Iyy,Izz,S,Scan,a] = params();
X = states; x(1) = X(1); y = X(2); z = X(3); psi = X(4); theta =
X(5); phi = X(6);
% rotate velocities into proper frame
A1 = [cos(phi),-sin(phi);sin(phi),cos(phi)];
qr = A1*[X(11);X(12)];
vw = A1*[X(8);X(9)];
v = vw(1); w = vw(2); %u = X(7);
q = qr(1); r = qr(2); p(1) = X(10);

% compute total V initial
V(1) = sqrt(X(7)^2 + X(8)^2 + X(9)^2);

% control weighting matrices
R = eye(2)*10^2;
P = diag([1000,10000,1000,1,1,1,1,1,0]);

% uncontrollable mode
w_prime = 1;

% if first time calling for ctrl, compute vacuum model coeffs
if count2 == 0
    [M,N2] = vac_pseudo(x(1),z,xt);
    count2 = 1;
else
    dummy2 = 1;
end

% compute dx; set up dz (called yp) for first segment to get
onto vac model
dx = (xt-x(1))/(Ns);
yp(1) = -z;
yp2 = M*(x(1)+dx) + N2*((x(1)+dx))^2;

% compute theta perturbation state and set up state vector
th_0 = atan((yp2 - yp(1))/dx);
dth = (theta - th_0);
```

```
xs = [y,z,dth,psi,v,w,q,r,w_prime]';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

% loop through and calculate prediction parameters and
Hamiltonians
for k = 1:Ns

    % grab segment properties and coeffs based on V and altitude
    [Mach,rho,c] = mach_find(V(k),-yp(k));
    coeffs = aero_props_lin(Mach);
    RMCP = coeffs(1); CX0 = coeffs(2); CNA = coeffs(3); CDD =
coeffs(4);
    CLP = coeffs(5); CYPA = coeffs(6); CMQ = coeffs(7); CMA =
coeffs(1)*CNA;

    % compute angle prediction
    x(k+1) = x(k) + dx;
    yp(k+1) = M*x(k+1) + N2*x(k+1)^2;
    dy = yp(k+1) - yp(k);
    ds = sqrt(dx^2 + dy^2);

    cth(k) = dx/ds;
    sth(k) = dy/ds;

    % set up state matrices
    cap_phi = [0,0,0,D*cth(k);0,0,-D*cth(k),0;0,0,0,0;0,0,0,0];
    cap_lam = [0,D*g/V(k)*cth(k),0,0]';
    cap_gam =
D/V(k)*[1,0,0,0;0,cth(k),0,0;0,0,1,0;0,0,0,1/cth(k)];
    cap_sig = [0,-D*sth(k),0,0]';
    casi1 = rho*S*D/(2*m)*CNA;
    casi3 = rho*S*D/(2*Iyy)*CMA;
    casi4 = rho*S*D^3/(4*Iyy)*CMQ;
    casi5 = D/V(k)*Ixx*p(k)/Iyy;

    casi = [-casi1,0,0,-D;0,-casi1,D,0;0,casi3,casi4,-casi5;...
        -casi3,0,casi5,casi4];

    b1 = rho*Scan*D/(2*m)*V(k);
    b2 = rho*Scan*D/(2*Iyy)*V(k)*(3.8-2.5);
    b = [0,-b1,b2,0;b1,0,0,b2]';

    A = [cap_phi,cap_gam,cap_sig;zeros(4,4),casi,cap_lam;...
        zeros(1,4),zeros(1,4),0];
    B = [zeros(4,2);b;zeros(1,2)];
    % compute current Hamiltonian

    F{k} = [A,-B*inv(R)*B';zeros(9,9),-A'];

    % compute arc-length to next state; find next values of p,V
    h(k) = ds/D;

    av = (pi*rho*D^3/(8*m))*CX0; bv = g*D*sth(k);
    Cp0 = 2*CDD*V(k)/(D*CLP); Cpe1 = p(k) + 2*CDD*V(k)/(D*CLP);
    Cpe2 = pi*rho*D^5/(16*Ixx)*CLP;
    p(k+1) = Cpe1*exp(Cpe2*h(k))-Cp0;
    V(k+1) = sqrt((V(k)^2+bv/av)*exp(-2*av*h(k))-bv/av);

end

% back propagate to get Z(1)
Z{Ns} = (eye(18) + h(Ns)/2*F{Ns})\[eye(9);P];

for k = Ns-1:-1:1
    Z{k} = ((eye(18)+h(k)/2*F{k}))\((eye(18)-
h(k)/2*F{k+1})*Z{k+1}) ;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

% values at current state
[Mach,rho,c] = mach_find(V(1),z);
b1 = rho*Scan*D/(2*m)*V(1);
b2 = rho*Scan*D/(2*Iyy)*V(1)*(3.8-2.5);
b = [0,-b1,b2,0;b1,0,0,b2]';

B = [zeros(4,2);b;zeros(1,2)];

% solve for Riccati at current state
W = Z{1}; Y = Z{1};
W1 = W(1:9,:); Y1 = Y(10:end,:);
N = Y1*inv(W1);

% compute control command: convert to dimensional form
u = -inv(R)*B'*N*xs;
CLalpha = inv_table_lookup(Mach);
u = u/CLalpha;

% limit the angles from [-1,1] radians in local frame
sat = 1;
```

```
ind = find(u>sat);
ind2 = find(u<-sat);
u(ind) = sat;
u(ind2) = -sat;

end
```

dispersion_vals.m: used to generate uniform dispersion yaw/pitch angles

```
function dispersion = dispersion_vals();


% create desired dispersion of pitch/yaw angles (NOTE: the
actual std dev. (a) and mean (b) may not be exactly a and b for
each chunk, but they will be close) For actual dispersion used
in this thesis, see Appendix E.

rng(0,'twister') % make repeatable
a = .0075;
b = 0;
yaws = a.*randn(100,1) + b;
a = 1.5;
b = 23;
pitches1 = (a.*randn(25,1) + b)*pi/180;
pitches1*180/pi;
clear a b
a = 1.5;
b = 57;
pitches2 = (a.*randn(25,1) + b)*pi/180;
pitches2*180/pi;
clear a b
a = 5;
b = 40;
pitches3 = (a.*randn(50,1) + b)*pi/180;
pitches3*180/pi;
clear a b

pitches = [pitches1;pitches2;pitches3];

rvals = 100*rand(100,1);
[vals inds] = sort(rvals);

dispersion = [yaws,pitches(inds)];
```

hitgrnd.m: terminal stop condition for ode45 at zero altitude

```
function [value,isterminal,direction] = hitgrnd(t,X,xt,dt)

% option add-on for ode45:  tells program to quit once zero
altitude is
% reached

value = X(3);
isterminal = 1;
direction = 1;

end
```

intermediates.m: used to calculate total projectile force and moment

```
function [XYZ,LMN,I] =
intermediates(coeffs,states,rho,c,t,xt,dt,flag)
% output:  force and moment vectors, inertia matrix
% inputs:  aero coeffs, current states, air density, speed of
sound,
%          current time, target, sampling period, flag for
control

% load necessary values
[D,m,g,Ixx,Iyy,Izz,S,Scan,a] = params();

% define the states

x = states(1); y = states(2); z = states(3); psi = states(4);
theta = states(5); phi = states(6); q = states(11);
r = states(12);
u = states(7); v = states(8); w = states(9);
p = states(10);

% inertia matrix
I = [Ixx,0,0;0,Iyy,0;0,0,Izz];

% total velocity
V = sqrt(u^2 + v^2 + w^2);

% assign coeffs to their proper names (note: CZA1 = CYB1 = CNA)
CNPA = 0;  CY0 = 0; CZ0 = 0;
CX0 = coeffs(1); CX2 = coeffs(2); CYB1 = coeffs(3); CZA1 =
coeffs(4);
CDD = coeffs(5); CLP = coeffs(6); CMQ  = coeffs(7); CNR =
coeffs(8); % CNR = CMQ always
RCAX = coeffs(9);
```

```
% calculate force and moment variables needed for states
derivatives
% first get canard contributions from canard function

[Xc,Yc,Zc,Lc,Mc,Nc] =
canard_reactions(states,rho,c,t,coeffs,xt,dt,flag);

XYZ_a = -pi/8*rho*V^2*D^2*[CX0+CX2*(v^2+w^2)/(V^2);CYB1*v/V;...
    CZA1*w/V];

XYZ_w = m*g*[-
sin(theta);sin(phi)*cos(theta);cos(phi)*cos(theta)];

XYZ = XYZ_a + XYZ_w + [Xc;Yc;Zc];

LMN_a = [0,0,0;0,0,-RCAX;0,RCAX,0]*XYZ_a;
LMN_u = pi/8*rho*V^2*D^3*[CDD+p*D*CLP/(2*V);q*D*CMQ/(2*V);...
    r*D*CMQ/(2*V)];

LMN = LMN_a + LMN_u + [Lc;Mc;Nc];

end
```

inv_table_lookpup.m: used to convert canard commands to dimensional form before rotation

```
function [CLalpha] = inv_table_lookup(Mach)

% load the data

%aerodata_c = csvread('canard_aero_props.csv');
global aerodata_c

% coeff order in vector:  CL,CL3,CL5,CD0,CD2,Ci
% use interp1 to get correct coeffs

if Mach >= max(aerodata_c(:,1))
    CLalpha = 4.135;
elseif Mach <= min(aerodata_c(:,1))
    CLalpha = aerodata_c(1,2);
else
   CLalpha = interp1(aerodata_c(:,1),aerodata_c(:,2),Mach);
end


end
```

mach_find.m: used to compute mach number

```
function [Mach,rho,c] = mach_find(V,z)
% output:  mach, density, speed of sound for current state
% input:  total velocity, altitude state (downward value)

% loop to get correct values for rho, speed of sound
zn = -z;
if zn < 35332.00
   rho = 0.0023784722*(1-0.0000068789*zn)^4.258;
   c = 49.0124*sqrt(518.4-0.003566*zn);
else
   rho = 0.00072674385*exp(-0.0000478*(zn-35332.00));
   c = 970.8985166;
end

% assign output

Mach = V/c;

end
```

params.m: used to store parameters needed for simulations

```
function [D,m,g,Ixx,Iyy,Izz,S,Scan,a] = params();

% no inputs:  just stores constants needed for simulation

D = .223;
m = 23/32.2;
g = 32.2;
Ixx = .005;
Iyy = 1.4;
Izz = Iyy;
S = pi/4*D^2;
Scan = 0.00433516492;
a = -g;

end
```

state_grab.m: computes state derivatives at next time step

```
function dx = state_grab(t,X,xt,dt)

[D,m,g,Ixx,Iyy,Izz,S,Scan,a] = params(); dx = zeros(1,12);
```

```
global tlast
diff = t-tlast;
if diff >= dt
    tlast = t;
    flag = 1;
else
    flag = 0;
end

% define each state variable from input vector

x = X(1); y = X(2); z = X(3); psi = X(4); theta = X(5); phi =
X(6);
u_til = X(7); v_til = X(8); w_til = X(9); p_til = X(10); q_til =
X(11);
r_til = X(12);

states = X;

% calculate total velocity

V = sqrt(u_til^2 + v_til^2 + w_til^2);

% call function to calculate mach number and air density

[Mach,rho,c] = mach_find(V,z);

% call function to determine aero coeffs; assign the coeffs
% coeff order in vector:  CX0,CX2,CYB1,CZA1,CDD,CLP,CMQ,CNR,RCAX

coeffs = aero_props_find(Mach);

CNPA = 0;  CY0 = 0; CZ0 = 0;
CX0 = coeffs(1); CX2 = coeffs(2); CDD = coeffs(5); CLP =
coeffs(6); CMQ  = coeffs(7); RCAX = coeffs(9);

% call intermediate function to calculate necessary quantities

[XYZ,LMN,I] = intermediates(coeffs,states,rho,c,t,xt,dt,flag);

% matrices needed to calculate state derivatives

cth = cos(theta); cps = cos(psi); cph = cos(phi);
sth = sin(theta); sps = sin(psi); sph = sin(phi); tth =
tan(theta);

A1 = [cth*cps,sth*sph*cps-cph*sps,cph*sth*cps+sph*sps;...
```

```
    cth*sps,sph*sth*sps+cph*cps,cph*sth*sps-sph*cps;...
    -sth,sph*cth,cph*cth];
A2 = [0,sph/cth,cph/cth;0,cph,-sph;1,sph*tth,cph*tth];
A3 = [0,-r_til,q_til;r_til,0,-p_til;-q_til,p_til,0];

dx(1:3) = A1*[u_til;v_til;w_til];
dx(4:6) = A2*[p_til;q_til;r_til];
dx(7:9) = 1/m*XYZ - A3*[u_til;v_til;w_til];
dx(10:12) = inv(I)*(LMN - A3*I*[p_til;q_til;r_til]);

% assign the output

dx = dx';

end
```

top_level.m: runs ode45 integration

```
clear
clc
close all

warning('off','all');
dispersion = dispersion_vals();

fprintf('Pitch      Yaw      Downrange     Crossrange     Altitude
DR Miss\n')
fprintf('-----      ---      ---------     ----------     --------
-------\n')

% define IC vector

y = 1;
x0 = [0 0 0 dispersion(y,1) dispersion(y,2) 0 2177.7 0 0 -58.928
0 -0.58031E-01]';
xt = 29000;
tf = 100;
dt = 0.025;
t = [0:0.003:tf];

global aerodata_l
global aerodata_nl
global aerodata_c
global count2
count2 = 0;
global M
global N2

aerodata_l = load('aeroprops.csv');
aerodata_nl = load('aeroprops_nonlin.csv');
aerodata_c = load('canard_aero_props.csv');

global tlast
global zetanr
tlast = 0;
tlast = 0;

% run the integration: call state_grab via ode45

options = odeset('events',@hitgrnd);
[time,x] = ode45(@state_grab,t,x0,options,xt,dt);

Downrange = x(:,1);
Crossrange = x(:,2);
Altitude = x(:,3);
YawAngle = x(:,4);
PitchAngle = x(:,5);
RollAngle = x(:,6);
u = x(:,7);
v = x(:,8);
w = x(:,9);
p = x(:,10);
q = x(:,11);
r = x(:,12);

d = Downrange(end);
c = Crossrange(end);
a = Altitude(end);

mval = M;
nval = N2;

fprintf(' %5.2f  %5.2f      %5.1f      %5.5f     %5.5f
%5.5f\n',x0(5)*180/pi,x0(4)*180/pi,d,c,a,xt-d)

clearvars -except d c a dispersion x0 xt mval nval
clearvars -GLOBAL
```

vac_pseudo.m: computes vacuum model

```
function [M,N] = vac_pseudo(x,z,xt)
```

```
% input:  current downrange and altitude, desired target
% output:  coeffs needed for vac. model

abc = [x,x^2;xt,xt^2]\[-z;0];

M = abc(1); N = abc(2);
end
```

NOTE:

The codes in Appendix A provide all of the information necessary to run a single trajectory, save for the presence of the .csv files for the aerodynamic coefficient interpolation. The top level code can be manipulated to run several trajectories in a loop, saving the needed information each time. There were many other MATLAB files used in the data mining process needed to create the plots in Chapter 5. These codes are not presented here.

Appendix B: Hydra-70 Mach-Dependent Properties

mach	**slcop**	**cx0**	**cna**	**clp**	**cdd**	**cmq**	**cx2**
0.01	1.46	0.7	8.19	-11.2	-0.12	-2120	6.16
0.6	1.46	0.7	8.19	-12.2	-0.12	-2120	6.16
0.8	1.4	0.72	8.48	-12.6	-0.12	-2920	6.64
0.9	1.35	0.81	8.94	-12.8	-0.12	-2920	6.96
0.95	1.32	0.89	9.02	-13.3	-0.11	-2876	7.36
1	1.29	0.96	9.1	-13.8	-0.1	-2830	8.64
1.05	1.27	0.98	9.18	-14.7	-0.09	-2816	9.84
1.1	1.24	1	9.26	-15.6	-0.08	-2800	11.21
1.2	1.2	1.01	9.19	-16.16	-0.05	-2642	13.42
1.35	1.23	1	8.71	-16.26	-0.02	-2368	12.87
1.5	1.31	0.99	8.29	-16.1	-0.03	-2222	12.29
1.75	1.38	0.93	8.01	-15.6	-0.04	-2040	11.68
2	1.42	0.86	7.79	-14.8	-0.05	-1898	11.17
2.25	1.44	0.8	7.64	-14.1	-0.06	-1805	10.93
2.5	1.46	0.75	7.51	-13.4	-0.06	-1712	10.67
2.75	1.46	0.71	7.36	-12.7	-0.06	-1656	10
3	1.47	0.68	7.22	-12	-0.05	-1600	9.3
5	1.47	0.62	6	-11	-0.05	-1300	6.2

Appendix C: Canard Mach-Dependent Properties and Dynamic Properties

Mach	CL alpha (rad^{-1})	CL alpha3	CL alpha5	CDO	CD2	Ci
0.4	2.576701	0	0	0.05726	0	0
0.6	2.863001	0	0	0.054079	0	0
0.7	3.212923	0	0	0.063622	0	0
0.75	3.40379	0	0	0.066803	0	0
0.8	3.531034	0	0	0.069984	0	0
0.85	3.69009	0	0	0.076347	0	0
0.875	3.753712	0	0	0.076347	0	0
0.9	3.849145	0	0	0.082709	0	0
0.925	3.912767	0	0	0.08589	0	0
0.95	3.97639	0	0	0.08589	0	0
0.975	4.071823	0	0	0.089071	0	0
1	4.135445	0	0	0.089071	0	0

Appendix D: Canard Angle Properties

Canard #	SL (ft)	BL(ft)	WL (ft)	γ (rad)	ϕ (rad)	δ (rad)	S (ft^2)
1	3.8	0.14167	0	0	0	variable	0.004335
2	3.8	-0.14167	0	0	Π	variable	0.004335
3	3.8	0	0.14167	0	$\pi/2$	variable	0.004335
4	3.8	0	-0.14167	0	$-\pi/2$	variable	0.004335

Appendix E: Initial Yaw and Pitch Angle Random Sets

Trial	Yaw Angle (°)	Pitch angle (°)	Trial	Yaw Angle (°)	Pitch angle (°)
1	0.23105	39.024	26	0.44463	43.301
2	0.78805	57.034	27	0.31236	56.591
3	-0.97067	40.918	28	-0.13039	34.399
4	0.37049	46.757	29	0.12628	58.052
5	0.13698	43.126	30	-0.33831	57.525
6	-0.56194	42.335	31	0.38176	57.762
7	-0.18632	58.691	32	-0.49292	25.568
8	0.14723	35.76	33	-0.45931	20.059
9	1.5377	35.673	34	-0.34786	57.423
10	1.1901	22.297	35	-1.2652	41.957
11	-0.58007	54.634	36	0.6181	52.63
12	1.3042	41.538	37	0.13974	21.668
13	0.31172	38.484	38	-0.32441	39.334
14	-0.027096	24.441	39	0.58884	45.093
15	0.30714	46.303	40	-0.73547	58.648
16	-0.088078	27.362	41	-0.043935	57.05
17	-0.053347	25.032	42	-0.10375	21.188
18	0.64015	39.349	43	0.13717	25.155
19	0.60549	24.109	44	0.13444	21.413
20	0.60899	55.765	45	-0.37165	55.265
21	0.28855	21.392	46	-0.012914	22.709
22	-0.51888	23.186	47	-0.070852	47.635
23	0.30821	19.792	48	0.26974	33.192
24	0.70054	42.6	49	0.4698	56.572
25	0.21009	55.531	50	0.47668	39.9

Trial	Yaw Angle (°)	Pitch angle (°)	Trial	Yaw Angle (°)	Pitch angle (°)
51	-0.37113	38.912	76	-0.60258	28.351
52	0.033243	43.957	77	-0.61122	33.714
53	-0.52173	44.31	78	0.20979	54.375
54	-0.47849	44.13	79	-0.076221	37.619
55	-0.0029433	54.999	80	-0.084248	33.34
56	0.6586	58.446	81	0.6099	38.531
57	-0.33074	24.261	82	0.1253	56.551
58	0.15959	53.996	83	0.085003	45.195
59	-0.096938	23.455	84	0.68226	21.741
60	0.48015	34.412	85	-0.34569	42.764
61	-0.46799	55.753	86	0.29935	22.183
62	0.013991	23.735	87	0.35885	38.876
63	0.23743	56.583	88	-0.10473	39.661
64	0.47295	53.922	89	0.092677	41.668
65	0.66358	35.756	90	-0.50098	32.755
66	0.036926	38.326	91	-0.4933	42.275
67	-0.64096	22.1	92	0.045067	24.238
68	-0.31898	39.826	93	0.31037	37.055
69	-0.45618	39.117	94	1.111	22.703
70	1.01	36.009	95	-0.28658	56.607
71	-0.26454	48.277	96	0.0805	23.15
72	0.32146	38.951	97	-0.035449	56.2
73	-0.082686	42.258	98	-0.83066	40.115
74	0.38185	56.469	99	-0.18863	25.068
75	-0.32867	40.256	100	-0.77121	36.427

Trial	Yaw Angle (°)	Pitch angle (°)
101	0.31172	30.478
102	-0.027096	50.479
103	0.30714	50.469
104	-0.088078	31.293
105	-0.053347	34.154
106	0.64015	51.64
107	0.60549	49.753
108	0.60899	34.552
109	0.28855	48.703
110	-0.51888	30.807
111	0.30821	30.514
112	0.70054	28.038
113	0.21009	48.868
114	0.44463	49.638
115	0.31236	35.368
116	-0.13039	26.612
117	0.12628	32.751
118	-0.33831	52.055
119	0.38176	27.975
120	-0.49292	50.942
121	-0.45931	49.955
122	-0.34786	47.433
123	-1.2652	51.664
124	0.6181	49.847
125	0.13974	29.35