MODELING AND SIMULATION OF A SOUNDING ROCKET ACTIVE STABILIZATION SYSTEM

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Aerospace Engineering

by

Steven MacLean

June 2017

COMMITTEE MEMBERSHIP

TITLE:	Modeling and Simulation of a Sounding Rocket Active Stabilization System
AUTHOR:	Steven MacLean
DATE SUBMITTED:	June 2017
COMMITTEE CHAIR:	Eric Mehiel, Ph.D. Professor of Aerospace Engineering
COMMITTEE MEMBER:	Amelia Greig, Ph.D. Professor of Aerospace Engineering
COMMITTEE MEMBER:	Jordi Puig-Suari, Ph.D. Professor of Aerospace Engineering
COMMITTEE MEMBER:	Damon Turner Raytheon Missile Systems

ABSTRACT

Modeling and Simulation of a Sounding Rocket Active Stabilization System

Steven MacLean

The Horizon Simulation Framework is a modeling and simulation framework developed to verify system level requirements. In this thesis, the framework is extended to include the Dynamic position type that existed in the early development phase of the framework. The Dynamic position type is tested through the modeling and simulation of a sounding rocket. An active control system based on linear-quadratic regulator (LQR) control theory is implemented and tested in the simulation to determine the overall effect on altitude. A first order aerodynamics and aeroprediction model are created within the framework to allow for rapid changes early in the design process of the sounding rocket. The flight dynamics are compared to two different sounding rocket flights and the aeroprediction model is validated against public wind tunnel test data.

ACKNOWLEDGMENTS

I would like to thank my family for their continued support throughout my life. And thanks you to the wonderful Allison for editing this even if you didn't understand what was going on and for supporting me through this whole process.

Thanks you to Dr. Mehiel for opening me to the world of modeling and simulation and guiding me through the process and helping me along the way. Thank you to Dr. Grieg for taking the time to advise CPSS and sit on my committee. I want to thank Damon Turner for his invaluable direction in this project. It would have been much more difficult without the resources you provided for me. Thank you Dr. Puig-Suari for taking the time out of your very busy schedule to serve on my committee.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Sounding rockets are relatively inexpensive suborbital rockets frequently used for upper atmosphere experiments and astronomical observations. The goal of these rockets is to maximize the scientific data collected during the flight while minimizing cost. To maximize the scientific value of the mission, the sounding rocket design will typically attempt to maximize the loft time, or altitude, of the flight. To further decrease cost, engineers rely on modeling and simulation to test and predict the performance of systems. The Horizon Simulation Framework (HSF), is a high-level system modeling and verification tool.

In this thesis, HSF is extended to model sounding rockets to demonstrate its scriptable subsystem capability and its dynamic trajectory simulation. Additionally, a sounding rocket is modeled with and without a control system to compare the performance results and to demonstrate a low-cost way to improve the altitude of a sounding rocket.

1.1 Sounding Rockets

Commercial sounding rockets travel higher than what is reachable by balloons and are much cheaper than satellites in launch cost. They reach as high as 1500 km above the surface of the Earth and have flight times on the order of 10-20 minutes. In the US, the boosters are typically surplus missile solid motors, decreasing the cost even more.

A typical launch sequence includes the boost phase, where the engine, or engines, fire to gain altitude. Large sounding rockets can include as many as four motor

stages[1]. Throughout the boost phase, the rockets are often spun up about the thrust axis to stabilize the system through the flight. Once the boost phase is complete, and the rocket has cleared the bulk of the atmosphere, the rocket executes a de-spin procedure and deploys the payload. The payload will continue to coast through the upper atmosphere until apogee where it will begin to fall back towards Earth. Depending on the exact mission and its profile, the payload will have some additional time for observation or measurement until the reentry phase begins. Some sounding rockets have small heat shields for this section of flight. After the sounding rocket payload section has reentered the atmosphere and slowed to an acceptable speed, a drogue parachute is deployed to slow the descent rate by increasing the area as well as provide stability to the system through descent, so it does not experience excessive rotation rates. Once much closer to the ground, the main parachute deploys, further slowing down the rocket so that it impacts the ground or ocean at a slow speed [2].

Sounding rockets are also frequently used by college rocketry programs as they are within the program's budget and allow students to perform science experiments in the upper atmosphere and low-g environments. Typically, the college-designed and built rockets do not leave the lower atmosphere, but some universities have developed rockets that exceed 50 km [3].

As alluded to before, sounding rockets are usually on an open loop control system by either spin stabilizing or designing a statically stable rocket. The open loop control dynamics mean that the rocket will tend to be at or near 0 angle of attack throughout the lower atmosphere flight. One way to increase the altitude is to control the system so that all of the thrust provided by the motor is used solely to counteract gravity. In the open loop system, this does not occur because of the tendency to 0 angle of attack. By adding a low-cost active stabilization system, the sounding rocket can be controlled to maintain the thrust axis pointing away from Earth throughout the flight, maximizing altitude.

1.1.1 Kronheim Rocket

The Kronheim rocket is a sounding rocket currently being developed at Cal Poly San Luis Obispo to carry a 4 kg payload to an altitude of 9 km as a stepping stone to larger and more powerful rockets. The rocket is a single stage rocket propelled by a student-designed and built hybrid motor. The motor series has been in development since 2008 and is currently on its 7th major revision. To increase the altitude of the rocket, a more advanced avionics system is required to accurately track the rocket during flight and determine where the rocket lands to aid recovery attempts. To build up this avionics system, HSF was used to develop a measurement subsystem and a state estimation subsystem so that the inertial position of the rocket can be tracked throughout the flight.

In addition to increasing motor power, a second way to increase altitude is to make sure that all thrust goes into generating vertical velocity. This can be accomplished by controlling the rocket to maintain an attitude normal to the Earth. An average angle with respect to the vertical flight path of 8 degrees will result in 99% of the maximum altitude reached. HSF was used to design a control subsystem based on optimal control theory to maximize the altitude of the rocket.

1.1.2 Related Work

Minh developed a real-time state estimator for a sounding rocket to control events that occur on the rocket, such as stage separation and parachute deployment [4]. The simulation was developed using preexisting sounding rocket simulator and was used to verify the performance of the on-board state estimator.

An open source model rocket program was developed by Niskanen to simulate and predict the performance of amateur rockets [5]. It has the capability to do a broad

range of geometries and propulsion types. It includes a GUI to place components. The simulator estimates the aerodynamics and does a 5DOF simulation of the trajectory, as it neglects forces and rotation in the yaw axis.

Chowdbury developed a sounding rocket simulator with a focus on structural design. It is a full 6 DOF simulator [6].The program can be used to assist in the coupled design and optimization of subsystems, as the simulator uses a genetic algorithm as a design optimization technique to allow the user to improve the design of the sounding rocket and optimize each of the modeled components.

1.2 Horizon Simulation Framework

Horizon Simulation Framework (HSF) is a framework that was developed in the mid-2000's at Cal Poly San Luis Obispo [7]. It was originally written in C++ with scripting support provided by Lua. The framework had undergone multiple iterations before version 3 of the framework was released in 2016 [8]. Version 3 of the framework was rewritten in C# and switched the scripting support to Python instead of Lua, as python had become a more popular language in the ten years since the initial release of the framework. The overall functionality remained the same, but the rewrite allowed for the underlying structure to be updated and improved.

HSF is designed around model-based systems engineering. It was built to perform requirement verification and validation throughout the design process but also built to be modular and flexible enough to be used in a variety of different modeling and simulation tasks. So while the design of the framework favors the simulation of a satellite mission, HSF can be extended to simulate thermal gliders [9] and sounding rockets.

1.2.1 Simulation Design and Principles

HSF is built around two main components: a scheduler and the system model. In most cases, the scheduler portion of code does not need any interaction from the system designer because the interactions between the scheduler and the subsystems are well-defined.

Scheduler

The scheduler runs an exhaustive search over the solution space to determine tasks that can be performed. As the scheduler tries to schedule tasks, it checks with each subsystem whether it can or cannot perform the task requested. After each scheduler time step, all of the potential schedules are evaluated based on a user-defined scoring method. As it is computationally prohibitive to keep all possible schedules, the scheduler will periodically cut the number of schedules to keep based on the lowest scoring schedules.

System Model

The system model is made up of the environment and the individual subsystems. Each subsystem in HSF has a CanPerform method that is the interface between the scheduler and each of the subsystems. As the scheduler tries to schedule tasks, it will ask each of the subsystems if they can perform the requested task. The bulk of the subsystem functionality is computed in this CanPerform method. Also contained within the subsystem models are dependency functions. These dependency functions are how subsystems pass information to each other and sets the evaluation order of the system for the scheduler. For example, a power subsystem may be dependent on a communications subsystem so that the communications subsystem would be

evaluated first and the power used to complete the task fed to the power subsystem through the dependency function. The power system would use this in its calculation to determine if it can provide enough power to perform the task.

1.2.2 Example Case

With each iteration and major release, HSF is tested against an example case, Aeolus. To further illustrate HSF, this test case will be explained briefly. Aeolus is an Earth-imaging satellite constellation made up of 2 satellites with a collection of targets on the ground for the satellites to image and communication ground stations to downlink this data. There are also simple models of power, attitude determination and control (ADC), an electro-optical (EO) sensor, communications, and data storage.

The simulation is of one day or day-in-the-life (DITL) of the constellation and attempts to maximize the score of all of the targets imaged on the ground. The scheduler performs a basic check of what targets could be within line of sight of the satellite and then asks whether each subsystem can complete the necessary tasks to meet that schedule. For example, the ADCS system will check if it can slew to the target fast enough and the data storage system will check that there is enough onboard storage to store the captured image. The designer can see what subsystems cause the task to fail and can adjust the design of the spacecraft as necessary to increase performance.

1.2.3 Dynamic State Type

In early versions of the framework, a dynamic-position dynamic state type existed [9], but in the rewrite and subsequent version 3.0 release of the framework [8], this state type was not included–only static and predetermined position types were. A static type does not change its position throughout the simulation and is typically

used for ground stations or targets on the surface of the Earth. The predetermined state is for objects where the state is not affected by any subsystems and can be completely predetermined before simulation. An example of this is a satellite in orbit around Earth. As the flight dynamics of a sounding rocket are neither static nor predetermined, the dynamic-position dynamic state type needed to be re-implemented. In addition to each of the types, the position can be represented in Earth Centered Inertial (ECI), and latitude, longitude, altitude (LLA) coordinates creating a total of six different dynamic state types within HSF.

1.2.4 Sounding Rockets in HSF

The sounding rocket model is developed here by using the dynamic-position dynamic state type. The sounding rocket dynamics are changed by the input of physical model data such as mass and the thrust can either be defined as a constant or by a time-based input file. All of these values determine how the equations of motion are affected. Also in the model are the sensor, state estimation, controller, and recovery subsystems. There are 3 different types of tasks that can be performed by the sounding rocket: control, drogue parachute deployment, and main parachute deployment. The parachute tasks can only be scheduled once, but have a very high schedule value in order to force the scheduler to schedule the task when available. The control task has a low schedule value but can be performed many times, allowing the generated schedule to always have a control task able to perform. All of the gain matrices are loaded through the XML input. The construction of the sounding rocket model in this fashion adheres to the design principles of modularity and flexibility in HSF because they models can be used for multiple sounding rocket vehicles and potentially other air-based systems. This generality to the models is important to the continued development of HSF and specific contributions of this thesis.

1.3 Systems Engineering

According to the NASA systems engineering handbook, systems engineering is an approach to the "design, creation, and operation of systems" that minimizes cost with respect to "performance, cost, schedule, and risk" [10]. The process to achieve this can be broken down into five distinct phases, each designated by a letter: preliminary analysis (A), definition (B), design (C), development (D), and operations (E). Phase A is the feasibility phase of the process, and this is where top-level requirements are created, and their evaluation criteria are determined. Phase B is where initial design occurs. This design is the baseline and can complete the mission determined in phase A. The system level requirements are set at this level. Phase C is where each system and their associated subsystems are designed and put through preliminary testing. The design may go through many iterations during this phase, but by the end, the complete "built-to" documentation is created. Phase D is where the construction of the design happens. The design is built, tested, and verified against the top level requirements. Phase E is the actual operation of the system where it performs the tasks set out in phase A.

In phase A of the mission, the CONcept of OPeration (CONOP) is defined. This is a critical phase in the systems engineering process as an error in the CONOPs definition will flow down through the system resulting in a mission that could fail to meet requirements. A common approach to validating the CONOPs against the requirements in aerospace engineering is to model and simulate the system before building test articles or the final system. This approach is used because of the high cost of many of the aerospace systems. Loper describes simulations as "an attempt to model a real-life or hypothetical situation [...] so that it can be studied to see how the system works" [11]. Furthermore, Loper explains that simulations predict the behavior of the system due to changing variables.

1.3.1 Model-Based Systems Engineering

In a typical systems engineering approach, the requirements set forth at the top level flow down to the subsystem level at each of the subsequent phases. The requirements of each piece are then validated at the low levels first, increasing in scope until finally, the system level requirements are determined to be met.

Model-Based Systems Engineering (MBSE) is the process of "validating models in the engineering process to a central and governing role in the specification, design, integration, validation, and operation of a system" [12]. In MBSE, the models developed are used throughout the life cycle of a project to check the system against the top level requirements. For example, instead of using testing to validate a design against the requirements flowed down from the CONOPS, the test is used to validate the model, and the model is used to verify the system against requirements. This improves the workflow as there is minimal need to generate the documentation for each subsystem because the model itself serves as the documentation. This eliminates errors in the creation of the specification and eliminates having to regenerate and update all the documentation when a system changes.

1.4 Organization of Thesis

The goal of this thesis is to extend the functionality and demonstrate the flexibility of the Horizon Simulation Framework and develop the sounding rocket model to predict the performance of the design. A state estimator based on the extended Kalman filter and a controller based on optimal control principles are developed and tested within the simulation framework.

Chapter 2 gives an overview of the different models created in this thesis. This includes the environmental models, aerodynamic prediction model, and the equations

of motion (EOMs). The Kalman filter and LQR controller are introduced in this chapter.

Chapter 3 discusses the implementation of the models discussed in Chapter 2 within HSF. The extended Kalman filter is implemented on an example sounding rocket, and its performance is compared to truth results output from the simulation. The full control loop is defined, and the design process of the controller is discussed. Finally, the implemented subsystems and their interactions within HSF are discussed.

In chapter 4, the simulation output is compared against real-world flight data. Data from two separate flights are used. The models are simulated using the HSF aeroprediction methods for aerodynamics. Additionally, the aeroprediction methods used in HSF are compared against publicly available wind tunnel data.

Chapter 5 introduces the Kronheim sounding rocket and implements a controller on it. It compares the performance of the uncontrolled and controlled flights.

Chapter 2

SOUNDING ROCKET MODEL

This chapter covers the various models created to run the simulation. The atmosphere and magnetic models are first presented to define the operating environment of the sounding rocket. The coordinate frame definitions are the presented before entering a detailed discussion and presentation of the aeroprediction model implemented within HSF. The general equations for both the Kalman filter and LQR controller are presented and the sensor models implemented are discussed.

2.1 Environment Models

This section presents the different environmental model constructed for this thesis including the atmosphere and magnetic field.

2.1.1 Atmosphere

U.S. Standard Atmosphere

The standard atmosphere model was first released in 1958. It is a model of temperature, pressure, density, and viscosity versus altitude in the atmosphere. The model is a standard version typically used in low fidelity simulations or early in the design process. It does not account for any seasonal or time of day effects that exist in more advanced models. The version used in HSF is the 1976 Standard Atmosphere [13].

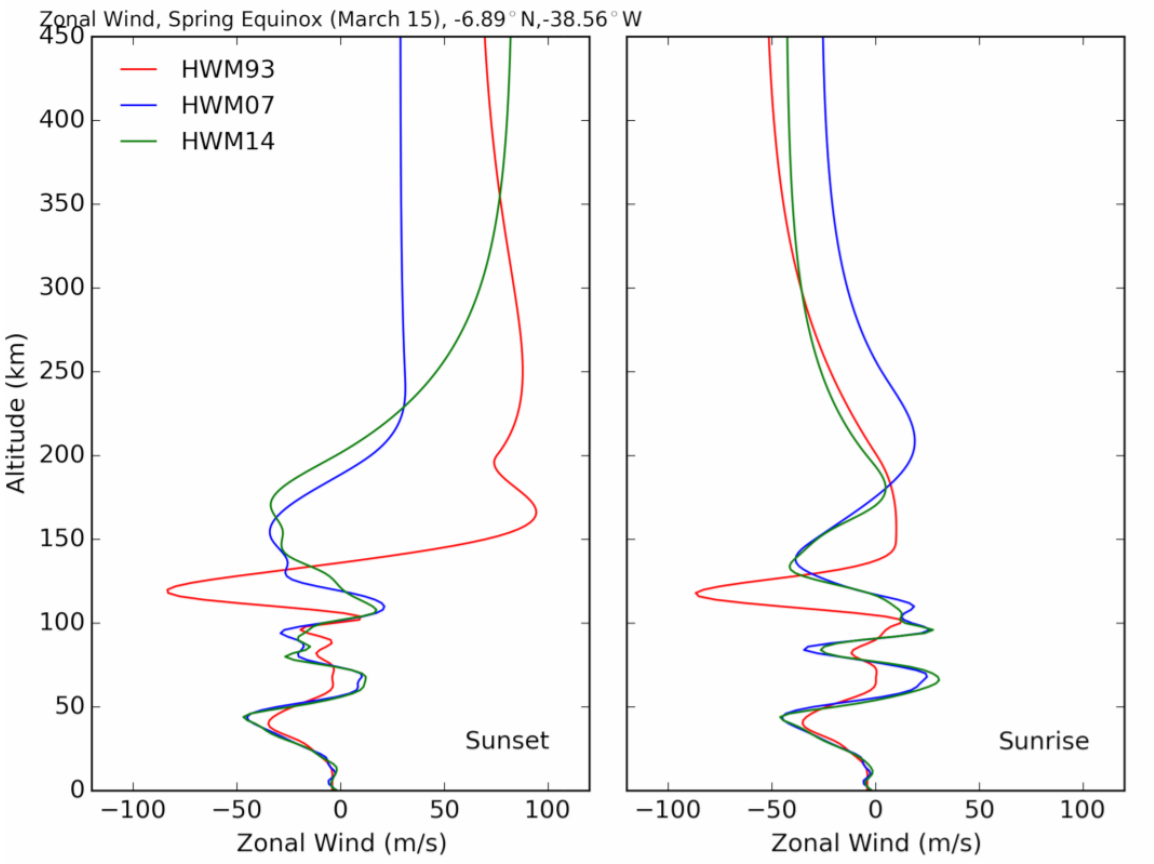

Figure 2.1: Horizontal Wind Model Sample Output [14]

Standard Wind Model

The standard atmosphere model above does not include any wind in its formulation. To more accurately estimate the trajectory and test controller capability, a standard wind model is implemented. The model used is the 2014 Horizontal Wind Model distributed by the Naval Research Laboratory [15]. It is is an empirical model based on measurements taken around the world and accounts for latitude, longitude, altitude, day of year, and time of day. The HWM provides measurements of zonal and meridional winds from 0 to 500 km.

Near Real Time Model

The near real-time model is based on outputs from the Global Forecasting System (GFS)[16]. The GFS is a worldwide forecasting model run by the National Weather Service (NWS). The model runs every 6 hours and the data from each of these runs is available online almost immediately. Model data dates back to 2004 and is available for download. The model predicts weather conditions 10.5 days in advance, but as the forecast time gets further from the model run time, the data becomes less reliable. It is advantageous to select the nearest 6-hour forecast data set to get the most accurate estimation of the atmospheric conditions.

The model outputs data into bins that are gridded in 0.5-degree increments around the globe, which correspond to approximately 28km x 28km areas. The model predicts temperature, pressure, wind velocity, and other atmospheric properties at different levels defined by pressure in the atmosphere and range from 1000 mbar to 1 mbar levels which typically corresponds to 50 km in altitude. Also, the model predicts of surface level temperature and wind speeds. HSF interprets the data file and stores the atmospheric properties at each altitude available, and then uses a linear interpolation scheme for values that are not provided by the model.

2.1.2 Magnetic Field

World Magnetic Model

The World Magnetic Model (WMM) is a model jointly developed by the US and UK governments. It is released every five years due to frequent random changes to the Earth's magnetic field that are impossible to model. The WMM takes the current latitude, longitude, altitude, and date to produce the magnetic field vector. The version included in HSF is the 2015 version, set to expire in 2020 [17].

2.2 Coordinate Frames

In HSF, all calculations are executed in the inertial frame. However, many of the dynamics and measurements of the sounding rocket occur in other frames which are described below.

Inertial

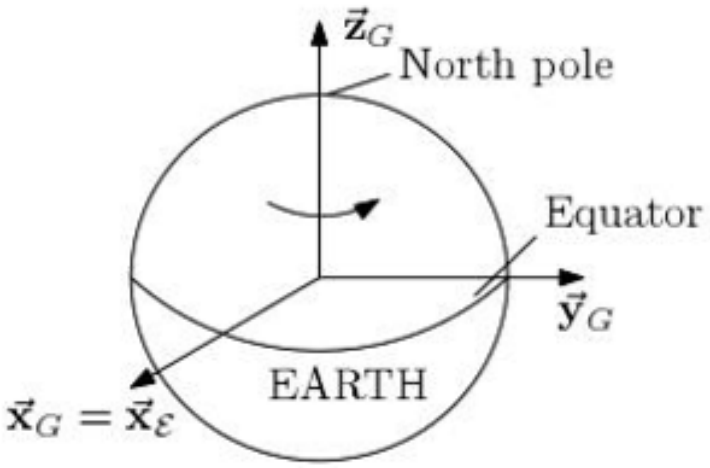

Figure 2.2: ECI Frame[18]

The Earth-centered inertial (ECI) frame is centered at the Earth with the z-axis through the spin axis and the x and y axes through the equator. This frame is considered inertial in HSF and is the frame in which all the equations of motion are valid.

Wind

The wind frame is defined by the relative velocity of the rocket with respect to the atmosphere. It is fixed inside the rocket with its origin at the body. This frame

is used to calculate the aerodynamic forces and moments acting on the body of the rocket.

Body

The body frame of the rocket is where most of the dynamics act and where measurements are made. The x-axis is through the nose of the rocket and the y and z axes line up with the measurements sensors. The rocket is assumed to be symmetric in this context and thus the dynamics are the same in both the y and z directions.

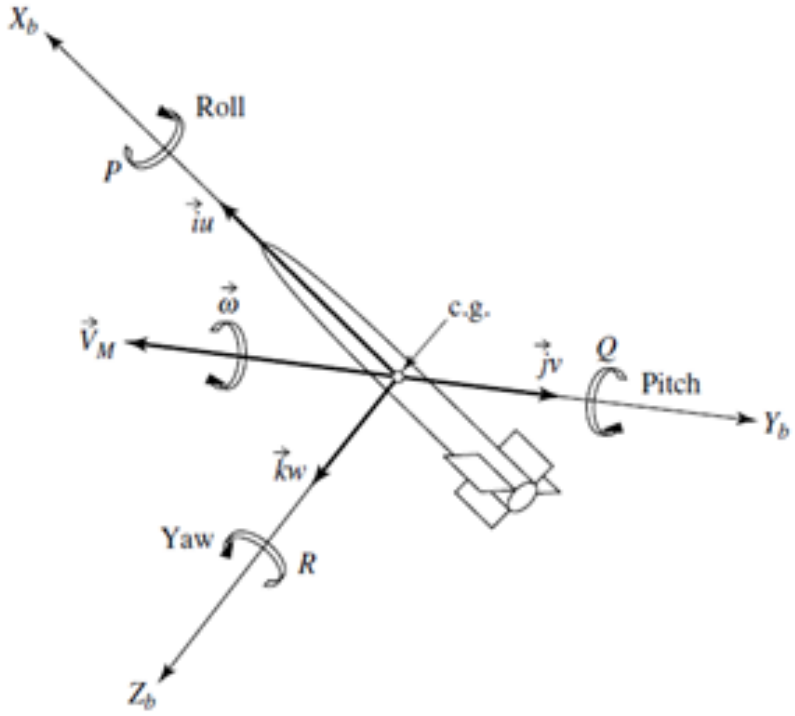

Figure 2.3: Body Frame Definition [19]

2.3 Aerodynamics

In this framework, there are two ways to model the aerodynamics of the system: through externally provided coefficients, or through a first order aeroprediction code

included within HSF for sounding rockets.

2.3.1 External Coefficients

The aerodynamics can be input into HSF as a text data file. This file consists of a 2-dimensional table, where each column contains the value at an angle of attack, and each row contains a value at a different Mach number. The first row and first column correspond to the breakpoints of the angle of attack and Mach number, respectively. A linear interpolation is used to obtain values in between each of these breakpoints. The coefficients that can be included in the simulation are the three axis force coefficients, the three axis moment coefficients, and the three axis moment damping coefficients.

2.4 Aeroprediction

The prediction model built into HSF is a semi-empirical model that follows the formulation in Niskanen [5] and Auld [20]. These models assume small angles of attack in their formulation.

2.4.1 Friction Drag

Friction drag is the dominant contributor to drag at subsonic speeds. The following equations are used to calculate the friction coefficient separately for each of the body parts. These common equations are multiplied by a modifier dependent on the type of object (body tube, fins, etc.).

$$Rn^* = \frac{aML}{12\nu}(1+0.0283M-0.043M^2+0.2107M^3-0.03829M^4+0.002709M^5) \quad (2.1)$$

$$Cf^* = 0.037036Rn^{*-0.155079} \quad (2.2)$$

$$Cf = Cf^*(1+0.00798M-0.01813M^2+0.0632M^3-0.00933M^4+0.000549M^5) \quad (2.3)$$

L is the characteristic length of the object in question. For the body, L is taken as the length of the entire body including nosecone; for the fins, L is taken as the root chord length; for protuberances, L is taken as the length of the protuberance.

$$Cf^*_{rough} = \frac{1}{[1.89 + 1.62\log_{10} L/K]^{2.5}} \tag{2.4}$$

$$Cf_{rough} = \frac{Cf^*_{rough}}{1 + 0.2044M^2} \tag{2.5}$$

$$Cf_{final} = max(Cf, Cf_{rough}) \tag{2.6}$$

Body Friction Drag

The body friction drag is the skin friction drag on both the nose cone and the body cylinder. In this prediction, it is assumed that the body is a constant diameter.

$$C_{d_f} = Cf_{final}[1 + \frac{60}{(L/d)^3} + 0.0025(L/d)]\frac{4S_B}{\pi d^2} \tag{2.7}$$

Fin Friction Drag

The fin friction drag is calculated based on the number of fins and the fin shape. In the build-up model, it is assumed that the fins have an airfoil shape with chord location of X_c. In the case of a flat plate fin, the chord location is assumed to be 0.

$$Rn = \frac{aMC_r}{12\nu} \tag{2.8}$$

$$\lambda = \frac{C_t}{C_r} \tag{2.9}$$

$$Cf_\lambda = Cf_{final}\frac{[\log_{10} Rn]^{2.6}}{\lambda^2 - 1}\left(\frac{\lambda}{[\log_{10} Rn\lambda]^{2.6}} - \frac{1}{[\log_{10} Rn]^{2.6}}\right) + 0.0564\left[\frac{\lambda^2}{[\log_{10} Rn\lambda]^{3.6}} - \frac{1}{[\log_{10} Rn]^{3.6}}\right]\Bigg) \tag{2.10}$$

$$C_{d_{fins}} = Cf_\lambda[1 + 60\left(\frac{t}{C_r}\right)^4 + 0.8(1 + 5X_c^2)\left(\frac{t}{C_r}\right)\frac{4NS_f}{\pi d^2} \tag{2.11}$$

Protuberance Drag

Protuberances on a sounding rocket are objects that extend into the flow around the rocket. Examples include launch rail hardware or camera bumps. It is assumed that these items are much smaller than length and diameter of the rocket.

$$Cf_{pro} = 0.8151 Cf_{final} (\frac{a}{L_p})^{-0.1243} \tag{2.12}$$

$$C_{d_{pro}} = Cf_{pro} \left[1 + 1.798 \left(\frac{\sqrt{A}}{L_p} \right)^{3/2} \right] \frac{4S_{pro}}{\pi d^2} \tag{2.13}$$

Excrescencies Drag

Excrescences are imperfections in the rocket mold line, such as scratches or divets. It is assumed these are randomly distributed about the entire rocket. Excrescencies drag is a smaller effect that can be neglected in most cases but is included here for completeness.

$$\begin{aligned} C_{d_e} &= K_e \frac{4S_r}{\pi d^2} \\ K_e &= 0.000038 \end{aligned} \tag{2.14}$$

2.4.2 Base Drag

Base, or pressure, drag occurs due to the difference in pressure between the front of the rocket and the pressure in the separated flow region behind the rocket. Base drag is difficult to predict analytically, as it depends on predicting separation of the flow behind the rocket, and it is affected by whether the motor is thrusting or not. In this model, the no thrust, or coasting, base drag will be used throughout the flight.

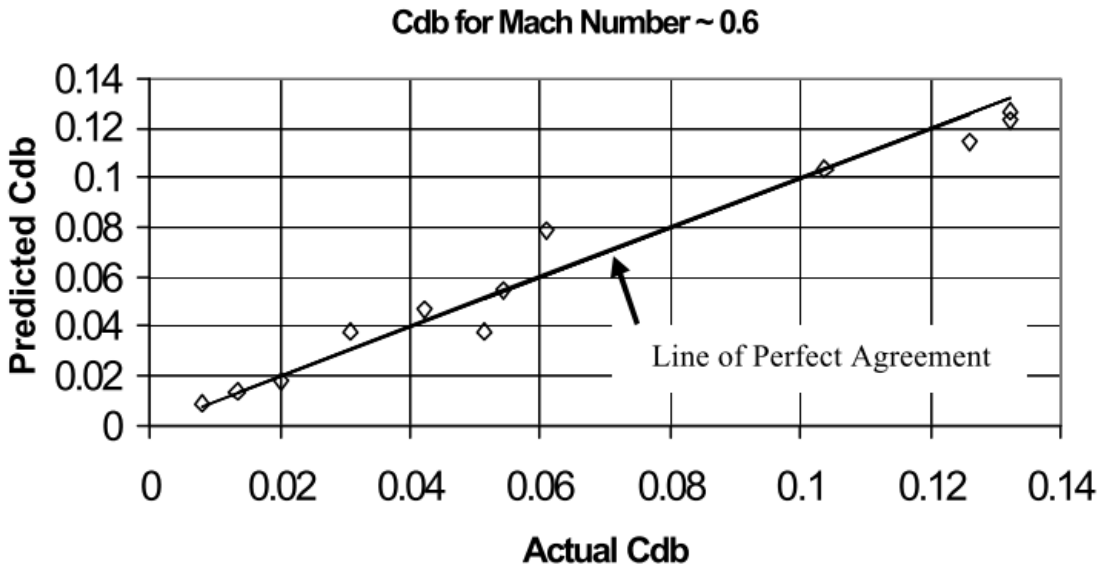

Figure 2.4: Base Drag Coefficient Predicted vs. Actual [20]

For flow below Mach 0.6, the base drag can be approximated by

$$
\begin{aligned}
&C_{d_b} = K_b \frac{\left(\frac{d_b}{d}\right)^n}{\sqrt{C_{d_f}}} \\
&\text{Where,} \\
&K_b = 0.0274 \tan^{-1}\left(\frac{L_0}{d} + 0.0116\right) \\
&n = 3.6542\left(\frac{L_0}{d}\right)^{-0.2733}
\end{aligned}
\tag{2.15}
$$

For calculating the base drag above Mach 0.6, the base drag at Mach 0.6 is calculated and then modified based on a piecewise function. This piecewise function is generated from [21].

$$
\begin{aligned}
&f_b = 1.0 + 215.8(M - 0.6)^6, \text{ for } 0.6 < M \leq 1 \\
&f_b = 2.0881(M-1)^3 - 3.7938(M-1)^2 + 1.4618(M-1) + 1.883917, \text{ for } 1 < M \leq 2 \\
&f_b = 0.297(M-2)^3 - 0.7937(M-1)^2 - 0.1115(M-1) + 1.64006, \text{ for } M > 2
\end{aligned}
\tag{2.16}
$$

2.4.3 Wave Drag

Wave drag is the drag term that results from transonic and supersonic shock waves. The following equations are curve fit to data sets. The curve fit imposes a restriction on the nosecone to body length ratio of the rocket. In this case, the nosecone length can be no more the 0.6 of the body length. Fortunately, for the majority of sounding rockets, this is the case.

Transonic Wave Drag

To first estimate the wave drag coefficient in the transonic region, the transonic region of flight must be defined. Transonic flight occurs when there is both subsonic and supersonic flow around the body of the air vehicle and typically occurs from Mach numbers of 0.8-1.2. The initial Mach number where the rocket enters the transonic region, and the Mach number where the rocket exits the transonic region can be calculated by a function of the nosecone length to the diameter.

$$M_D = -0.0156\left(\frac{L_N}{d}\right)^2 + 0.136\left(\frac{L_N}{d}\right) + 0.6817$$

$$M_E = a\left(\frac{L}{d}\right)^b + 1.0275$$

Where

$$a = 2.4 \text{ for } \left(\frac{L_N}{L}\right) < 0.2 \tag{2.17}$$

$$a = -321.94\left(\frac{L_N}{L}\right)^2 + 264.07\left(\frac{L_N}{L}\right) - 36.348 \text{ for } \left(\frac{L_N}{L}\right) \geq 0.2$$

$$b = -1.05 \text{ for } \left(\frac{L_N}{L}\right) < 0.2$$

$$b = 19.634\left(\frac{L_N}{L}\right)^2 - 18.369\left(\frac{L_N}{L}\right) + 1.7434 \text{ for } \left(\frac{L_N}{L}\right) \geq 0.2$$

If the current mach number falls within the predicted transonic region, a ΔC_D is calculated to add to the total drag coefficient. The maximum transonic wave drag

can be calculated by

$$\Delta C_D = c\left(\frac{L}{d}\right)^g \text{ for } \frac{L}{d} \geq 6$$
$$\Delta C_D = c(6)^g \text{ for } \frac{L}{d} < 6$$

Where (2.18)

$$c = 50.676\left(\frac{L_N}{L}\right)^2 - 51.734\left(\frac{L_N}{L}\right) + 15.642$$
$$g = -2.2538\left(\frac{L_N}{L}\right)^2 + 1.3108\left(\frac{L_N}{L}\right) - 1.7344$$

The maximum transonic drag coefficient is multiplied by a polynomial F

$$F = -8.3474x^5 + 24.543x^4 - 24.946x^3 + 8.6321x^2 + 1.1195x$$
$$x = \left[\frac{M - M_D}{M_E - M_D}\right] \quad (2.19)$$

Supersonic Wave Drag

Once beyond the transonic region, the supersonic wave drag rise is assumed to be equivalent to the max transonic wave drag coefficient as defined in (2.18).

2.4.4 Normal Coefficient

The normal coefficient is the coefficient in each of the y and z axes because this model is derived for uncoupled aerodynamics. Because the rocket is assumed to be symmetric, this coefficient is calculated using the same method for each axis. The only differences in their values arise from different control deflections or aerodynamic angles. In all of the equations, α is used to represent the angle of attack.

Body

Typical derivations of normal force coefficients for body components results in a derivation that requires a change in the cross-sectional area to generate a force. In

the case of a cylindrical body, these predictions would produce a 0 force coefficient. However, at a non-zero angle of attack, the body will produce lift. Galejs proposed a correction term to account for body lift on a cylindrical body [22].

$$C_N = K\frac{A_{plan}}{A_{ref}}\sin^2\alpha \tag{2.20}$$

K is a constant $K \approx 1.1$ and the plan form area is defined by $A_{plan} = d * l$.

Fins

To calculate the normal coefficient produced by each of the fins, Niskanen proposes that the derivative with respect to angle of attack be calculated instead [5]. At angles of attack near 0, the derivative reduces to a linear relationship between C_N and α as shown in (2.21).

$$C_{N_\alpha} = \frac{\partial C_N}{\partial \alpha} \qquad C_{N_\alpha} \approx \frac{C_N}{\alpha} \tag{2.21}$$

The C_{N_α} for an individual fin is derived from Diederich finite flat plate theory [23]. The model was extended to trapezoidal fins in Barrowman [24] and to free form fin shapes in Niskanen [5]. This analysis will follow the derivation for trapezoidal fins, as the aeroprediction is intended to be used as first order approximation and the method is more computationally inexpensive as compared to free form fin calculation. For each fin or canard

$$C_{N_{\alpha_{fin}}} = K_t\frac{8\left(\frac{h}{d}\right)^2}{1+\sqrt{1+\left(\frac{\beta L}{C_r+C_t}\right)^2}}\sin^2(\alpha+\delta) \tag{2.22}$$

Where β is the compressibility factor defined by (2.23) and L is the length along the mid chord of the fin.

$$\beta = \sqrt{|M^2-1|} \tag{2.23}$$

For supersonic flow, the normal force coefficient is a 3rd order expansion as described by Niskanen and derived from Busemann theory[25], assuming that the fins are flat

plates.

$$
\begin{aligned}
K_1 &= \frac{2}{\beta} \\
K_2 &= \frac{(\gamma+1)M^4 - 4\beta^2}{4\beta^4} \\
K_3 &= \frac{(\gamma+1)M^8 + (2\gamma^2 = 7\gamma - 5)M^6 + 10(\gamma+1)M^4 + 8}{6\beta^7} \\
C_{N_{\alpha_{fin}}} &= \frac{A_{fin}}{A_{ref}}(K_1 + K_2\alpha + K_3\alpha^2)
\end{aligned}
\tag{2.24}
$$

The final effect on normal coefficient that is accounted for here is the body fin interaction. This is accounted for by a ratio of the body size to the fin size. [24] gives this ratio term as

$$K_F = 1 + \frac{r_t}{h + r_t} \tag{2.25}$$

The final normal coefficient for each fin can then be written as

$$C_{N_{fin}} = \alpha K_F C_{N_{\alpha_{fin}}} \tag{2.26}$$

2.4.5 Rolling Dynamics

In an ideal sounding rocket, roll would not occur. The largest contributor on the rocket is canted or deflected fins. Canted fins of less than a few degrees often occur due to manufacturing imperfections, and in the case of a controlled or actively stabilized rocket, the control deflections may induce a roll. In this thesis, two roll dynamic effects will be accounted for: roll forcing and roll damping.

Roll Moment Coefficient

The roll forcing coefficient arises from the moment applied to the rocket by a fin or canard deflection with respect to the free stream flow. As calculated in section 2.4.4, the force produced by a fin is a function of the geometry, Mach number, and angle of attack. The roll moment coefficient is simply this calculated coefficient multiplied by the moment arm

$$C_l = \frac{(y_{mac} + r_t)C_{N_\alpha}\delta}{d} \tag{2.27}$$

where δ is the sum of the cant angle and the control deflection.

Roll Damping Coefficient

Roll damping occurs from the local velocity of the fins as the rocket rotates. As the rocket spins, a radial velocity is generated as seen in figure 2.5. This velocity causes a high pressure zone on one side of the fin and a low pressure zone on the other side. This pressure differential is what causes the damping coefficient. As the damping effect grows further away from the roll axis, Niskanen [5] and Barrowman [24] propose dividing the fins into discrete strips. In this analysis it is assumed that the deflection and cant angles are both 0.

$$\begin{aligned} C_{ld} &= \frac{C_{N_{\alpha_{fin}}}\omega}{A_{ref}dv_0}F_{trap} \\ &\text{Where,} \\ F_{trap} &= \frac{C_r + C_t}{2}r_t^2 h + \frac{C_r + C_t}{2}r_t h^2 + \frac{C_r + C_t}{2}h^3 \end{aligned} \tag{2.28}$$

F_{trap} is shape-specific and is shown here for a trapezoidal fin.

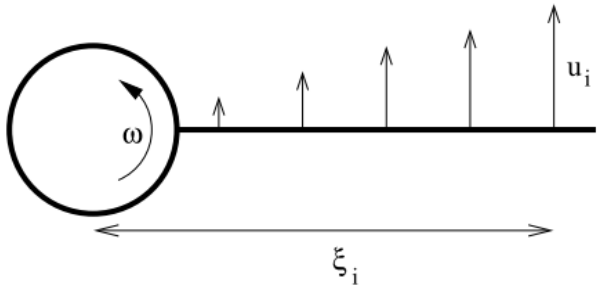

Figure 2.5: Radial Velocity of Fins [5]

2.5 Equations of Motion

In this simulation, standard 6 degree of freedom (6DOF) equations of motion are used. These take into account the 3 translational and 3 rotational equations of motion.

2.5.1 Translation

$$\mathbf{F} = m\mathbf{a} \tag{2.29}$$

The translational equations of motion are a result of the forces experienced during flight. The largest of which are thrust (during boost phase), drag, and gravity. These forces are applied in the body frame of the rocket. However, the relationship between force and acceleration shown in (2.29) are only valid in an inertial frame. The inertial position and velocity of the rocket must also be known to model the aerodynamics accurately.

By calculating all of the forces in the body frame either through direct calculation or a rotation matrix, the sum of the forces that act on the rocket can be written as

$$\begin{aligned} \ddot{x} &= G_x + \tfrac{T}{m} - \tfrac{F_x}{m} \\ \ddot{y} &= G_y - \tfrac{F_y}{m} \\ \ddot{z} &= G_z - \tfrac{F_z}{m} \end{aligned} \tag{2.30}$$

where **G** is the gravity vector that has been rotated into the body frame, T is the applied thrust, and F is the aerodynamic force in the body frame. These equations must then be rotated into the inertial frame before being integrated.

Rotation Matrix

The rotation matrix is transformation matrix that rotates one frame into another. The body to inertial matrix is a function of the Euler angles and can be written as

$$\begin{bmatrix} \cos\psi\cos\theta & \cos\theta\sin\psi & -\sin\theta \\ \cos\psi\sin\phi\sin\theta - \cos\phi\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\psi\sin\theta & \cos\theta\sin\phi \\ \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi & \cos\phi\cos\theta \end{bmatrix} \tag{2.31}$$

2.5.2 Rotation

The rotation angles of the rocket are dependent on the inertia tensor, the current angles, and the applied moments. Much like the translational equations of motion, the rotational equations of motion can be derived from first principles. The moment vector, $\bar{M}$, or the moment in each of the 3 body axes, is related to the angular momentum vector, $\bar{H}$, via the equations

$$\begin{aligned} \bar{M} &= \dot{\bar{H}} \\ \bar{H} &= \bar{I}\bar{\omega} \end{aligned} \tag{2.32}$$

Taking the derivative of $\bar{H}$ as shown in Jenkins [26], and assuming that the cross axis inertias (I_{xy}, I_{xz}, I_{yz}) are equal to 0, the moment equation for the rotation of the rocket can be written as

$$\begin{aligned} M_x &= I_{xx}\dot{P} + (I_{zz} - I_{yy})QR \\ M_y &= I_{yy}\dot{Q} + (I_{xx} - I_{zz})PR \\ M_z &= I_{zz}\dot{R} + (I_{yy} - I_{xx})PQ \end{aligned} \tag{2.33}$$

These equations are also known as Euler's equations of motion and are solved for the body rate derivatives and integrated to obtain the body rates.

2.6 State Estimation

The state estimator used in this thesis is the extended Kalman filter. The extended Kalman filter is an extension of the Kalman filter, both of which are presented in their general forms below.

2.6.1 The Kalman Filter

The Kalman Filter is used to estimate the time-varying state of a system. It was first proposed in 1960 by Rudolf Kalman [27] and has since been used extensively in the aerospace industry. The filter uses a system model to predict the next state, called the *a priori* estimate, and then uses weighted measurements from the system to correct for any errors in the propagation. The weighting matrix used is the Kalman gain which is a function of the estimated errors in the design of the filter.

The system dynamics of the system can be written in a matrix form called the state space form. The state space form of a system is written as

$$\begin{aligned} \dot{x} &= Ax + Bu \\ z &= Cx + Du \end{aligned} \tag{2.34}$$

A is the state transition matrix between the current state and the derivative of the current state. **B** is the transition matrix between the control input, u, and the resultant change in the state derivative. Similarly, C and D represent the state transition matrices for the measurement equations.

To formulate the Kalman filter a rewrite of the system must be performed to put the system in a discrete form. In this step, noise will be added to the system. w and v are noise matrices that are assumed to have a Gaussian distribution. w is the process noise or the random changes in the state that are unaccounted for in the

system dynamics.

$$\hat{x}_k^- = F_k \hat{x}_{k-1} + B_k u_k + w_k \tag{2.35}$$

For example, in the case of the sounding rocket, one may choose the coefficient of drag (C_x) as a state variable. While C_x is relatively constant over the flight of most sounding rockets, there will be slight changes with changes as Mach number and angle of attack change. The process noise helps to correct for these small deviations.

v is the measurement noise. As discussed in section 2.8, the sensors typically used in low-cost sounding rockets are noisy, which results in large errors in the estimated state over time. The Kalman filter assumes that both of the noise matrices can be estimated by a Gaussian white noise process.

$$z_k = y_k + v_k \tag{2.36}$$

The Kalman gain is the weighting that combines the *a priori* state estimate and the measurements. The higher the Kalman gain, the more trust in the measurements for the final estimated state. The lower the gain, the more confidence in the projected state. To calculate the Kalman gain, the filter uses covariance matrices, which are statistical random noise processes that relate different states together. The covariance matrix diagonals are the covariance of each state.

$$\begin{aligned} \mathbf{P}_k^- &= \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- + \mathbf{R_k} \end{aligned} \tag{2.37}$$

$\mathbf{H}$ is the measurement matrix and transforms the full state vector into the observable states.

Finally, the Kalman gain can be calculated. As seen in (2.38), the noise covariance matrix, $\mathbf{R}_k$, must be positive definite because it will be inverted.

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{2.38}$$

The final estimated state for time step k is then estimated as

$$\hat{x}_k = \hat{x}^- + \mathbf{K}_k(z_k - \mathbf{H}\hat{x}^-) \tag{2.39}$$

at which point the filter begins again with the *a priori* state estimate at time $k+1$.

2.6.2 The Extended Kalman Filter

The extended Kalman filter (EKF) is a method to use the Kalman filter for non-linear states. The EKF accomplishes this by linearizing the state transition matrix at each time step through the use of the Jacobian. The state projection then takes the form

$$\hat{x}_k = \mathbf{I} + \mathbf{J}_k T_s + ... \tag{2.40}$$

which is the Taylor expansion of $\mathbf{J}_k$ about 0.

The EKF also can be used for states with nonlinear measurements by following the same process for $\mathbf{H}$. However, here the measurement is assumed to be linear.

2.7 Control Theory

This section discusses the control law use in HSF to command the fin deflections in the general form. Sounding rocket specific implementations are presented in chapters 3 and 5.

2.7.1 LQR Controller

The Linear Quadratic Regulator (LQR) controller is a solution to the optimal control problem that minimizes an energy-like cost function. More generally, optimal control theory finds the controller that optimizes the performance with respect to some measure of performance [28].

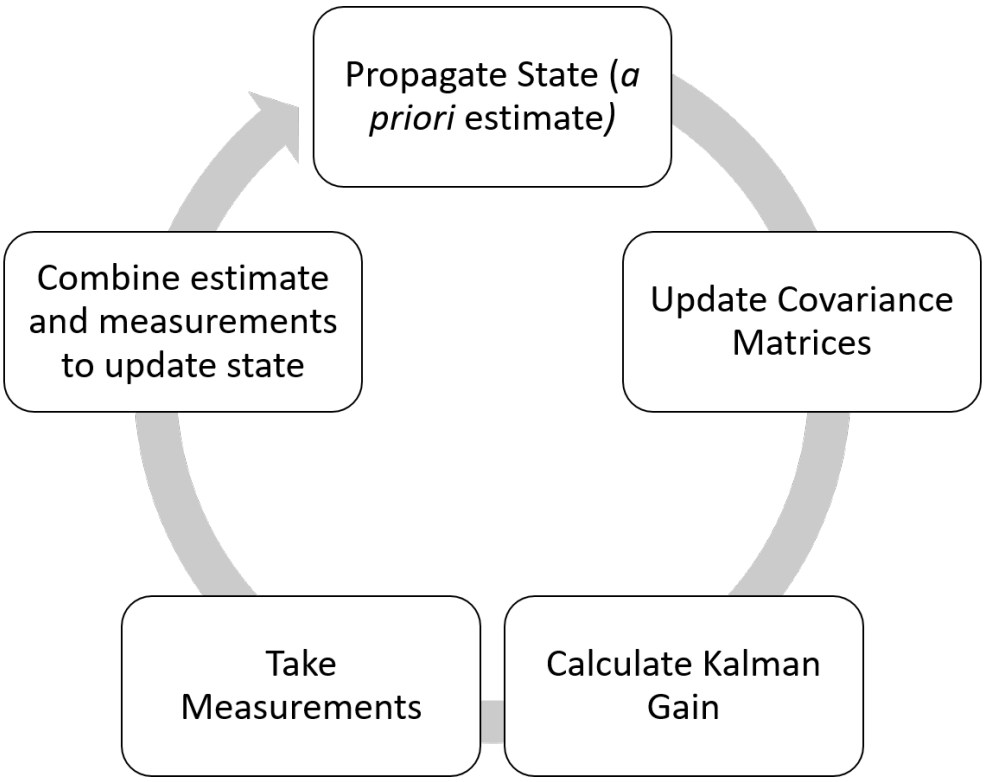

Figure 2.6: The Extended Kalman Filter Process

The control law is defined as a simple negative full state feedback control

$$u = -Kx \tag{2.41}$$

where K results from the solution to the Algebraic-Ricatti Equation.

$$0 = A^T P + PA + C^T QC - (PB + G^T QD)(H^T QD + \rho R)^{-1}(B^T P + D^T QC) \tag{2.42}$$

(2.42) is solved for P for use in

$$K = (C^T QD + \rho R)^{-1}(B^T P + D^T QC) \tag{2.43}$$

A, B, C, and D are defined in section 2.6.1 and are the state space variables. Q and R are the weighting matrices. For an initial guess for these two matrices, Bryson's rule was employed.

$$Q = \frac{1}{max(y_k^2)} \quad R = \frac{1}{max(u_k^2)} \tag{2.44}$$

Bryson's rule normalizes the state error and the control inputs so that they are weighted the same. This is typically just a starting point in the controller design; $\mathbf{Q}$ and $\mathbf{R}$ are adjusted through trial-and-error to improve performance of the controller.

2.8 Sensors

There are multiple sensors implemented inside of HSF to compute the necessary measurements for the state estimator. The sensor models are generic for each model type and take the truth value returned from the equations of motion and adds the necessary noise terms to accurately simulate the sensors. The noise specifications are loaded through the model input XML file.

Accelerometers

Accelerometers measure the total acceleration on the sensor. There are many different types of accelerometers that range in both cost and accuracy. As the focus of this thesis is on relatively low-cost sounding rockets, microelectromechanical (MEMs) devices are discussed and simulated here.

Accelerometers are subject to many noise sources. The largest sources are bias, white noise, scale factors errors, mounting errors, and non-linearity. Other effects include cross-axis sensitivity, and 2nd order and higher effects. In this thesis, only the larger effects are accounted for in the model. It is also assumed that the accelerometer is located at the center of gravity (CG) of the rocket, therefore neglecting any centripetal acceleration effects.

$$\mathbf{a}_{meas} = \mathbf{a}_{true} + \mathbf{a}_{bias} + \mathbf{a}_{cross} + \delta\mathbf{a}_{scale} + \omega \tag{2.45}$$

a_{true} is the actual value of acceleration, a_{bias} is the measurement bias and is assumed

to be constant over the course of the flight, a_{cross} is defined by (3.2) and encompasses both mounting errors and cross-axis sensitivity, δa is the error in scale factor, and ω is the Gaussian noise term.

$$\mathbf{a}_{cross} = \delta_{cross}\mathbf{a}_{true} = \begin{bmatrix} 0 & a_y & a_z \\ a_x & 0 & a_z \\ a_x & a_y & 0 \end{bmatrix} \mathbf{a}_{true} \tag{2.46}$$

In its default configuration in HSF, the accelerometer sensor is representative of a low-cost MEMs accelerometer, but this can be modified by the user to match their specific sensor configuration.

Gyroscopes

Gyroscopes directly measure the angular rate in the sensor frame, and they are similar to accelerometers in that there are many different variations, each with their advantages and disadvantages. MEMs gyroscopes also have a very similar noise and bias profile to that of a MEMs accelerometer. Additionally, a gyroscope has an acceleration dependency, but this term is small [4] and is neglected in this thesis.

$$\mathbf{g}_{meas} = \mathbf{g}_{true} + \mathbf{g}_{bias} + \mathbf{g}_{cross} + \delta\mathbf{g}_{scale} + \omega \tag{2.47}$$

Barometer

A barometer measures the static pressure at the sensor. Using this pressure and a model of the atmosphere the altitude can be estimated. There are some problems with using the sensor during the flight which will cause errors in the altitude. The atmosphere is not constant over time, and a specified pressure does not always correspond to the same altitude. Fortunately, the lower atmosphere tends to follow the same exponential pressure gradient. The measured pressure to altitude conversion begins to diverge as the rocket approaches the transonic region due to shock waves

causing rapid changes in pressure. In the simulation, the barometer stops reporting data once above Mach 0.6, but in flight code, the barometer measurements will need to be ignored if the estimated velocity exceeds Mach 0.6. Another potential source of error is a difference of pressure between the inside of the rocket and the atmospheric pressure. This error can be mitigated by proper design and vent holes, so it is neglected in this simulation. The final source of error is the noise in the pressure measurement from the sensor.

The pressure measurement also needs to be converted to an altitude. This conversion utilizes an exponential atmosphere model and compensates the measurement for local temperature variation. P_{true} comes from the atmospheric model selected and the corresponding truth EOMs.

$$P_{meas} = P_{true} + \omega \qquad P_{atmos} = 1.225\,\mathrm{e}^{-x/8420} \tag{2.48}$$

Magnetometer

The magnetometer measures the local magnetic field vector, which can be used to find the attitude in two dimensions. In this simulation, it is used to determine the heading in the inertial frame. Similar to the gyroscope and accelerometer, its errors include cross-axis sensitivity, Gaussian white noise, and bias. In addition to the sensor errors, magnetometers are also subject to so-called soft iron and hard iron local effects. These cause changes in the local magnetic field by the introduction of magnets or large electrical currents. These effects are not modeled in HSF, as the effects are typically realized by a bias in the sensor or are beyond the scope of this thesis to estimate.

$$\mathbf{m}_{meas} = \mathbf{m}_{true} + \mathbf{m}_{bias} + \mathbf{m}_{cross} + \omega \tag{2.49}$$

GPS

GPS is a collection of satellites run by the US Air Force that provides absolute position measurements around the globe. In HSF, a simplified GPS model is implemented. The GPS subsystem takes the Position_LLA dynamic state property and adds Gaussian white noise to the position.

This chapter presented all the models and tools that are necessary to build up the complete sounding rocket model. Most of these models and tools are not specific to the sounding rocket problem and can be used for many different vehicles.

Chapter 3

SIMULATION IMPLEMENTATION

The models specific to the sounding rocket are presented here. The extended Kalman filter and LQR controller equations and design points for the sounding rockets used in this thesis are presented. The performance of the Kalman filter is compared to the truth simulation values and finally the interactions between subsystems in HSF are described in detail.

3.1 The Extended Kalman Filter

For the extended Kalman filter, 18 states were selected: position, velocity, Euler angles, body rates, and the six primary aerodynamic coefficients. These were chosen as they fully define the states to estimate the trajectory flown. The aerodynamic coefficients are included in the state estimate because they directly influence the other states. Without including estimates for these states that change over the course of the flight, the Kalman estimated state would begin to diverge. Unlike in (3.8), a truth model for these states, are difficult to create as they are highly dependent on the external environment.

$$\bar{x} = \begin{bmatrix} \mathbf{P} \\ \mathbf{v} \\ \mathbf{E} \\ \omega \\ \mathbf{C_T} \\ \mathbf{C_R} \end{bmatrix} \quad \text{where} \quad \begin{aligned} \mathbf{P} &= \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \quad \mathbf{E} = \begin{bmatrix} \psi \\ \theta \\ \phi \end{bmatrix} \\ \omega &= \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad \mathbf{C_T} = \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \quad \mathbf{C_R} = \begin{bmatrix} C_l \\ C_N \\ C_M \end{bmatrix} \end{aligned} \tag{3.1}$$

In the interest of brevity, only one of each of vector states will be shown here.

Each of the other states are each similar and only vary in angles and constants. The complete formulation can be found in Appendix 6.2. The derivatives of the states that will be used are shown in (3.7).

A priori State Estimate

$$\dot{\hat{x}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\psi} \\ \dot{p} \\ \dot{C}_x \\ \dot{C}_l \end{bmatrix} = \begin{bmatrix} \ddot{x} \\ g\cos\psi\sin\theta + \dfrac{T}{m} - \dfrac{Q_p A_{ref}}{m} C_x \\ q\sin\phi + \dfrac{r\cos\phi}{\cos\theta} \\ \dfrac{Q_p A_{ref}}{m} C_l - \dfrac{(I_{zz} - I_{yy})qr}{I_{xx}} \\ 0 \\ 0 \end{bmatrix} \tag{3.2}$$

In this formulation of the state, it is assumed that the thrust (T), mass (m), inertia matrix ($\mathbf{I}$), and dynamic pressure (Q_p) are known. Any deviations in these values are accounted for in the process noise matrix, $\mathbf{Q}$. It is also assumed that the aerodynamic coefficients' derivatives are 0. While in reality, this is not the case, the change in the coefficients is relatively small throughout the flight, and it is hard to predict with any degree of confidence with the information available to the state estimator.

The dynamic pressure is a function of velocity and density. As the rocket will not have the same atmospheric information as the simulation, the estimator uses an exponential atmosphere model where density is defined as

$$\rho = 1.225\,\mathrm{e}^{-x/8420} \tag{3.3}$$

which returns the density at altitude x in kg/m^3. Using the estimated density and

velocity, the dynamic pressure can be written as

$$Q_p = \frac{\hat{\rho}\dot{\hat{x}}^2}{2} \tag{3.4}$$

Using this information and (3.7), the state is propagated forward using a simple Euler integration with a time step of 0.001 seconds to create the *a priori* state estimate.

Measurements

In the sounding rocket model, measurements come from the accelerometer, gyroscope, barometer, and GPS. All of these measurements are fed into the EKF through **H**. As can be seen in (3.10), the **H** for the reduced system consists of the states that are directly measured and one–velocity–that is not directly measured but can be calculated from the accelerometer data. The equation for this is simply $v = aT_s$.

$$z_k = \begin{bmatrix} 1 & T_s & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \psi \\ p \\ C_x \\ C_l \end{bmatrix} + v_k \tag{3.5}$$

The measurement noise matrix v comes from the modeled sensors and is used to generate $\mathbf{R}_k$. In this filter, $\mathbf{R}_k$ is generated by taking the square of the variance in measurement noise of each sensor.

$$\mathbf{R}_{k_{ii}} = \sigma_{k_i}^2 \text{ for sensor i} \tag{3.6}$$

However, as stated above, **R** needs to be positive definite to be invertible. To accomplish this all the zeros on the diagonal are replaced with a small number. This

does not affect the resultant Kalman gain because these small numbers are "removed" from the solution through **H**.

Kalman Gain

To properly fuse the *a priori* estimate and measurements to generate the *a posteriori* estimate of the state, the Ricatti equations must be solved. The following section will outline the process used to create the necessary matrices specific to the sounding rocket problem. The first step is to generate the state transition matrix. As this thesis uses the Extended Kalman Filter, the state transition matrix is a function of the Jacobian of the state. The Jacobian for the reduced system is shown in (3.12).

$$F = \begin{bmatrix} \frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial \dot{x}} & \frac{\partial \dot{x}}{\partial \psi} & \frac{\partial \dot{x}}{\partial p} & \frac{\partial \dot{x}}{\partial C_x} & \frac{\partial \dot{x}}{\partial C_l} \\ \frac{\partial \ddot{x}}{\partial x} & \frac{\partial \ddot{x}}{\partial \dot{x}} & \frac{\partial \ddot{x}}{\partial \psi} & \frac{\partial \ddot{x}}{\partial p} & \frac{\partial \ddot{x}}{\partial C_x} & \frac{\partial \ddot{x}}{\partial C_l} \\ \frac{\partial \dot{\psi}}{\partial x} & \frac{\partial \dot{\psi}}{\partial \dot{x}} & \frac{\partial \dot{\psi}}{\partial \psi} & \frac{\partial \dot{\psi}}{\partial p} & \frac{\partial \dot{\psi}}{\partial C_x} & \frac{\partial \dot{\psi}}{\partial C_l} \\ \frac{\partial \dot{p}}{\partial x} & \frac{\partial \dot{p}}{\partial \dot{x}} & \frac{\partial \dot{p}}{\partial \psi} & \frac{\partial \dot{p}}{\partial p} & \frac{\partial \dot{p}}{\partial C_x} & \frac{\partial \dot{p}}{\partial C_l} \\ \frac{\partial \dot{C}_x}{\partial x} & \frac{\partial \dot{C}_x}{\partial \dot{x}} & \frac{\partial \dot{C}_x}{\partial \psi} & \frac{\partial \dot{C}_x}{\partial p} & \frac{\partial \dot{C}_x}{\partial C_x} & \frac{\partial \dot{C}_x}{\partial C_l} \\ \frac{\partial \dot{C}_l}{\partial x} & \frac{\partial \dot{C}_l}{\partial \dot{x}} & \frac{\partial \dot{C}_l}{\partial \psi} & \frac{\partial \dot{C}_l}{\partial p} & \frac{\partial \dot{C}_l}{\partial C_x} & \frac{\partial \dot{C}_l}{\partial C_l} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{\hat{\rho}\hat{\dot{x}}^2 A_{ref}}{16840m} C_x & \frac{\hat{\rho}\hat{\dot{x}} A_{ref}}{m} C_x & -g\sin\psi & 0 & \frac{\hat{\rho}\hat{\dot{x}}^2 A_{ref}}{2m} & \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-\hat{\rho}\hat{\dot{x}}^2 A_{ref}}{16840m} C_l & \frac{\hat{\rho}\hat{\dot{x}} A_{ref}}{m} C_l & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.7}$$

By using a Taylor series expansion, the state transition matrix can be approximated from the Jacobian, and because the state transition matrix will only be used to solve the Ricatti equations and not propagate the state, only a two-term expansion is used [29].

$$\Phi \approx \mathbf{I} + \mathbf{F}T_s \tag{3.8}$$

where $\mathbf{I}$ is the identity matrix.

As discussed above, the process noise matrix, $\mathbf{Q}$, represents the uncertainty in the propagated state. The higher the $\mathbf{Q}$ for a state, the more the measurement will be trusted. In the case of the sounding rocket, the dynamics are relatively well understood, and the biggest unknowns arise in the estimation of the aerodynamic coefficients and the variance in the known quantities, ρ, T, and m. Looking at $\mathbf{Q}$ below it is seen that the values chosen represent the relative certainty in each of the propagated values and it is assumed that all of the states are decoupled.

$$\mathbf{Q} = \begin{bmatrix} 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} 0.01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1e8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix} \tag{3.9}$$

These matrices and equations discussed above can be used in the equations presented in section 2.6.2 to estimate the Kalman gain and state covariance matrices.

3.1.1 Performance

To look at the performance of the Kalman filter, the filter was applied to the sounding rocket described in section 4.2.1. The simulation is conducted using the

real-time atmosphere model from the date of flight and no control system.

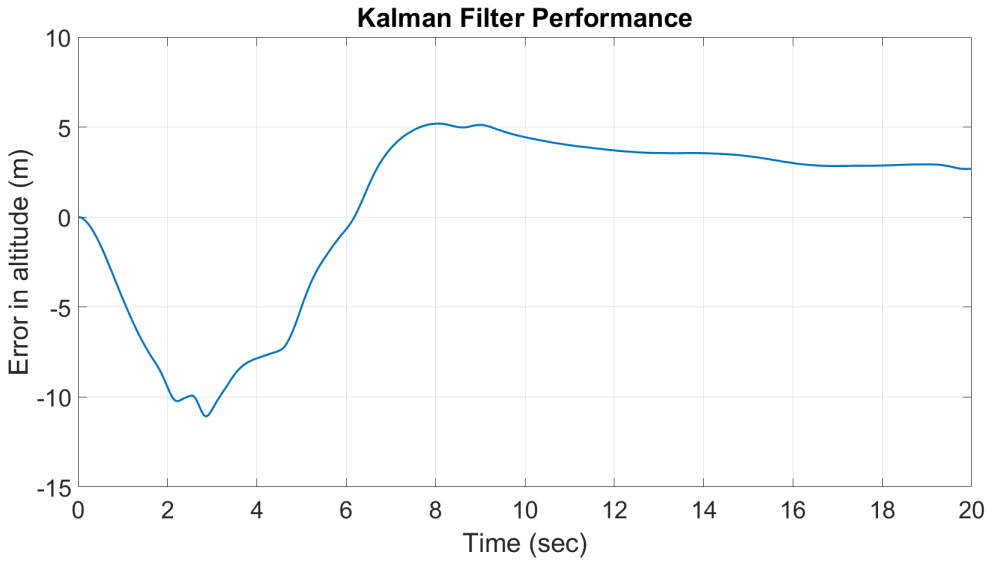

Figure 3.1: Kalman Filter Error in Estimated Altitude

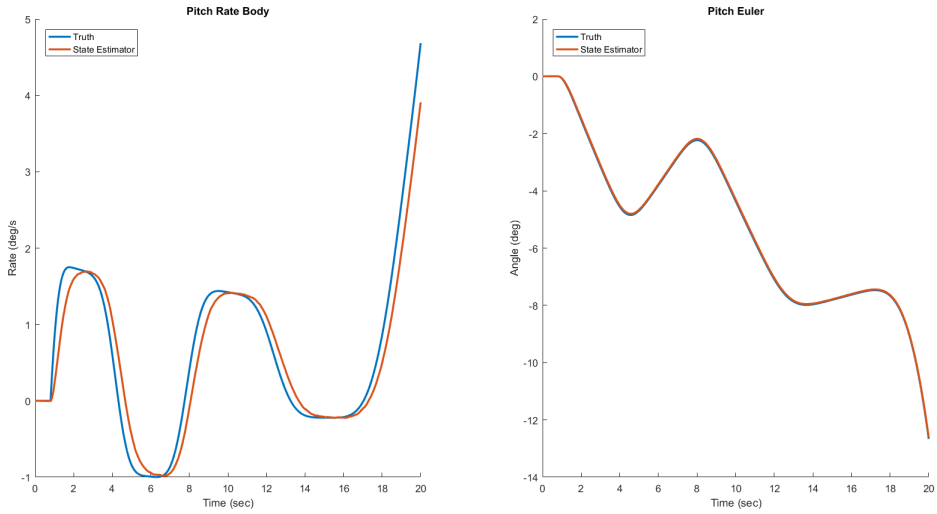

Figure 3.2: Kalman Filter Pitch Angle and Rate Estimation Compared to Truth Angles

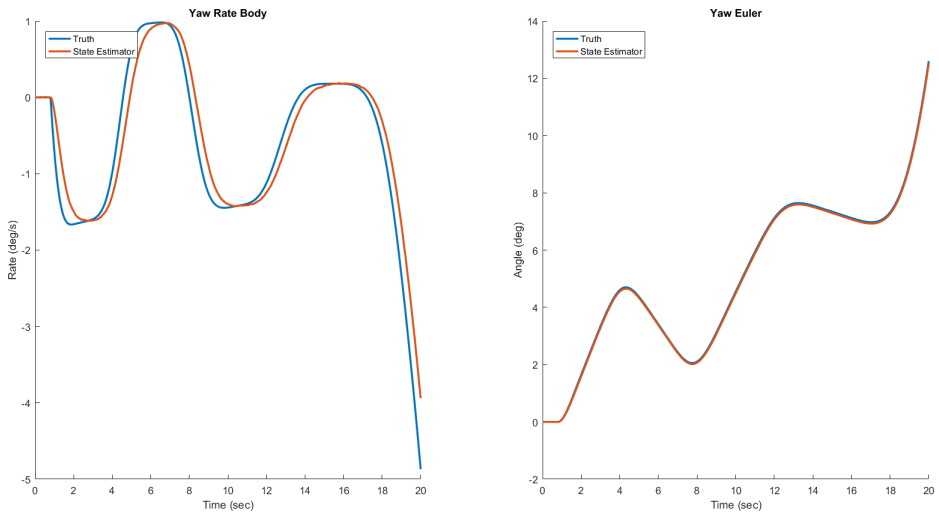

Figure 3.3: Kalman Filter Yaw Angle and Rate Estimation Compared to Truth Rates

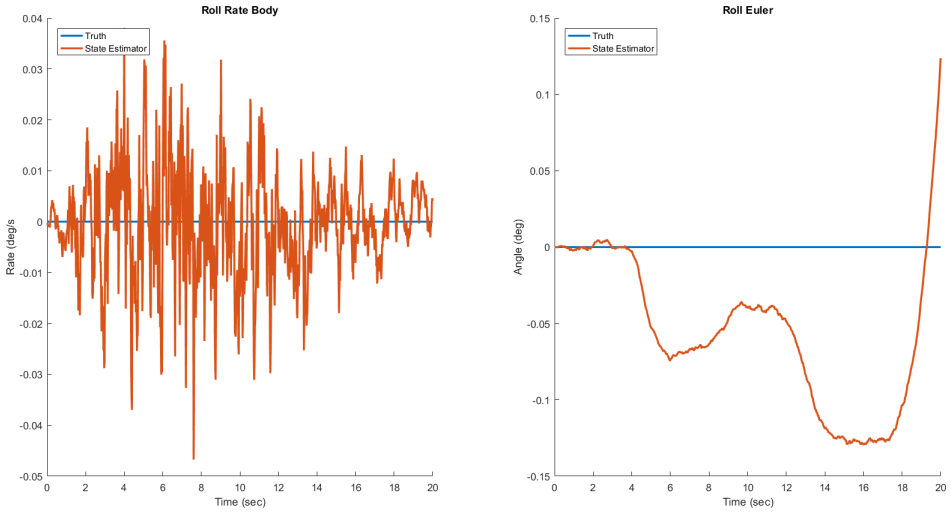

Figure 3.4: Kalman Filter Roll Angle and Rate Estimation Compared to Truth Rates

As seen in the figures above, the state estimator tracks the "truth" motion fairly well. In both pitch and yaw, the estimate lags behind the actual value slightly. This can be attributed to the assumption of a zero derivative of the aerodynamic coefficients, and the states are only corrected by the measurement from the gyroscope.

This lag can be decreased by increasing the process noise for the state. However, more sensor noise is included in the estimate when this is done. With the goal of using the state estimate in a controller, the EKF was designed to have a smoother profile with less process noise and more measurement noise.

It is unexpected that the Euler angles appear to be matched closer than the body rates, as the body rates are measured directly. This may have been caused by the noisy measurement from the gyroscope. This noise directly affects the estimate of the angle but only indirectly affects the Euler angle.

3.2 LQR Controller

Building on the general results presented in 2, the specific design points and linearization points are discussed below.

3.2.1 Control Loop

In this simulation, the fins are being used as the control surface. This means that only attitude can be controlled. To simplify the controller, the full state was not used as the input, but instead, only the Euler angles and body rates were used.

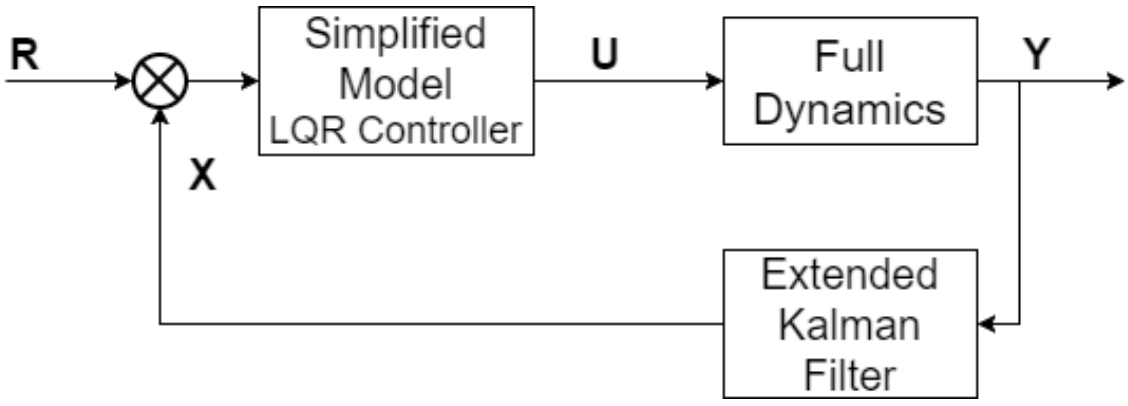

Figure 3.5: Control and Estimator Loop

3.2.2 Design

The LQR controller was designed by linearizing the controller around zero angle and zero angular rate. In normal flight conditions with no controller, this is an acceptable approximation, as the observed angles in flight typically are less than 5 degrees and with the inclusion of the controller, the approximation improves, because the controller will drive the rocket to this orientation. The plant was linearized by taking the Jacobian of the full plant dynamics and substituting in 0 for the angles and rates. Using this linear model, **K** was calculated in MATLAB using the lqrd command. This matrix was then put into the HSF controller subsystem to run the simulation.

$$\mathbf{A} = \begin{bmatrix} \frac{\sin\theta(r\cos\phi + q\sin\phi)}{\cos^2\theta} & 0 & \frac{q\cos\phi - r\sin\phi}{\cos\theta} \\ 0 & 0 & r\cos\phi - q\sin\phi \\ 0 & \frac{\sin^2\theta(r\cos\phi + q\sin\phi)}{\cos^2\theta} & \sin\theta\frac{r\cos\phi - q\sin\phi}{\cos\theta} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \\ 0 & \cos\phi & \sin\phi \\ 1 & 0 & 0 \\ 0 & \frac{-r(I_{zz} - Iyy)}{I_{xx}} & \frac{-q(I_{zz} - Iyy)}{I_{xx}} \\ \frac{-r(I_{xx} - Izz)}{I_{yy}} & 0 & \frac{-p(I_{xx} - Izz)}{I_{yy}} \\ \frac{-q(I_{yy} - Ixx)}{I_{zz}} & \frac{-p(I_{yy} - Ixx)}{I_{zz}} & 0 \end{bmatrix} \tag{3.10}$$

The control input matrix was estimated by calculating the estimated C_{N_α} based on the formulation made in section 2.4.4. Because the system is linearized, it is assumed

that the deflection angle of the fin corresponds to the angle of attack of the fin. While this is not always the case, especially as the wind speeds increase in the upper atmosphere and the rocket is controlled to a near 0 Euler angle, this approximation allows for a linear controller to be used. A potential improvement to the controller and state estimator would be to add angle of attack as an estimated state and control the fins based on the angle of attack of the fin instead of the deflection angle. The control matrix is then linearized by taking the Jacobian and can be written symbolically as

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ C_l/I_{xx} & C_l/I_{xx} & C_l/I_{xx} & C_l/I_{xx} \\ C_{N_\alpha}/I_{yy} & C_{N_\alpha}/I_{yy} & C_{N_\alpha}/I_{yy} & C_{N_\alpha}/I_{yy} \\ C_{M_\alpha}/I_{zz} & C_{M_\alpha}/I_{zz} & C_{M_\alpha}/I_{zz} & C_{M_\alpha}/I_{zz} \end{bmatrix} \tag{3.11}$$

3.3 Subsystems

The HSF subsystems created, in order of evaluation, are shown in figure 3.6. The arrows show the dependency functions that transfer information in between each subsystem. The equations of motion (EOMs) are not a subsystem in HSF but feeds information into the other subsystems, so it is included in this description.

IMU and GPS

The IMU subsystem consists of the accelerometer, gyroscope, magnetometer, and barometer sensors. The IMU subsystem adds the appropriate noise and biases to each sensor as described in section 2.8. In the CanPerform method, the subsystem gets the most recent dynamic state propagated by the EOMs and feeds this information to the sensor models. It stores the returned measurements the _newstate property

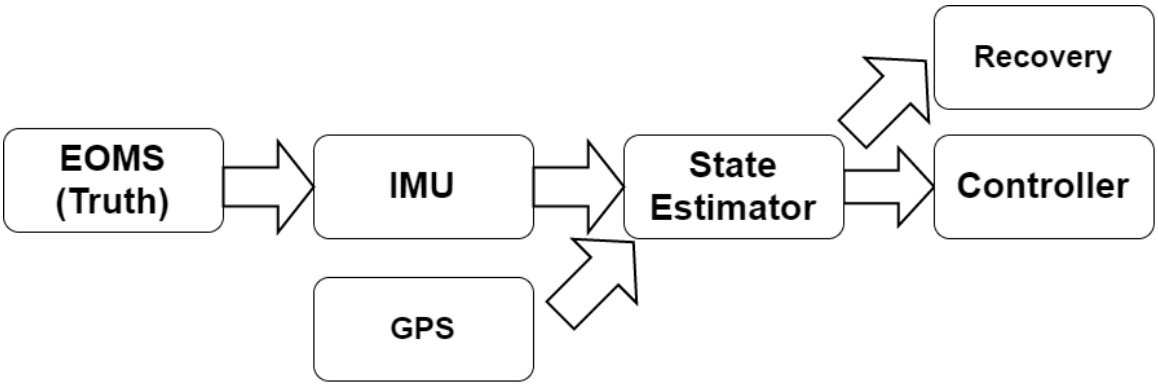

Figure 3.6: Subsystem Interactions within the Horizon Simulation Framework

of the subsystems. The dependency functions then access this _newstate property to get the needed data.

The GPS subsystem follows a similar implementation as the IMU, but it is implemented in Python instead of C# because this model is only valid for air vehicles.

State Estimator

The state estimator takes input through the dependency functions of both the IMU and GPS subsystems. The CanPerform method has no functionality and always returns true; the entire state estimator lies within the dependency function for the controller. The implementation of the estimator is described in section 2.6.1.

Controller

The controller subsystem takes the estimated state from the state estimator dependency and extracts the reduced controlled state. This state consists of the Euler angles and body rates. It then multiplies the LQR Kalman gain matrix and this state to get the desired fin deflections. The controller then applies limitations to the desired input base on specifications for the control actuators, such as rate limitation and saturation. This is a simple actuator model and is similar to using a second

order transfer function to represent the actuator. All of these calculations are done in the CanPerfom method and the control deflection angles are added to the integrator parameters object to be sent to the EOMs.

Recovery

The recovery subsystem is used to deploy both the main and drogue parachutes. It takes the estimated state as a dependency, and when the inertial x velocity crosses below 0, it deploys the drogue parachute. It notifies the rest of the simulation with a boolean StateVarKey. Other subsystems and the EOMs modify their behavior based on the status of this flag. The main parachute is similarly deployed at a user-configurable altitude (default is 300 m).

In this chapter, the implementation details of both the EKF and the LQR controller were discussed. Additionally, subsytems interactions within HSF were defined.

Chapter 4

VERIFICATION AND VALIDATION

This chapter compares the results of both the aeroprediction model and the full 6 DOF model to experimental results. The predicted drag and normal coefficients are compared to public wind tunnel test data and the 6DOF model is compared to flight tests conducted by the student rocketry group at Cal Poly San Luis Obispo. Errors between the simulation and the experimental tests are discussed.

4.1 Aerodynamics Prediction Verification

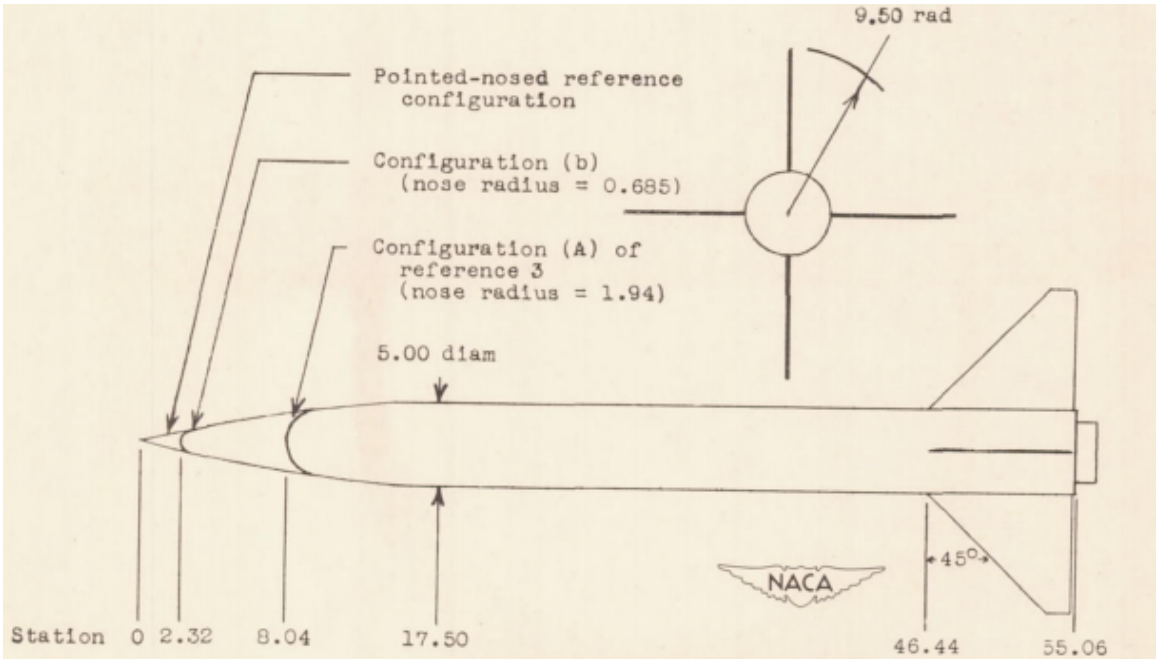

Figure 4.1: Dimensions of Drag Comparison Rocket (inches) [30]

To validate the aeroprediction models, wind tunnel data from publicly available reports were used. Hart tested multiple bodies and nose cone configurations in the transonic regime to characterize the coefficient of drag versus Mach number of each

of these configurations [30]. Here, the configuration that most closely resembles the typical configuration of sounding rockets today is the cylindrical body tube and a conical nosecone model and the dimensions are shown in figure 4.1, and as with all of the other sounding rockets used here, the surface roughness was selected to be 0.0012.

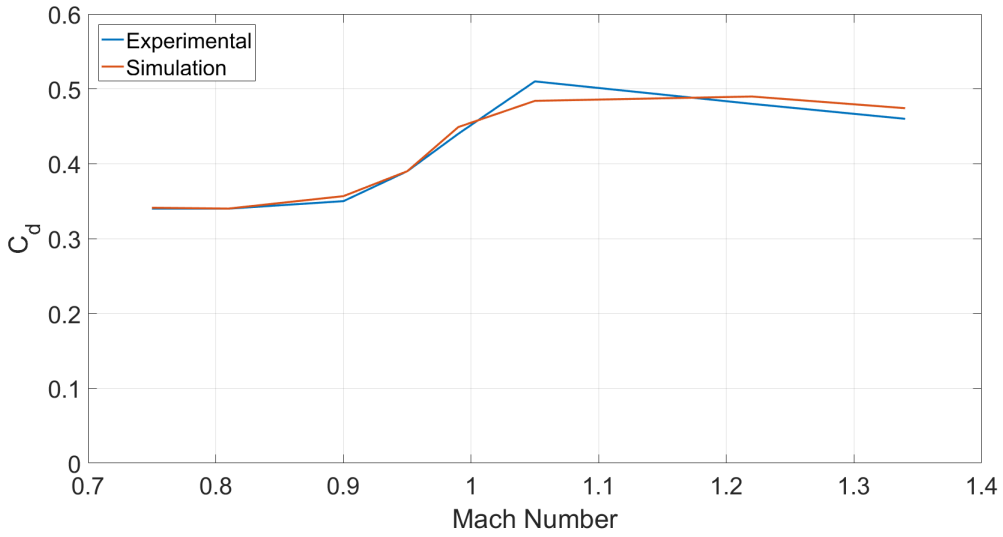

Figure 4.2: C_D Comparison of Aeroprediction Data vs Wind Tunnel Data

The experimental data compared to the output from the HSF aeroprediction method is shown in figure 4.2. There is good agreement between the two data sets. The largest error occurs at Mach 1.05, but it is hard to predict the aerodynamics at this speed because it is just after the point at which the vehicle goes supersonic. There is also a slight over-prediction of drag as the Mach number increases into the supersonic regime. This error can be attributed to modeling a slightly different nosecone than what was tested. The nose tested was a purely conical nose with a sharp point, but because of manufacturing difficulties and decreased drag at lower Mach numbers, the nose cone shape was assumed to have a slightly rounded tip in this derivation of the model. The data presented by Hart indicates that more rounded nosecones begin to produce more drag as the test enters the supersonic regime–similar to the modeled data in the figure. Even with this slight divergence, the error in C_D at a

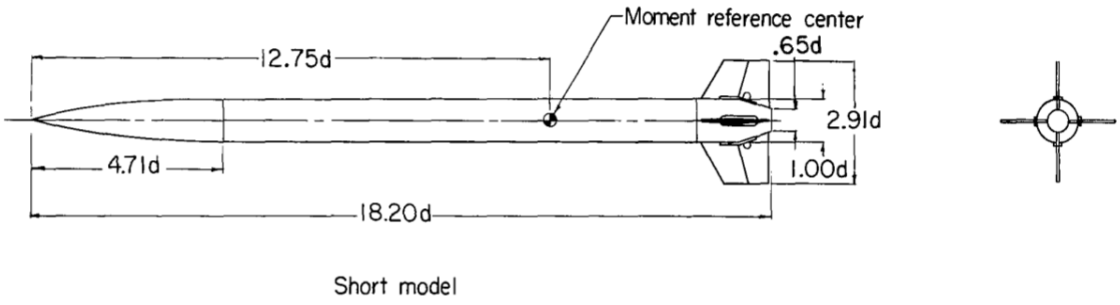

Figure 4.3: Dimensions of C_N Comparison Rocket (d = 5.72cm)[31]

Mach number of 1.34 is only 3%.

To validate the normal coefficient model data generated by Ferris was used [31]. The data for C_N was generated for multiple angles of attack at Mach numbers through the transonic regime. Because of limitations in the developed model, the fins were modeled as flat plates, but the sounding rocket tested had wedge-shaped fins. Additionally, the boat-tail in the rocket was not modeled as this functionality does not exist within the created aeroprediction model. However, the addition of a boat-tail has a small effect on the normal force.

At the lower Mach numbers, the coefficient is significantly under-predicted as the angle of attack increases. This under-prediction is a potential cause of the slight oscillation in the pitch and yaw angles that is discussed in section 4.2.1. However, at Mach 1.2, the prediction almost perfectly matches the experimental data.

4.2 Verification Flights

In order to validate the simulation, two flights were performed with small-scale sounding rockets. Both were launched on commercial solid motors that have a thrust curve repeatable within 10%. Each rocket was also flown with a full 9-axis IMU (accelerometer, gyroscope, and magnetometer), barometer, and GPS unit.

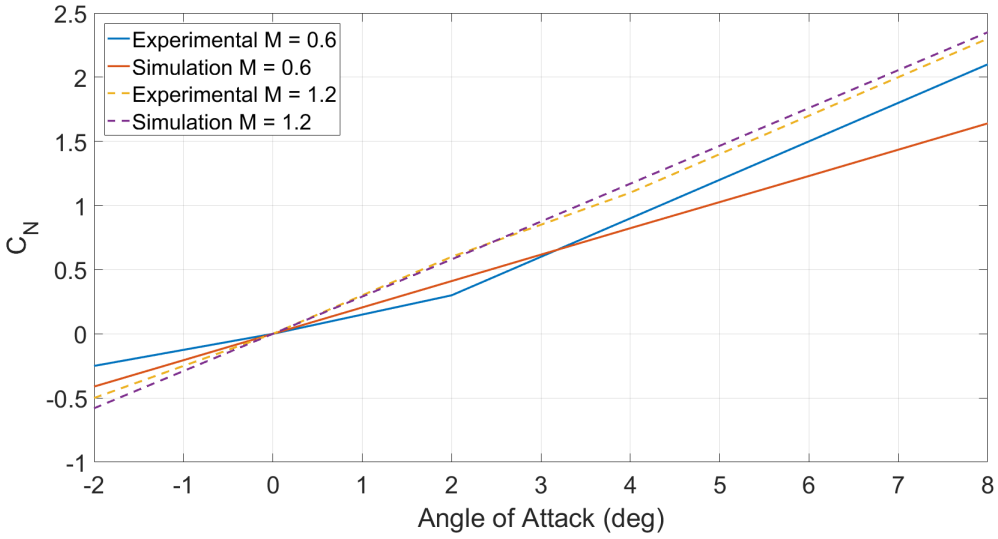

Figure 4.4: C_N Comparison of Aeroprediction Data vs Wind Tunnel Data

4.2.1 Flight of the Concord Sounding Rocket

The Concord Sounding Rocket was launched in May 2016 in the Mojave Desert and was conducted using a L-952 solid motor. Both the barometer and the GPS were used to estimate the altitude. However, a simple comparison between these two values show a large discrepancy in the measured value. Because the flight was conducted up to a relatively low altitude and at subsonic Mach numbers, the barometer should be more accurate than the GPS and will be used here.

Compared to the flight data, the simulation over-predicts the altitude by 15%. The cause is readily apparent in figure 4.7; the simulation under-predicts the cross-

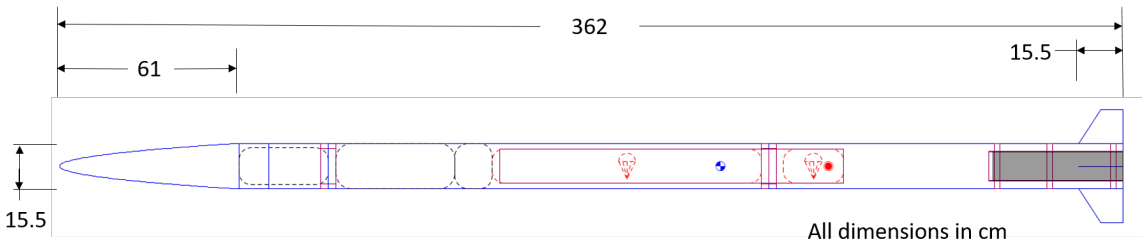

Figure 4.5: Outer Mold Line of the Concord Sounding Rocket

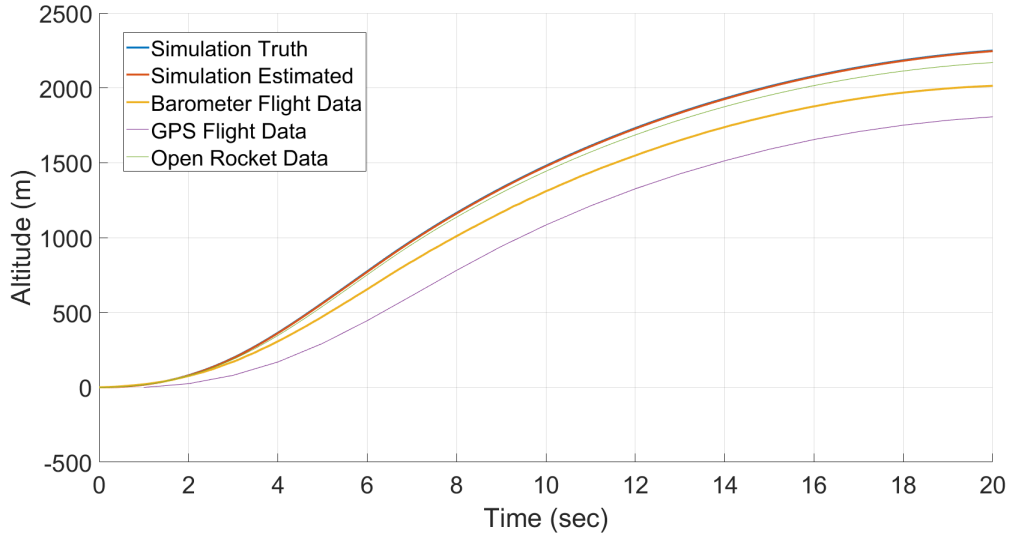

Figure 4.6: Altitude Comparison of Onboard Sensors and Simulation

range distance traveled. There are a few contributing factors to this under-prediction: a smaller than measured modeled wind in the atmosphere, the launch angle of the flight, and errors in the aerodynamic model. The model used for the atmosphere in the simulation was the real-time model at the date and time of the launch. However, when the surface level model output was compared to the launch site weather station data, there was an under-prediction in the wind speed. There were only measurements taken at the surface level, but this error in the model output would conceivably be experienced at higher altitude. In terms of the model, this is not unexpected as the gridded bins are relatively large in size as compared to the total distance traveled by the sounding rocket and there are many small-scale weather effects that would cause a higher than predicted wind speed. Another wind effect that was not modeled here is gusting winds. Gusting winds are short intervals of increased wind speed that can occur throughout the atmosphere. Since this effect is not modeled, it is unknown if this is what affected the trajectory of the rocket.

The second contributor to the error in the flight profile is an angle of launch off the launch rail in flight. In the simulation it was assumed that the launch was

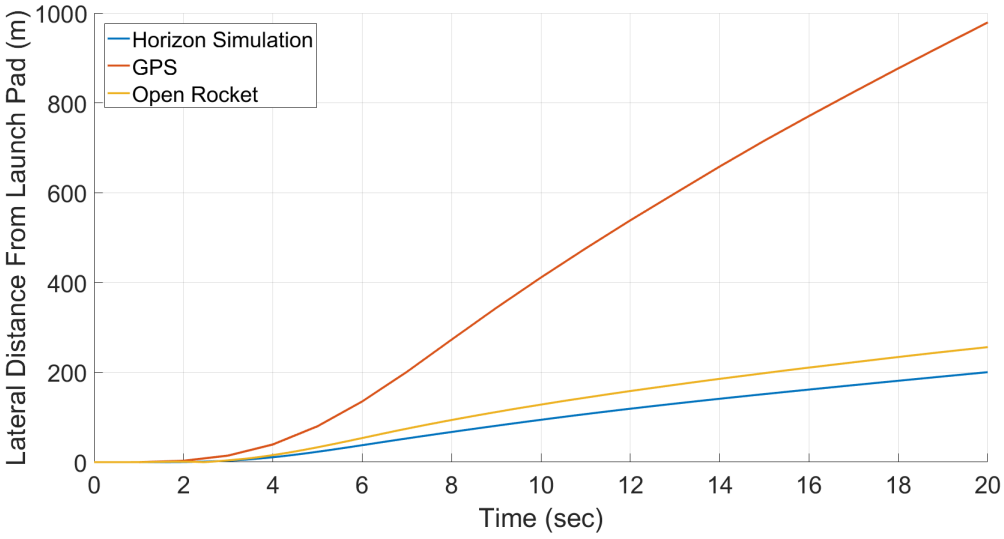

Figure 4.7: Downrange Distance Comparison of GPS and Simulation

perfectly vertical, while in reality this is difficult to attain. Through multiple runs of the simulation, it was observed that an error of 2 degrees off vertical causes a change in downrange distance of approximately 500 m.

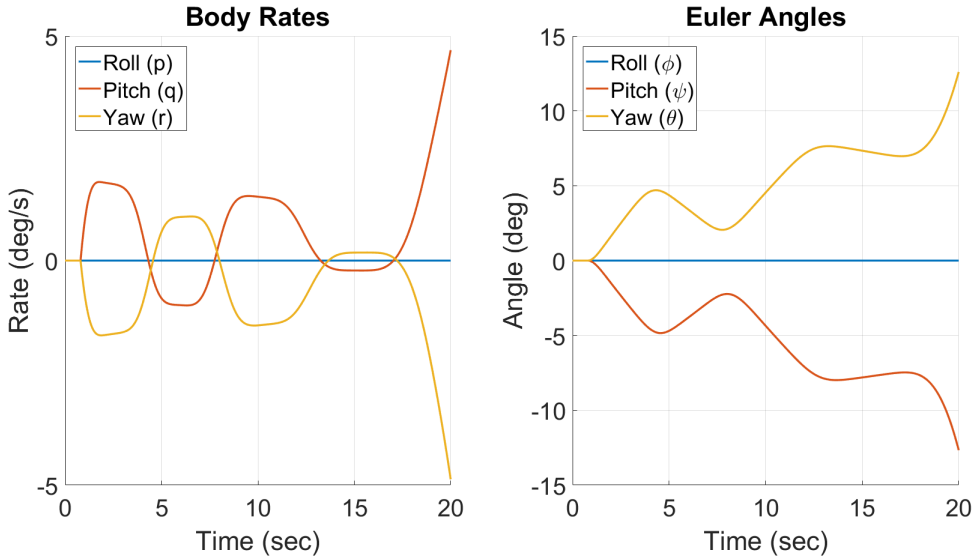

Figure 4.8: Simulated Euler Angles and Body Rate of Concord

The final contributor to the trajectory error discussed here stems from the aero-

dynamic model. Figure 4.8 shows an oscillation in the pitch and yaw axes. Roll was assumed to be 0 in this simulation as no fin cant was modeled. This oscillation occurs as the rocket tends to reduce the angle of attack to 0, but as a full aerodynamic model was not developed in this thesis, there is not enough damping to slow the attitude body rates as it approaches an angle of 0. In the simulation, the angle of attack had a maximum oscillation of approximately 1.5 degrees, but the oscillation was damped as flight continued. This effect can be more clearly seen in figure 5.5.

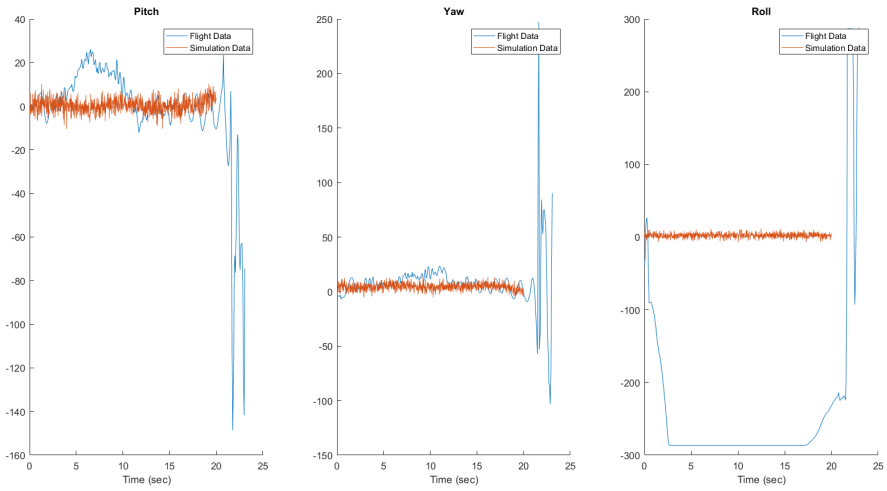

Figure 4.9: Body Rate Comparison Between Gyroscope and Simulation

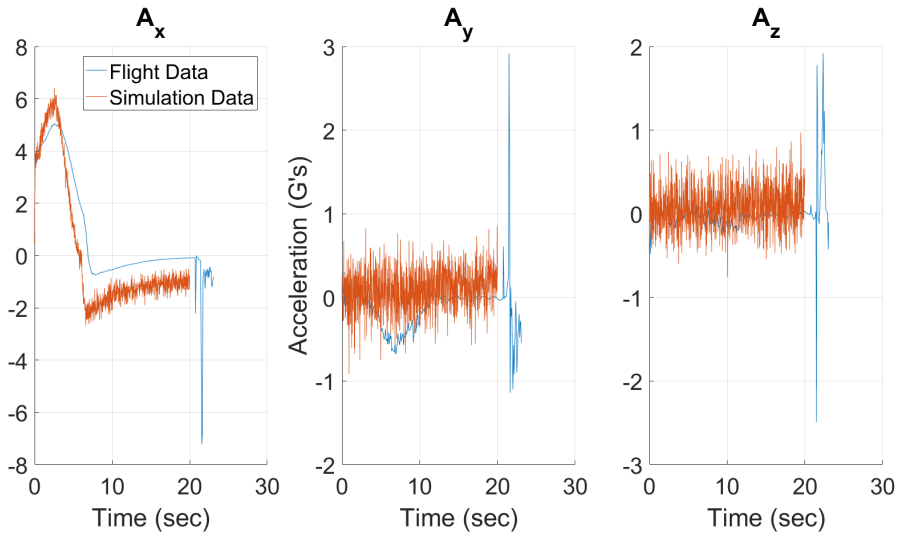

Figure 4.10: Body Rate Comparison Between Accelerometer and Simulation

In looking at the data measured compared to the simulation measurements, the data was not recorded as fast as the simulation time step. There was also not very good agreement in the gyroscope data. This can be explained by the discussion above, as the simulation did not model the correct rotational dynamics. As stated above, roll does not show in the model because the fins were modeled as not canted, but it is clear that there was a significant roll rate in flight as the sensor was saturated at over 250 deg/sec.For the accelerometer, the correct trend appears, but the magnitude in x does not match. This is expected because in the flight rocket, the IMU was not located at the center of gravity. This results in a centripetal acceleration that is measured by the accelerometer, but was not modeled here. The centripetal acceleration is also seen in the y and z axes when the pitch rate and y acceleration have the same shaped curve at the same time.

4.2.2 Flight of the Spirit of Uncle SAM Sounding Rocket

The Spirit of Uncle SAM was launched in April 2017 in the Mojave Desert. Unfortunately, the data recorded from the sensors does not seem to track the flight and is much noisier than what is expected. Raw accelerometer data is shown in figure 4.13. Ignition is seen in both the x and y axes but there is minimal change in the z-axis measurements. Its noise is still on the order of 1 g. Compared to the Concord sounding rocket flight (figure 4.10), which used the same sensor package, this noise is much higher than expected. Because of this noise, the flight was unable to be used to validate the accelerometer, gyroscope, and magnetometer readings. However, the pressure sensor, which is a different chip, recorded data that was much less noisy and followed the correct trend. In this test, the external GPS unit also only had a recording of data every 5 seconds. Because of this, the simulation data and barometer data needed to be shifted over 2 seconds to line up the data.

Looking at figure 4.13, it appears that the simulation tends to over-predict the altitude when compared to the flight data. However, the simulation data matches closely to other simulations. This suggests that the effect is from an assumption between both simulations. Experimenting with the aerodynamic coefficients revealed that increasing C_x by 0.09 throughout the flight causes the simulation to nearly match the flight barometer data. However, based on the discussion in section 4.1, a deviation this far from the actual C_x value seems unlikely. Instead, decreasing the performance of the motor by a straight 5% results in a flight profile that is much closer. As the

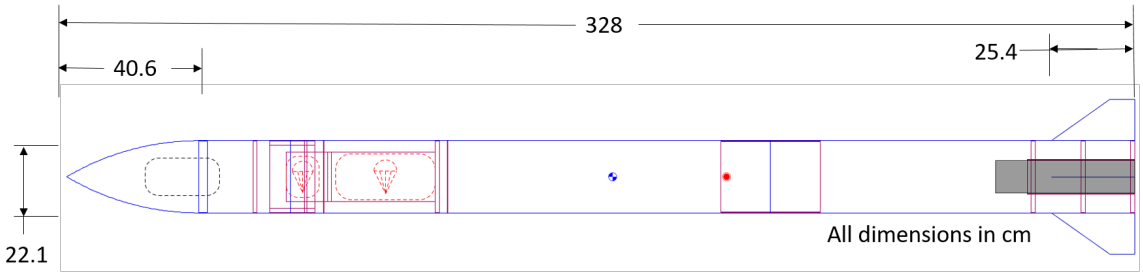

Figure 4.11: Outer Mold Line of Spirit of Uncle SAM Sounding Rocket

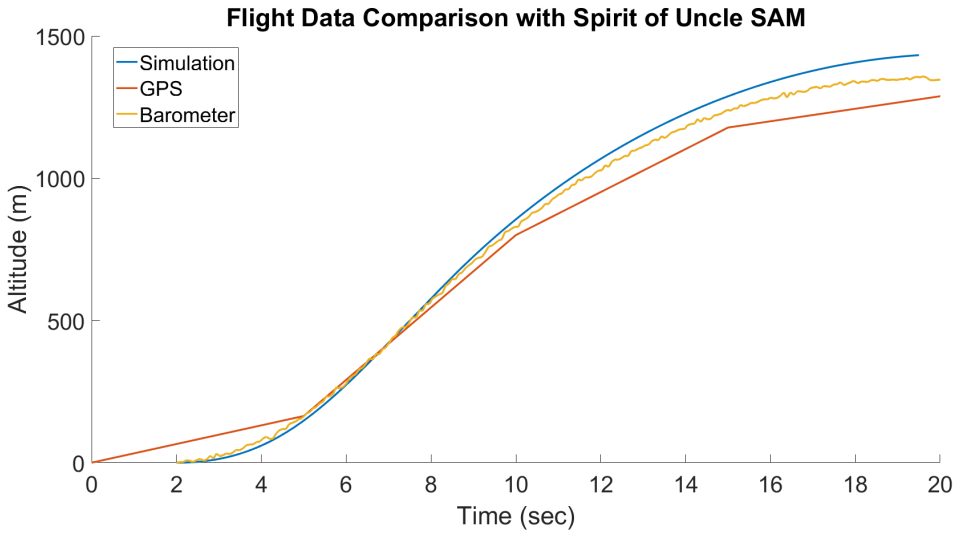

Figure 4.12: Comparison of Simulation and Flight Altitudes

thrust curve used was only an average curve, this seems the most likely large source of error between the flight data and simulation data.

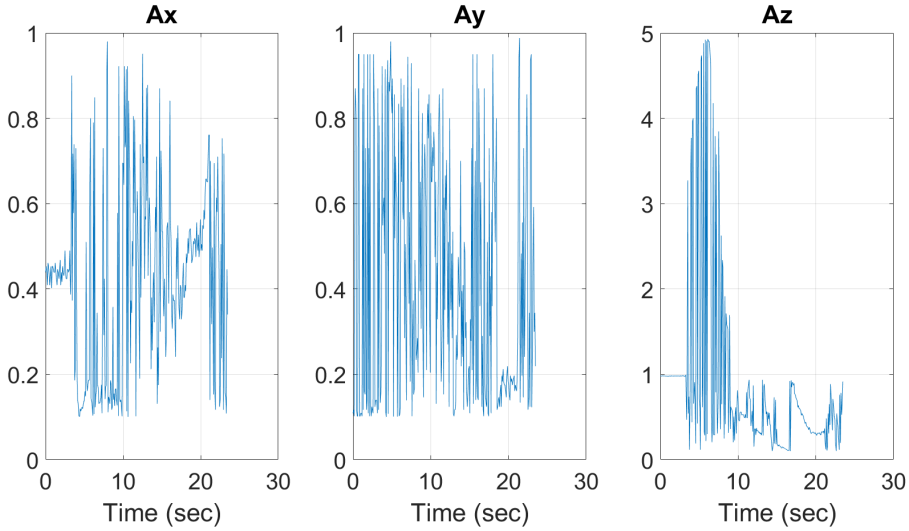

Figure 4.13: Accelerometer Data Recorded During Flight

This chapter presented validation data compared to the simulation output from HSF. Good agreement was shown a possible errors in both the physical models and the simulation models were discussed.

Chapter 5

CONTROLLER DESIGN AND RESULTS

This chapter covers the controller design process and gain matrix. The controlled versus uncontrolled results are presented. MATLAB was used for the determination of the control gain matrix using a linear model.

5.1 Sounding Rocket Physical Model

As part of the design process of the sounding rocket, an investigation of the altitude gain by including a control system on a sounding rocket is performed. The Kronheim rocket, developed by Cal Poly Space Systems, is used to test the simulated control system in both the uncontrolled and controlled results.

The avionics package is similar to the two sounding rockets discussed in section 4.2 and this package is what was simulated in HSF for the Kronheim rocket.

The control surfaces are the fins at the bottom of the rocket. The fins were chosen over canards as the control surface because the rocket was designed to be statically stable, and choosing to not add canards allows the same model to be used as what

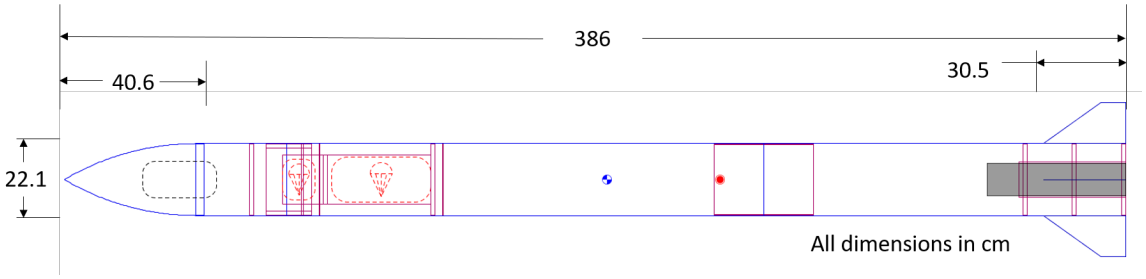

Figure 5.1: Outer Mold Line of Kronheim Rocket

Table 5.1: Inertia of the Kronheim Rocket

Inertia Term	Initial ($kg * m^2$)	Final ($kg * m^2$)
I_{xx}	0.334	0.331
I_{yy}	45.451	42.187
I_{zz}	45.451	42.187

will physically be flown.

The propulsion system on the rocket is a custom-built hybrid motor. At this time, the engine has not yet been tested, and thus a constant thrust profile was used based on the design point. Based on previous motors developed by the group, the thrust curve should be relatively constant. The rocket was assumed to be symmetrical with no off-axis inertia terms and a linear decrease in the inertia during the boost phase of the flight.

5.2 Controller Design

As discussed in section 2.7, the controller was developed using a linearized model of the rotational dynamics. It was linearized using the Jacaobian about 0 degress for the Euler angles and 0 degrees for the body rates. This results in **A**, shown in (5.1). For **B**, it was assumed that the angle of attack was 0. This means that the deflection input becomes the local angle of attack on the fin and C_{N_α} can be used to get the forcing coefficient from the fin. The C_{N_α} from each fin was estimated using the aeroprediction methods from section 2.3. The Mach number selected for this design point was mach 0.6 and the moment of inertia used was the final moment of inertia

of the rocket.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.1}$$

The initial $\mathbf{R}$ and $\mathbf{Q}$ were selected using Bryson's rule (section 2.7). Because the max deflection of the fins was assumed to be 15 degrees, $\mathbf{R}$ was selected as

$$\mathbf{R} = diag([0.0044 \quad 0.0044 \quad 0.0044 \quad 0.0044] * \frac{\pi}{180}) \tag{5.2}$$

$\mathbf{Q}$ was selected so that cosine of the Euler angle would stay above 0.99, or more generally, the flight would reach 99% of its maximum altitude. The associated rates were selected to be small.

$$\mathbf{Q} = diag([0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1]) \tag{5.3}$$

Using these matrices, and the lqrd() function in Matlab, the estimated Kalman matrix is generated using a time step for the discretization of the system of 0.02 seconds.

$$\begin{bmatrix} 0.47 & -0.528 & 0.531 & 0.482 & 1.94 & -1.41 \\ 0.47 & 0.531 & -0.528 & 0.484 & -1.79 & 1.56 \\ 0.47 & -0.528 & 0.531 & 0.482 & 1.94 & -1.41 \\ 0.47 & 0.531 & -0.528 & 0.484 & -1.79 & 1.56 \end{bmatrix} \tag{5.4}$$

5.2.1 Performance

Immediately it is seen that the controller has a positive effect of the altitude of the control system. The controller effect can also be seen in the plot of Euler angles and

body rates (figure 5.5). As mentioned in section 4.2.1, the simulation tends to under-damp the oscillation around the zero degree angle of attack. which is also observed here. As expected, the controller acts as a damping term on this oscillation [32], but the stable design of the rocket means that the control fins do not have enough control authority to damp this oscillation out completely. It also does not have the authority to maintain a perfect zero degree Euler angle. It is also observed that the cross range distance and inertial velocity are decreased in the controlled case. This is expected as the acceleration provided by the thrust is contributing to the altitude and vertical velocity.

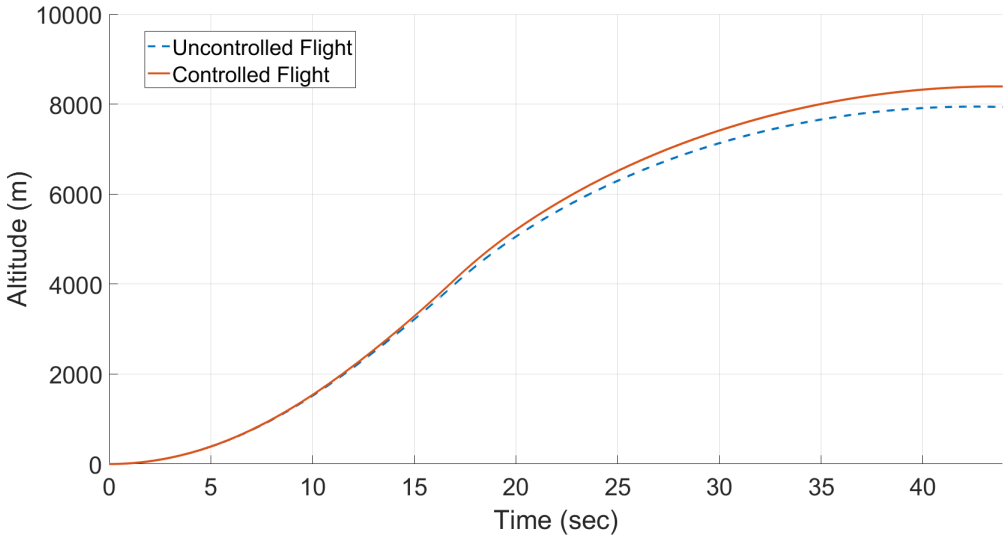

Figure 5.2: Altitude Comparison of Controlled and Uncontrolled Flight

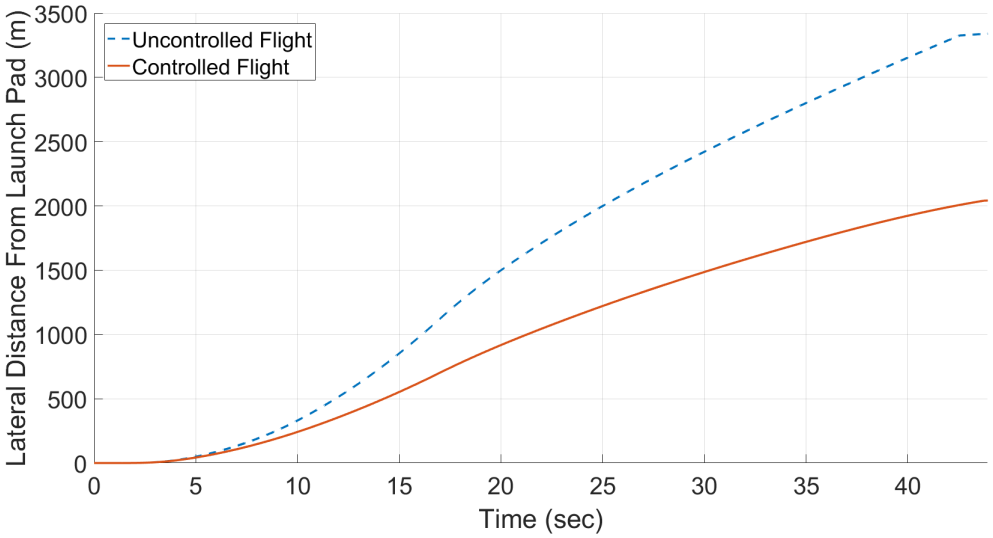

Figure 5.3: Lateral Ground Comparison Comparison of Controlled and Uncontrolled Flight

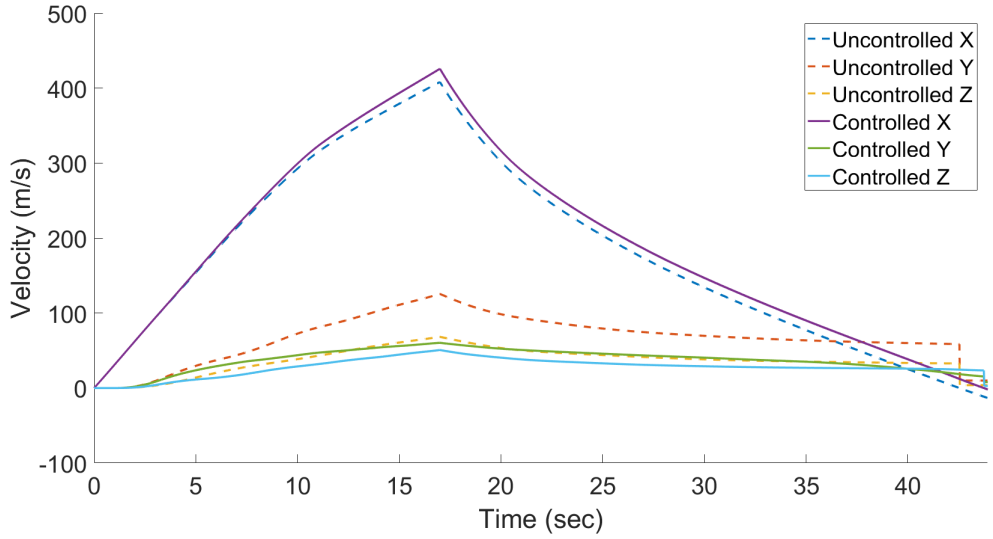

Figure 5.4: 3 Axis Velocity Comparison of Controlled and Uncontrolled Flight

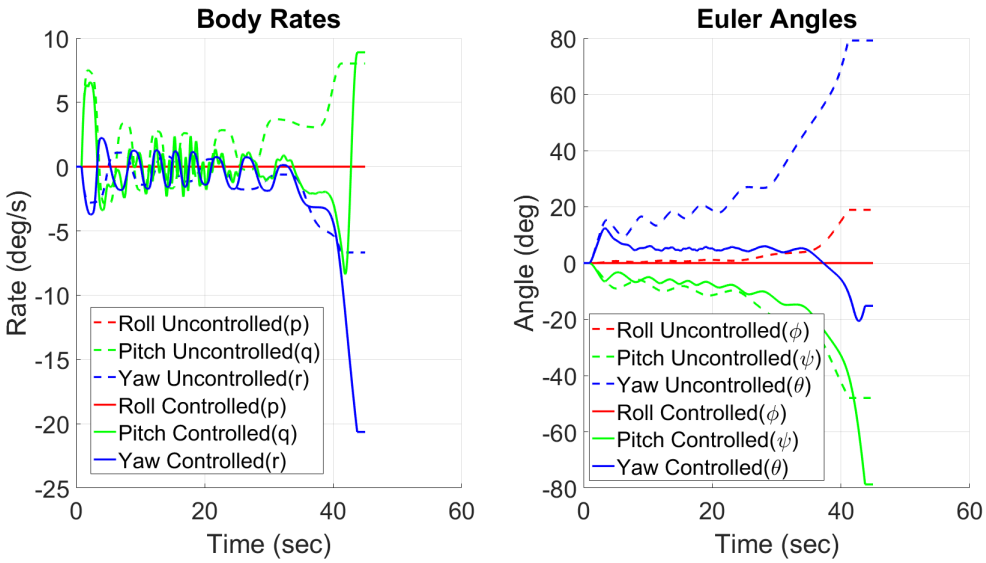

Figure 5.5: Angles Comparison of Controlled and Uncontrolled Flight

Chapter 6

FUTURE WORK AND CONCLUSION

6.1 Future Work

6.1.1 Improved Aerodynamic Model

As noted in section 4.2.1, the aerodynamics model used in this thesis had difficulties with oscillations around a 0 angle of attack. To remedy this issue two sections of the simulation must be updated, the equations of motion and the aeroprediction code. If just the EOMs are updated to account for more coefficients and these coefficients are provided by external files, the simulation will become more accurate. After the EOMs have been modified to accept more than the current aerodynamic coefficient, the aeroprediction code can be expanded to increase the fidelity of the model.

Along with these changes to the code to increase the fidelity of the simulation, aerodynamic unit tests to validate the predicted coefficients should be implemented. Currently, only one set of data is included in these tests, and while the simulation model agrees well with this data set, additional data sets will improve confidence in the model.

6.1.2 Graphical User Interface (GUI)

The version of HSF before the rewrite by Yost included a GUI to aid the designer in the modeling process. Yost proposed that this task be completed for the framework [8], and this recommendation is restated here. A GUI will reduce the learning curve of new users and allow the designer to focus more on the modeling and less on the code semantics of Python and XML.

6.1.3 Expansion of Testing

In this thesis, a corresponding unit test is developed for all of the subsystems which allows extraction of data at specific conditions and makes the code resilient to bugs occurring when changes are made. To generate the aerodynamic verification plots, unit tests were written to input the wind tunnel test conditions and extract out the model resulting aerodynamic coefficients. While the unit tests for code unique to this thesis are written, an expansion of the unit test code to the rest of the framework should be included.

In addition to the unit tests for each of the smaller functions, a unit test suite should be added to the overall framework utilizing the Aeolus test case. This test suite will capture if any smaller changes or additions to the code change the final output. Once this test suite is defined, an extensive review should be conducted to verify that the outputs are correct. Additionally, if the tests begin to fail, care should be taken before modifying the expected test results.

6.1.4 Expansion of Base Framework

This thesis contributed some models to the overall framework, but the base framework should be continued to be expanded with additional environment models and generic subsystems. With this thesis, models for use mainly in the lower atmosphere were implemented, but models should be created for the upper atmosphere, Earth orbits, and interplanetary trajectories. These additions will allow the designer to focus on the system and developing the subsystem models.

6.2 Conclusion

The HSF framework was developed to be modular and flexible. It was originally designed around scheduling tasks on a satellite and optimizing the value of these tasks in the system. However, in this thesis, an expansion of the framework capabilities is developed to re-implement dynamic motion types that existed in early versions of the framework. To test this expanded capability of the framework a sounding rocket model was developed to predict performance and a control system was implemented to improve the maximum altitude reached by the rocket. To fully develop the model and make the framework self-contained, a first order method was implemented to predict the aerodynamic coefficients. The sounding rocket model was compared to two different experimental flights, and the aeroprediction model was compared to experimental data. Both of these comparisons showed relatively good agreement between the simulated and experimental data.

BIBLIOGRAPHY

[1] NASA. *Black Brant XII Capability Catalog.* Accessed:2017-06-06.

[2] NASA. *Sounding Rocket Overview.* Accessed:2017-02-07.

[3] Record-breaking flight: Usc students rocket shoots up, up and away. https://news.usc.edu/118405/up-up-and-away-usc-rocket-propulsion-lab-breaks-a-record/. Accessed:2017-06-06.

[4] Bryan Tong Minh. Real-time Position and Attitude Determination of the Stratos II Sounding Rocket. Master's thesis, Delft University of Technology, 2012.

[5] Sampo Niskanen. Development of an Open Source Model Rocket Simulation Software. 2009.

[6] Seffat Mohammad Chowdhury. Design and Performance Simulation of a Hybrid Sounding Rocket. 2012.

[7] Cory O'Connor and Eric A Mehiel. Horizon: A system modeling and simulation framework for systems engineering utility analysis: A thesis. Master's thesis, California Polytechnic State University, San Luis Obispo, 2007.

[8] Morgan Yost. An iteration on the horizon simulation framework to include .net and python scripting. Master's thesis, California Polytechnic State University, San Luis Obispo, 2016.

[9] Zhenhua Morton Li. Modeling and simulation of autonomous thermal soaring with horizon simulation framework: A thesis. Master's thesis, California Polytechnic State University, San Luis Obispo, 2010.

[10] R. Shishko and R. Aster. NASA systems engineering handbook. *NASA Special Publication*, 6105, 1995.

[11] Margaret L. Loper. *Modeling and simulation in the systems engineering life cycle: core concepts and accompanying lectures*. Springer, 2015.

[12] Juan M Ibarra-zannatha, Rafael Cisneros Limón, Wilder Eduardo Castellanos Hernandez, Ingeniero Electronico, Jeff A Estefan, Mohammad S Obaidat, Noureddine A Boudriga, W. M. P. van der Aalst, M. Voorhoeve, Ernest H Page, Richard E Nance, James D Arthur, Hugo García de la Torre, Gonzalo Herradón Berzal, D. Rafael Herradón Díez, Cell Breathing, Simuladores Software, Umts Protocol, Simulator Umtsprosim, Iman Gholami, W.H. Tranter, and K.L. Kosbar. Survey of Model-Based Systems Engineering (MBSE) Methodologies 2 . Differentiating Methodologies from Processes , Methods , and Lifecycle Models. *Environment*, 32(7):397–438, 1994.

[13] US Standard Atmosphere. National oceanic and atmospheric administration. *Aeronautics and Space Administration, United States Air Force, Washington, DC*, 1976.

[14] Douglas P. Drob, John Emmert, and Manbharat Dhadly. Horizontal wind model (HWM). Technical report, US Naval Research Laboratory.

[15] Douglas P. Drob, John T. Emmert, John W. Meriwether, Jonathan J. Makela, Eelco Doornbos, Mark Conde, Gonzalo Hernandez, John Noto, Katherine A. Zawdie, Sarah E. McDonald, Joe D. Huba, and Jeff H. Klenzing. An update to the horizontal wind model (hwm): The quiet time thermosphere. *Earth and Space Science*, 2(7):301–319, 2015. 2014EA000089.

[16] National Centers for Environmental Prediction. Gfs. http://www.emc.ncep.noaa.gov/index.php?branch=GFS. Accessed:2017-05-24.

[17] Arnaud Chulliat, Susan Macmillan, Patrick Alken, Ciaran Beggan, Manoj Nair, Brian Hamilton, Adam Woods, Victoria Ridley, Stefan Maus, and Alan Thomson. The us/uk world magnetic model for 2015-2020. 2015.

[18] Anton H De Ruiter, Christopher Damaren, and James R Forbes. *Spacecraft dynamics and control: an introduction*. John Wiley & Sons, 2012.

[19] Swapnil Pramod Kanade and Abraham T Mathew. 2 dof h-infinity loop shaping robust control for rocket attitude stabilization. *International Journal of Aerospace Sciences*, 2(3):71–91, 2013.

[20] Douglass Auld. Drag Coefficient Prediction. 2013.

[21] J.H. Wickman and CP Technologies. *How to Make Amateur Rockets*. CP Technologies, 1997.

[22] Robert Galejs. Wind instability, what barrowman left out. *Advanved Rocketry Group of Switzerland*, 2009.

[23] Franklin Diederich. A Plan-Form Parameter for Correlating Certain Aerodynamic Characteristics of Swept Wings. *Naca TN-2335*, 1951.

[24] Judith A Barrowman. The Theoretical Prediction of the Center of Pressure. 1996.

[25] JS Barrowman. Fin-a computer program for calculating the aerodynamic characteristics of fins at supersonic speeds. 1966.

[26] Philip N Jenkins. Missile Dynamics Equvations for Guidance La-L. 1984.

[27] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[28] João Hespanha. *Topics in Undergraduate Control Systems Design*. 2014.

[29] Paul Zarchan. *Progress In Astronautics and Aeronautics: Fundamentals of Kalman Filtering: A Practical Approach*, volume 208. Aiaa, 2005.

[30] Roger G Hart. Flight investigation at mach numbers from 0.8 to 1.5 to determine the effects of nose bluntness on the total drag of two fin-stabilized bodies of revolution. 1955.

[31] James C Ferris. Static stability investigation of a single-stage sounding rocket at mach numbers from 0.60 to 1.20. 1967.

[32] Michael J Hemsch and Jack N Nielsen. Tactical missile aerodynamics. 1986.

[33] Lal Bahadur Prasad, Barjeev Tyagi, and Hari Om Gupta. Optimal control of nonlinear inverted pendulum system using PID controller and LQR: Performance analysis without and with disturbance input. *International Journal of Automation and Computing*, 11(6):661–670, 2014.

[34] David Titterton and John L Weston. *Strapdown inertial navigation technology*, volume 17. IET, 2004.

APPENDIX

COMPLETE KALMAN FILTER FORMULATION

$$\bar{x} = \begin{bmatrix} \mathbf{P} \\ \mathbf{v} \\ \mathbf{E} \\ \omega \\ \mathbf{C_T} \\ \mathbf{C_R} \end{bmatrix} \quad \text{where} \quad \begin{aligned} \mathbf{P} &= \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \quad \mathbf{E} = \begin{bmatrix} \psi \\ \theta \\ \phi \end{bmatrix} \\ \omega &= \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad \mathbf{C_T} = \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \quad \mathbf{C_R} = \begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix} \end{aligned} \tag{6.1}$$

$$\dot{\bar{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{C_x} \\ \dot{C_y} \\ \dot{C_z} \\ \dot{C_L} \\ \dot{C_M} \\ \dot{C_N} \end{bmatrix} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ -g\cos\psi\cos\theta + \frac{T}{m} - \frac{Q_p A_{ref}}{m} C_x \\ g * (\cos\phi\sin\psi - \cos\psi\cos\phi\sin\theta) - \frac{Q_p A_{ref}}{m} C_y \\ -g * (\sin\phi\sin\psi + \cos\psi\cos\phi\sin\theta) - \frac{Q_p A_{ref}}{m} C_z \\ \frac{q*\sin\phi + r\cos\phi}{\cos\theta} \\ q * \cos\phi + r\sin\phi \\ p + \dot{\psi}\sin\theta \\ \frac{\frac{Q_p A_{ref}}{m} C_L - (I_{zz} - I_{yy})qr}{I_{xx}} \\ \frac{\frac{Q_p A_{ref}}{m} C_M - (I_{xx} - I_{zz})pr}{I_{yy}} \\ \frac{\frac{Q_p A_{ref}}{m} C_N - (I_{yy} - I_{xx})pq}{I_{zz}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.2}$$

$$
\mathbf{f}_{11} = \begin{bmatrix} \frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial y} & \frac{\partial \dot{x}}{\partial z} \\ \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial z} \\ \frac{\partial \dot{z}}{\partial x} & \frac{\partial \dot{z}}{\partial y} & \frac{\partial \dot{z}}{\partial z} \end{bmatrix} \mathbf{f}_{12} = \begin{bmatrix} \frac{\partial \dot{x}}{\partial y} & \frac{\partial \dot{x}}{\partial z} & \frac{\partial \dot{x}}{\partial \dot{x}} \\ \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial z} & \frac{\partial \dot{y}}{\partial \dot{x}} \\ \frac{\partial \dot{z}}{\partial y} & \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial \dot{x}} \end{bmatrix} \mathbf{f}_{13} = \begin{bmatrix} \frac{\partial \dot{x}}{\partial z} & \frac{\partial \dot{x}}{\partial \dot{x}} & \frac{\partial \dot{x}}{\partial \dot{y}} \\ \frac{\partial \dot{y}}{\partial z} & \frac{\partial \dot{y}}{\partial \dot{x}} & \frac{\partial \dot{y}}{\partial \dot{y}} \\ \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial \dot{x}} & \frac{\partial \dot{z}}{\partial \dot{y}} \end{bmatrix}
$$

$$
\mathbf{f}_{14} = \begin{bmatrix} \frac{\partial \dot{x}}{\partial \dot{x}} & \frac{\partial \dot{x}}{\partial \dot{y}} & \frac{\partial \dot{x}}{\partial \dot{z}} \\ \frac{\partial \dot{y}}{\partial \dot{x}} & \frac{\partial \dot{y}}{\partial \dot{y}} & \frac{\partial \dot{y}}{\partial \dot{z}} \\ \frac{\partial \dot{z}}{\partial \dot{x}} & \frac{\partial \dot{z}}{\partial \dot{y}} & \frac{\partial \dot{z}}{\partial \dot{z}} \end{bmatrix} \mathbf{f}_{15} = \begin{bmatrix} \frac{\partial \dot{x}}{\partial \dot{y}} & \frac{\partial \dot{x}}{\partial \dot{z}} & \frac{\partial \dot{x}}{\partial \phi} \\ \frac{\partial \dot{y}}{\partial \dot{y}} & \frac{\partial \dot{y}}{\partial \dot{z}} & \frac{\partial \dot{y}}{\partial \phi} \\ \frac{\partial \dot{z}}{\partial \dot{y}} & \frac{\partial \dot{z}}{\partial \dot{z}} & \frac{\partial \dot{z}}{\partial \phi} \end{bmatrix} \mathbf{f}_{16} = \begin{bmatrix} \frac{\partial \dot{x}}{\partial \dot{z}} & \frac{\partial \dot{x}}{\partial \phi} & \frac{\partial \dot{x}}{\partial \psi} \\ \frac{\partial \dot{y}}{\partial \dot{z}} & \frac{\partial \dot{y}}{\partial \phi} & \frac{\partial \dot{y}}{\partial \psi} \\ \frac{\partial \dot{z}}{\partial \dot{z}} & \frac{\partial \dot{z}}{\partial \phi} & \frac{\partial \dot{z}}{\partial \psi} \end{bmatrix}
$$

$$
\mathbf{f}_{21} = \begin{bmatrix} \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial z} \\ \frac{\partial \dot{z}}{\partial x} & \frac{\partial \dot{z}}{\partial y} & \frac{\partial \dot{z}}{\partial z} \\ \frac{\partial \dot{\dot{x}}}{\partial x} & \frac{\partial \dot{\dot{x}}}{\partial y} & \frac{\partial \dot{\dot{x}}}{\partial z} \end{bmatrix} \mathbf{f}_{22} = \begin{bmatrix} \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial z} & \frac{\partial \dot{y}}{\partial \dot{x}} \\ \frac{\partial \dot{z}}{\partial y} & \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial \dot{x}} \\ \frac{\partial \dot{\dot{x}}}{\partial y} & \frac{\partial \dot{\dot{x}}}{\partial z} & \frac{\partial \dot{\dot{x}}}{\partial \dot{x}} \end{bmatrix} \mathbf{f}_{23} = \begin{bmatrix} \frac{\partial \dot{y}}{\partial z} & \frac{\partial \dot{y}}{\partial \dot{x}} & \frac{\partial \dot{y}}{\partial \dot{y}} \\ \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial \dot{x}} & \frac{\partial \dot{z}}{\partial \dot{y}} \\ \frac{\partial \dot{\dot{x}}}{\partial z} & \frac{\partial \dot{\dot{x}}}{\partial \dot{x}} & \frac{\partial \dot{\dot{x}}}{\partial \dot{y}} \end{bmatrix}
$$

$$
\mathbf{f}_{24} = \begin{bmatrix} \frac{\partial \dot{y}}{\partial \dot{x}} & \frac{\partial \dot{y}}{\partial \dot{y}} & \frac{\partial \dot{y}}{\partial \dot{z}} \\ \frac{\partial \dot{z}}{\partial \dot{x}} & \frac{\partial \dot{z}}{\partial \dot{y}} & \frac{\partial \dot{z}}{\partial \dot{z}} \\ \frac{\partial \dot{\dot{x}}}{\partial \dot{x}} & \frac{\partial \dot{\dot{x}}}{\partial \dot{y}} & \frac{\partial \dot{\dot{x}}}{\partial \dot{z}} \end{bmatrix} \mathbf{f}_{25} = \begin{bmatrix} \frac{\partial \dot{y}}{\partial \dot{y}} & \frac{\partial \dot{y}}{\partial \dot{z}} & \frac{\partial \dot{y}}{\partial \phi} \\ \frac{\partial \dot{z}}{\partial \dot{y}} & \frac{\partial \dot{z}}{\partial \dot{z}} & \frac{\partial \dot{z}}{\partial \phi} \\ \frac{\partial \dot{\dot{x}}}{\partial \dot{y}} & \frac{\partial \dot{\dot{x}}}{\partial \dot{z}} & \frac{\partial \dot{\dot{x}}}{\partial \phi} \end{bmatrix} \mathbf{f}_{26} = \begin{bmatrix} \frac{\partial \dot{y}}{\partial \dot{z}} & \frac{\partial \dot{y}}{\partial \phi} & \frac{\partial \dot{y}}{\partial \psi} \\ \frac{\partial \dot{z}}{\partial \dot{z}} & \frac{\partial \dot{z}}{\partial \phi} & \frac{\partial \dot{z}}{\partial \psi} \\ \frac{\partial \dot{\dot{x}}}{\partial \dot{z}} & \frac{\partial \dot{\dot{x}}}{\partial \phi} & \frac{\partial \dot{\dot{x}}}{\partial \psi} \end{bmatrix}
$$

$$
\mathbf{f}_{31} = \begin{bmatrix} \frac{\partial \dot{z}}{\partial x} & \frac{\partial \dot{z}}{\partial y} & \frac{\partial \dot{z}}{\partial z} \\ \frac{\partial \dot{\dot{x}}}{\partial x} & \frac{\partial \dot{\dot{x}}}{\partial y} & \frac{\partial \dot{\dot{x}}}{\partial z} \\ \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial z} \end{bmatrix} \mathbf{f}_{32} = \begin{bmatrix} \frac{\partial \dot{z}}{\partial y} & \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial \dot{x}} \\ \frac{\partial \dot{\dot{x}}}{\partial y} & \frac{\partial \dot{\dot{x}}}{\partial z} & \frac{\partial \dot{\dot{x}}}{\partial \dot{x}} \\ \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial z} & \frac{\partial \dot{y}}{\partial \dot{x}} \end{bmatrix} \mathbf{f}_{33} = \begin{bmatrix} \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial \dot{x}} & \frac{\partial \dot{z}}{\partial \dot{y}} \\ \frac{\partial \dot{\dot{x}}}{\partial z} & \frac{\partial \dot{\dot{x}}}{\partial \dot{x}} & \frac{\partial \dot{\dot{x}}}{\partial \dot{y}} \\ \frac{\partial \dot{y}}{\partial z} & \frac{\partial \dot{y}}{\partial \dot{x}} & \frac{\partial \dot{y}}{\partial \dot{y}} \end{bmatrix}
$$

$$
\mathbf{f}_{34} = \begin{bmatrix} \frac{\partial \dot{z}}{\partial \dot{x}} & \frac{\partial \dot{z}}{\partial \dot{y}} & \frac{\partial \dot{z}}{\partial \dot{z}} \\ \frac{\partial \dot{\dot{x}}}{\partial \dot{x}} & \frac{\partial \dot{\dot{x}}}{\partial \dot{y}} & \frac{\partial \dot{\dot{x}}}{\partial \dot{z}} \\ \frac{\partial \dot{y}}{\partial \dot{x}} & \frac{\partial \dot{y}}{\partial \dot{y}} & \frac{\partial \dot{y}}{\partial \dot{z}} \end{bmatrix} \mathbf{f}_{35} = \begin{bmatrix} \frac{\partial \dot{z}}{\partial \dot{y}} & \frac{\partial \dot{z}}{\partial \dot{z}} & \frac{\partial \dot{z}}{\partial \phi} \\ \frac{\partial \dot{\dot{x}}}{\partial \dot{y}} & \frac{\partial \dot{\dot{x}}}{\partial \dot{z}} & \frac{\partial \dot{\dot{x}}}{\partial \phi} \\ \frac{\partial \dot{y}}{\partial \dot{y}} & \frac{\partial \dot{y}}{\partial \dot{z}} & \frac{\partial \dot{y}}{\partial \phi} \end{bmatrix} \mathbf{f}_{36} = \begin{bmatrix} \frac{\partial \dot{z}}{\partial \dot{z}} & \frac{\partial \dot{z}}{\partial \phi} & \frac{\partial \dot{z}}{\partial \psi} \\ \frac{\partial \dot{\dot{x}}}{\partial \dot{z}} & \frac{\partial \dot{\dot{x}}}{\partial \phi} & \frac{\partial \dot{\dot{x}}}{\partial \psi} \\ \frac{\partial \dot{y}}{\partial \dot{z}} & \frac{\partial \dot{y}}{\partial \phi} & \frac{\partial \dot{y}}{\partial \psi} \end{bmatrix}
$$

$$
\mathbf{f}_{41} = \begin{bmatrix} \frac{\partial \dot{\dot{x}}}{\partial x} & \frac{\partial \dot{\dot{x}}}{\partial y} & \frac{\partial \dot{\dot{x}}}{\partial z} \\ \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial z} \\ \frac{\partial \dot{z}}{\partial x} & \frac{\partial \dot{z}}{\partial y} & \frac{\partial \dot{z}}{\partial z} \end{bmatrix} \mathbf{f}_{42} = \begin{bmatrix} \frac{\partial \dot{\dot{x}}}{\partial y} & \frac{\partial \dot{\dot{x}}}{\partial z} & \frac{\partial \dot{\dot{x}}}{\partial \dot{x}} \\ \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial z} & \frac{\partial \dot{y}}{\partial \dot{x}} \\ \frac{\partial \dot{z}}{\partial y} & \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial \dot{x}} \end{bmatrix} \mathbf{f}_{43} = \begin{bmatrix} \frac{\partial \dot{\dot{x}}}{\partial z} & \frac{\partial \dot{\dot{x}}}{\partial \dot{x}} & \frac{\partial \dot{\dot{x}}}{\partial \dot{y}} \\ \frac{\partial \dot{y}}{\partial z} & \frac{\partial \dot{y}}{\partial \dot{x}} & \frac{\partial \dot{y}}{\partial \dot{y}} \\ \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial \dot{x}} & \frac{\partial \dot{z}}{\partial \dot{y}} \end{bmatrix}
$$

$$
\mathbf{f}_{44} = \begin{bmatrix} \frac{\partial \dot{\dot{x}}}{\partial \dot{x}} & \frac{\partial \dot{\dot{x}}}{\partial \dot{y}} & \frac{\partial \dot{\dot{x}}}{\partial \dot{z}} \\ \frac{\partial \dot{y}}{\partial \dot{x}} & \frac{\partial \dot{y}}{\partial \dot{y}} & \frac{\partial \dot{y}}{\partial \dot{z}} \\ \frac{\partial \dot{z}}{\partial \dot{x}} & \frac{\partial \dot{z}}{\partial \dot{y}} & \frac{\partial \dot{z}}{\partial \dot{z}} \end{bmatrix} \mathbf{f}_{45} = \begin{bmatrix} \frac{\partial \dot{\dot{x}}}{\partial \dot{y}} & \frac{\partial \dot{\dot{x}}}{\partial \dot{z}} & \frac{\partial \dot{\dot{x}}}{\partial \phi} \\ \frac{\partial \dot{y}}{\partial \dot{y}} & \frac{\partial \dot{y}}{\partial \dot{z}} & \frac{\partial \dot{y}}{\partial \phi} \\ \frac{\partial \dot{z}}{\partial \dot{y}} & \frac{\partial \dot{z}}{\partial \dot{z}} & \frac{\partial \dot{z}}{\partial \phi} \end{bmatrix} \mathbf{f}_{46} = \begin{bmatrix} \frac{\partial \dot{\dot{x}}}{\partial \dot{z}} & \frac{\partial \dot{\dot{x}}}{\partial \phi} & \frac{\partial \dot{\dot{x}}}{\partial \psi} \\ \frac{\partial \dot{y}}{\partial \dot{z}} & \frac{\partial \dot{y}}{\partial \phi} & \frac{\partial \dot{y}}{\partial \psi} \\ \frac{\partial \dot{z}}{\partial \dot{z}} & \frac{\partial \dot{z}}{\partial \phi} & \frac{\partial \dot{z}}{\partial \psi} \end{bmatrix}
$$

(6.3)

$$\mathbf{f}_{51} = \begin{bmatrix} \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial z} \\ \frac{\partial \dot{z}}{\partial x} & \frac{\partial \dot{z}}{\partial y} & \frac{\partial \dot{z}}{\partial z} \\ \frac{\partial \dot{\phi}}{\partial x} & \frac{\partial \dot{\phi}}{\partial y} & \frac{\partial \dot{\phi}}{\partial z} \end{bmatrix} \mathbf{f}_{52} = \begin{bmatrix} \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial z} & \frac{\partial \dot{y}}{\partial \dot{x}} \\ \frac{\partial \dot{z}}{\partial y} & \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial \dot{x}} \\ \frac{\partial \dot{\phi}}{\partial y} & \frac{\partial \dot{\phi}}{\partial z} & \frac{\partial \dot{\phi}}{\partial \dot{x}} \end{bmatrix} \mathbf{f}_{53} = \begin{bmatrix} \frac{\partial \dot{y}}{\partial z} & \frac{\partial \dot{y}}{\partial \dot{x}} & \frac{\partial \dot{y}}{\partial \dot{y}} \\ \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial \dot{x}} & \frac{\partial \dot{z}}{\partial \dot{y}} \\ \frac{\partial \dot{\phi}}{\partial z} & \frac{\partial \dot{\phi}}{\partial \dot{x}} & \frac{\partial \dot{\phi}}{\partial \dot{y}} \end{bmatrix}$$

$$\mathbf{f}_{54} = \begin{bmatrix} \frac{\partial \dot{y}}{\partial \dot{x}} & \frac{\partial \dot{y}}{\partial \dot{y}} & \frac{\partial \dot{y}}{\partial \dot{z}} \\ \frac{\partial \dot{z}}{\partial \dot{x}} & \frac{\partial \dot{z}}{\partial \dot{y}} & \frac{\partial \dot{z}}{\partial \dot{z}} \\ \frac{\partial \dot{\phi}}{\partial \dot{x}} & \frac{\partial \dot{\phi}}{\partial \dot{y}} & \frac{\partial \dot{\phi}}{\partial \dot{z}} \end{bmatrix} \mathbf{f}_{55} = \begin{bmatrix} \frac{\partial \dot{y}}{\partial \dot{y}} & \frac{\partial \dot{y}}{\partial \dot{z}} & \frac{\partial \dot{y}}{\partial \phi} \\ \frac{\partial \dot{z}}{\partial \dot{y}} \frac{\partial \dot{z}}{\partial \dot{z}} & \frac{\partial \dot{z}}{\partial \phi} & \\ \frac{\partial \dot{\phi}}{\partial \dot{y}} & \frac{\partial \dot{\phi}}{\partial \dot{z}} & \frac{\partial \dot{\phi}}{\partial \phi} \end{bmatrix} \mathbf{f}_{56} = \begin{bmatrix} \frac{\partial \dot{y}}{\partial \dot{z}} & \frac{\partial \dot{y}}{\partial \phi} & \frac{\partial \dot{y}}{\partial \psi} \\ \frac{\partial \dot{z}}{\partial \dot{z}} & \frac{\partial \dot{z}}{\partial \phi} & \frac{\partial \dot{z}}{\partial \psi} \\ \frac{\partial \dot{\phi}}{\partial \dot{z}} & \frac{\partial \dot{\phi}}{\partial \phi} & \frac{\partial \dot{\phi}}{\partial \psi} \end{bmatrix}$$

$$\mathbf{f}_{61} = \begin{bmatrix} \frac{\partial \dot{z}}{\partial x} & \frac{\partial \dot{z}}{\partial y} & \frac{\partial \dot{z}}{\partial z} \\ \frac{\partial \dot{\phi}}{\partial x} & \frac{\partial \dot{\phi}}{\partial y} & \frac{\partial \dot{\phi}}{\partial z} \\ \frac{\partial \dot{\psi}}{\partial x} & \frac{\partial \dot{\psi}}{\partial y} & \frac{\partial \dot{\psi}}{\partial z} \end{bmatrix} \mathbf{f}_{62} = \begin{bmatrix} \frac{\partial \dot{z}}{\partial y} & \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial \dot{x}} \\ \frac{\partial \dot{\phi}}{\partial y} & \frac{\partial \dot{\phi}}{\partial z} & \frac{\partial \dot{\phi}}{\partial \dot{x}} \\ \frac{\partial \dot{\psi}}{\partial y} & \frac{\partial \dot{\psi}}{\partial z} & \frac{\partial \dot{\psi}}{\partial \dot{x}} \end{bmatrix} \mathbf{f}_{63} = \begin{bmatrix} \frac{\partial \dot{z}}{\partial z} & \frac{\partial \dot{z}}{\partial \dot{x}} & \frac{\partial \dot{z}}{\partial \dot{y}} \\ \frac{\partial \dot{\phi}}{\partial z} & \frac{\partial \dot{\phi}}{\partial \dot{x}} & \frac{\partial \dot{\phi}}{\partial \dot{y}} \\ \frac{\partial \dot{\psi}}{\partial z} & \frac{\partial \dot{\psi}}{\partial \dot{x}} & \frac{\partial \dot{\psi}}{\partial \dot{y}} \end{bmatrix}$$

$$\mathbf{f}_{64} = \begin{bmatrix} \frac{\partial \dot{z}}{\partial \dot{x}} & \frac{\partial \dot{z}}{\partial \dot{y}} & \frac{\partial \dot{z}}{\partial \dot{z}} \\ \frac{\partial \dot{\phi}}{\partial \dot{x}} & \frac{\partial \dot{\phi}}{\partial \dot{y}} & \frac{\partial \dot{\phi}}{\partial \dot{z}} \\ \frac{\partial \dot{\psi}}{\partial \dot{x}} & \frac{\partial \dot{\psi}}{\partial \dot{y}} & \frac{\partial \dot{\psi}}{\partial \dot{z}} \end{bmatrix} \mathbf{f}_{65} = \begin{bmatrix} \frac{\partial \dot{z}}{\partial \dot{y}} & \frac{\partial \dot{z}}{\partial \dot{z}} & \frac{\partial \dot{z}}{\partial \phi} \\ \frac{\partial \dot{\phi}}{\partial \dot{y}} & \frac{\partial \dot{\phi}}{\partial \dot{z}} & \frac{\partial \dot{\phi}}{\partial \phi} \\ \frac{\partial \dot{\psi}}{\partial \dot{y}} & \frac{\partial \dot{\psi}}{\partial \dot{z}} & \frac{\partial \dot{\psi}}{\partial \phi} \end{bmatrix} \mathbf{f}_{66} = \begin{bmatrix} \frac{\partial \dot{z}}{\partial \dot{z}} & \frac{\partial \dot{z}}{\partial \phi} & \frac{\partial \dot{z}}{\partial \psi} \\ \frac{\partial \dot{\phi}}{\partial \dot{z}} & \frac{\partial \dot{\phi}}{\partial \phi} & \frac{\partial \dot{\phi}}{\partial \psi} \\ \frac{\partial \dot{\psi}}{\partial \dot{z}} & \frac{\partial \dot{\psi}}{\partial \phi} & \frac{\partial \dot{\psi}}{\partial \psi} \end{bmatrix} \tag{6.4}$$

$$\mathbf{f_{11}} = \mathbf{f_{13}} = \mathbf{f_{14}} = \mathbf{f_{15}} = \mathbf{f_{16}} = \mathbf{f_{23}} = \mathbf{f_{24}} = \mathbf{f_{25}} = \mathbf{f_{26}} = [\mathbf{0}_{3x3}] \mathbf{f_{12}} = [\mathbf{1}_{3x3}]$$

$$\mathbf{f_{21}} = \begin{bmatrix} \frac{\hat{\rho}\hat{\dot{x}}^2 A_{ref}}{16840m} C_x & 0 & 0 \\ \frac{\hat{\rho}\hat{\dot{y}}^2 A_{ref}}{16840m} C_y & 0 & 0 \\ \frac{\hat{\rho}\hat{\dot{z}}^2 A_{ref}}{16840m} C_z & 0 & 0 \end{bmatrix} \quad \mathbf{f_{22}} = \begin{bmatrix} \frac{\hat{\rho}\hat{\dot{x}} A_{ref}}{m} C_x & 0 & 0 \\ 0 & \frac{\hat{\rho}\hat{\dot{y}} A_{ref}}{m} C_y & 0 \\ 0 & 0 & \frac{\hat{\rho}\hat{\dot{z}} A_{ref}}{m} C_z \end{bmatrix}$$

$$\mathbf{f_{33}} = \begin{bmatrix} 0 & \frac{\sin\theta(r\cos\phi + q\sin\phi)}{\cos^2\theta} & \frac{q\cos\phi - r\sin\phi}{\cos\theta} \\ 0 & 0 & r\cos\phi - q\sin\phi \\ 0 & r\cos\phi + q\sin\phi + \frac{\sin^2\theta(r\cos\phi + q\sin\phi)}{\cos^2\theta} & \frac{\sin\theta(q\cos\phi - r\sin\phi)}{\cos\theta} \end{bmatrix}$$

$$\mathbf{f_{34}} = \begin{bmatrix} 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \\ 0 & \cos\phi & \sin\phi \\ 1 & \frac{\sin\phi\sin\theta}{\cos\theta} & \frac{\cos\phi\sin\theta}{\cos\theta} \end{bmatrix} \mathbf{f_{31}} = \mathbf{f_{32}} = \mathbf{f_{35}} = \mathbf{f_{36}} = [\mathbf{0}_{3x3}]$$

$$\mathbf{f_{41}} = \begin{bmatrix} \frac{Q_p A_{ref}}{16840 I_{xx} m} C_l & 0 & 0 \\ \frac{Q_p A_{ref}}{16840 I_{yy} m} C_m & 0 & 0 \\ \frac{Q_p A_{ref}}{16840 I_{zz} m} C_n & 0 & 0 \end{bmatrix} \mathbf{f_{42}} = \begin{bmatrix} \frac{Q_p A_{ref}}{I_{xx} m} C_l & 0 & 0 \\ \frac{Q_p A_{ref}}{I_{yy} m} C_m & 0 & 0 \\ \frac{Q_p A_{ref}}{I_{zz} m} C_n & 0 & 0 \end{bmatrix} \tag{6.5}$$

$$\mathbf{f_{43}} = [\mathbf{0}_{3x3}] \quad \mathbf{f_{44}} = \begin{bmatrix} 0 & -\frac{r(I_{zz} - I_{yy})}{I_{xx}} & -\frac{q(I_{zz} - I_{yy})}{I_{xx}} \\ -\frac{r(I_{xx} - I_{zz})}{I_{yy}} & 0 & -\frac{p(I_{xx} - I_{zz})}{I_{yy}} \\ -\frac{q(I_{yy} - I_{xx})}{I_{zz}} & -\frac{p(I_{yy} - I_{xx})}{I_{zz}} & 0 \end{bmatrix}$$

$$\mathbf{f_{45}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1.5 Q_p A_{ref} d}{m} & 0 \\ 0 & 0 & \frac{1.5 Q_p A_{ref} d}{m} \end{bmatrix} \mathbf{f_{46}} = \begin{bmatrix} \frac{Q_p A_{ref}}{I_{xx} m} & 0 & 0 \\ 0 & \frac{Q_p A_{ref}}{I_{yy} m} & 0 \\ 0 & 0 & \frac{Q_p A_{ref}}{I_{zz} m} \end{bmatrix}$$

$$\mathbf{f_{51}} = \mathbf{f_{52}} = \mathbf{f_{53}} = \mathbf{f_{54}} = \mathbf{f_{55}} = \mathbf{f_{56}} = \mathbf{f_{61}} = \mathbf{f_{62}} = \mathbf{f_{63}} = \mathbf{f_{64}} = \mathbf{f_{65}} = \mathbf{f_{66}} = [\mathbf{0}_{3x3}]$$