2014

Fuzzy Guidance, Navigation and Control of a Spacecraft Simulator

Brian Wisniewski
Lehigh University

FUZZY GUIDANCE, NAVIGATION AND CONTROL OF A HOPPER SPACECRAFT SIMULATOR

By

Brian Wisniewski

A Thesis

Presented to the Graduate and Research Committee

Of Lehigh University

In Candidacy for the Degree of

Master of Science

In

Mechanical Engineering and Mechanics

Lehigh University

Fall 2014

CERTIFICATION OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

_____

Date

_____

Thesis Advisor, Dr. Terry Hart

_____

Chairperson of Department, Dr. Gary Harlow

ACKNOWLEDGEMENTS

First and foremost, I cannot express more thanks to my advisor, Terry Hart; his continued support, encouragement, and the opportunities he has provided me has been influential in my success. I am entirely grateful for the learning opportunities he has provided for me, during my education at Lehigh University.

The undergraduate support team for this project has been outstanding, and, without them, much of this work could not have been completed as quickly as it was. Specifically, I would like to thank Trevor Hayes and Joachim Amoah, who were both very dedicated members of this project. Their continued technical support in both manufacturing and hardware interfacing was greatly appreciated throughout their summer involvement.

Special thanks also must go towards Billy Hau. Billy was the first graduate student I met at Lehigh and also was the one who first introduced me to this project. Billy has taught me a lot during our partnership, specifically in the field of electronic and software interfacing and integration. He has been an essential asset to this work, especially in terms of hardware-software troubleshooting and software library development. I was fortunate to have such a hardworking and dedicated team member, and I couldn't have done this work without him.

Without the next individual, I may not have pursed a master's degree at Lehigh University. Dr. Vallorie Peridier was one of my most influential professors at Temple University. During my undergraduate studies, she constantly pushed me to solve increasingly complex engineering problems. Due to her courses, I was able to develop and refine my analytical and problem solving skills, while gaining invaluable confidence in my abilities to overcome challenges. The

skills and confidence I developed has pushed me to seek challenging opportunities in unfamiliar areas.

My deepest heartfelt appreciation goes out to Dana Reuther. Dana has been by my side through both our undergraduate and graduate studies, and I could not have survived it without her. During challenges in both work and life, Dana was always there to lessen the mood and remind me to take a deep breath, smile, and enjoy life. Her passion, motivation, drive, and work ethic has always been something I admired. If more people were like her, the world would truly be a better place.

Last, but not least, I would like to thank my father, Leo Wisniewski. Out of this entire thesis, this was the hardest paragraph to write. There are not words to express my appreciation for everything he has done for me and my sisters while growing up. Without a doubt in my mind, I am the man I am today because of him. With that being said, this thesis and my Masters in Science Degree is dedicated to him. Thank you.

TABLE OF CONTENTS

List of Figures

List of Tables

ABSTRACT

To further facilitate the development of the guidance, navigation, and control systems of the future extra-planetary vehicles, there is a need for a simplified, easy-to-repair test bed that is dynamically similar to the full scale spacecraft. To achieve such a platform, a 3:1 thrust-to-weight ratio modular simulator was designed. The simulator is constructed from high strength-low density composite materials coupled with hobby grade electronic motors and a custom flexible landing gear system to increase stability and reduce capsizing while landing.

For attitude control, a nonlinear Fuzzy Logic style control system was developed and analyzed against more traditional PID style control schemes used in the past generations. This new style of controller offers increased performance in attitude control. After a comprehensive and complete simulation analysis, the fuzzy logic controller was implemented using the open source computer BeagleBone Black. Feedback was deliver by the use of an inertial measurement unit

In addition to the development of a fuzzy logic attitude control system, work began on the development of a full guidance, navigation, and control (GNC) system. The GNC system that was developed was a trajectory controller in the form of a fuzzy logic cascade control law. The simplified control law was developed to mimic the control systems used in commercial aircraft autopilots, in which the trajectory is assumed to be 2D, where the spacecraft simulator remains pointing in the direction of its destination point. The controller was developed to accept different styles of trajectory and the entire system is modular in nature.

From the simulation analysis of the closed-loop system, system level design specification were determined for the flight hardware. Ultimately, after programming the controller and integrating the electronics, it was determined the total time-delay of the system exceeded the design specification. Because of the hardware limitations, the attitude controller was, at best, n neutrally stable. Future work is proposed to integrate a real time microcontroller to account for the limitations of the BeagleBone and the programming language chosen.

Chapter 1: Introduction

Mankind has always been a very curious species. Throughout history, this curiosity, driven by an inherent competitive nature, has driven humanity to adventure into the unknown. Space exploration is the best example of this. During the 20th century two Cold War rivals, the United States and the Soviet Union, competed against one other to be the world leader in space exploration. This struggle ultimately led to the moon race and the first man on the moon. The space exploration struggle that began during the Cold War has helped push humanity to develop advanced technologies for robotic systems capable of navigating the hazardous environments of space. To do so, advances in control systems were required that surpassed basic tele-operated control. Unmanned planetary exploration robots that have been developed out of the space race era include landers and rovers. Rover-style vehicles offer a range of advantages over their stationary counter-parts. The major advantages of rovers are the ability to explore larger areas of territory. Though the ability to traverse a planetary body increases the rover's exploration area, these areas are not limitless; traversing by ground is inheritably restrictive. To overcome this restriction, new exploration vehicles must be developed that offer more efficient ways of traveling.

The work encompassed by this thesis is devoted to the future of planetary exploration vehicles. A hopper-style planetary robotic exploration system is proposed to overcome the restrictive nature of the current Rovers. If successful, the hopper-style spacecraft will allow future exploration missions to have many new possible exploration locations on planetary surfaces. This type of system introduces new risks in both development and production stages, especially at the University level. The need for a simple and safe test bed, where control systems can be tested and optimized, is essential. The purpose of this work is to further develop an easy to use test bed and GNC control system that can be used for testing applications at Lehigh University.

1.1 Project Description

In September of 2007 the Google Lunar XPRIZE was announced. The prize offers a total of thirty million dollars to the first privately funded team who can land a spacecraft on the moon and travel more than 500 meters while transmitting high definition images and videos. Since the announcement of the competition, Lehigh University has been developing hopper spacecraft simulators for control system testing as part of a partnership with Penn State University, who has been developing a hopper spacecraft with which they are planning to compete. To facilitate development, Lehigh's hopper spacecraft simulators are designed to be dynamically similar to the Penn State Lunar Lion's vehicle.

Chapter 2: Previous Work

The following chapter is devoted to summarize the work that was previously done at Lehigh University regarding the hopper spacecraft simulator design and the control systems that were developed. This work was essential in moving forward with the space hopper project and, ultimately, was the frame work for the development that was continued in this thesis. The complete overview of the work previously done please see [5].

2.1: Simulator Evolution

During the initial development, phase 2 full-scale space hopper spacecraft simulator was developed. The first generation and second generation designs will be discussed in detail, along with respective knowledge learned. Ultimately, the two previous designs led the way to the design that was developed by this work and the designs of the future.

2.1.1: First Generation

During Lehigh's initial efforts with Penn State's Lunar Lion team, the following design constraints were the best to-date design constraints of the actual vehicle. Thus, as a point-of-reference, these design constraints where adopted by the original Lehigh University design team as a basis for the first generation space hopper spacecraft testing platform.

Initial Design Constraints:

1. Minimum thrust to weight ratio of 3:1
2. 5 kg. payload capability
3. Four Thruster propulsion system
4. Thrust vector control
5. Real time data acquisition
6. 15 min. flight time
7. Low cost components

The first generation spacecraft simulator was designed to feature four ducted fans which were mounted on a single-axis capable of rotating approximating twenty degrees in either direction. This allowed the vehicle to stay approximately level while controlling its position. The vehicle's frame was constructed from aluminum plates that were machined to minimize weight. The entire vehicle, including the power supply, had an approximate mass of 22.5 kg.

Figure 2-1: First Generation Hopper Spacecraft Simulator

The results of the first few flight tests quickly reveled major design flaws, the most significant was the propulsion system. Due to the total weight of the system, the maximum thrust to weight ratio was only 1.25:1. This caused the controller to constantly saturate the propulsion system while stabilizing, ultimately effecting flight stability. In addition, the propulsion system introduced high frequency, low-amplitude vibration, which led to several problems. Most significantly, the propulsion system would literally vibrate itself apart during flight. This was not only time-consuming to fix but also a safety hazard. Besides the excessive vibration problem, there were overall power requirements. Due to the power requirement of the system, the overall flight time was very limited. Several design modifications were attempted, but ultimately it was decided to scrape the prototype and go back to redesign.

2.1.2: Second Generation

Before design began on the second generation prototype, the initial design constraints were revised and modified, partly due to the Lunar Lion team, and also from the knowledge taken from the previous design.

Version 2 Design Constraints:

1. Composite Construction
2. Minimum thrust to weight ratio of 3:1
3. 5 kg. payload capability
4. Tri-rotor geometry
5. Real time data acquisition
6. 15 min. flight time
7. Low cost components

The second generation hopper spacecraft simulator was designed to be stiffer and lighter than its predecessor and thus a custom composite material was chosen for the frame design. To ensure control saturation and vibration did not occur, a comprehensive trade study was preformed to ensure a minimum thrust to weight ratio of 3:1 was obtained, while utilizing

hobby-grade DC motors for thrust generation. To reduce system complexity, the propulsion system was transitioned from a thrust vector control to attitude control. The second generation geometry was also changed to mirror the research being conducted at Penn State.

Figure 2-2: Second Generation Hopper Spacecraft Simulator

Testing revealed that the major design flaws of the first generation vehicle were corrected, but new design problems were discovered. The second generation hopper spacecraft simulator had a much higher center of mass and a nonflexible landing gear system which increased the risk of capsizing during landing. This was particularly the case when the vehicle was translating at a small velocity while trying to land. Capsizing caused propellers to break often and increased the risk of structural damage. Since the system was constructed from a single piece of custom composite, if any of the three arms broke, the entire structure would have to be reconstructed. Ultimately, this effected the development pace for the GNC system, and more precautions were taken during testing to ensure structural damage would not occur.

2.2: Control System Development

The original work done at Lehigh University regarding control system development was focused on attitude control using linear control systems. To do so, flight electronics and control laws where developed an implement.

2.2.1: Control System Hardware

The control system hardware that was integrated into the first generation and second generation hopper spacecraft simulators consisted of a flight computer, inertial measurement unit, brushless motor controller, and a wireless communication module.

For the attitude stabilizing flight computer, an Arduino Mega was used. This computer was chosen due to its low cost, and large open source community. The Arduino series of microcontrollers are very powerful but had a limited memory, making them not suitable for more comprehensive and complex control systems.

Figure 2-3: Arduino Mega Microcontroller

The inertial measurement unit (IMU) was used to communicate the vehicles angular position to the flight computer. The IMU that was chosen previously was the VN-100 AHRS. This IMU was chosen due to its nine degrees of freedom capabilities and its small footprint and mass.

Unfortunately, it was found that this specific IMU was difficult to use and also extremely expensive.

Figure 2-4: VN-100 IMU

To control the speed of the DC brushless motors, an electronic speed controller (ESC) is need. The ESC is sent a PWM signal from the flight computer, and then it supplies the correlating voltage to each motor. Speed controllers are relatively universal but can vary significantly in respect to PWM response across various products, the particular ESC chosen was the Phoenix Ice 50. The Phoenix Ice 50 has internal filtering and timing software which made it an attractive model over others. The filtering and timing software that is built-in allows the user to adjust the signal timing and response time, which ultimately makes it more versatile.

Figure 2-5: Phoenix Ice 50 ESC

Lastly, to control the spacecraft simulator remotely, a wireless module was used. The wireless module that was chosen was the Xbee Pro. The Xbee Pro was chosen because of its ability to integrate into the Arduino flight computer flawlessly. Also the Xbee Pro is a very cost effective solution for wireless communication within a moderate range.

Figure 2-6: Xbee Pro Wireless Module

2.2.2: Roll & Pitch Control

For roll and pitch control a P-P controller and P-PD controller were designed and implemented. For the P-P controller, stabilization was achieved, but the system required increased damping to overcome excessive oscillations. To reduce the oscillatory nature of the system, a derivative term was added. The P-PD controller was simulated, and the results showed decreased overshoot, decreased steady state time, and decreased the overall oscillation of the system. The flight test results showed the system was able to perform close to the simulated benchmark analysis.

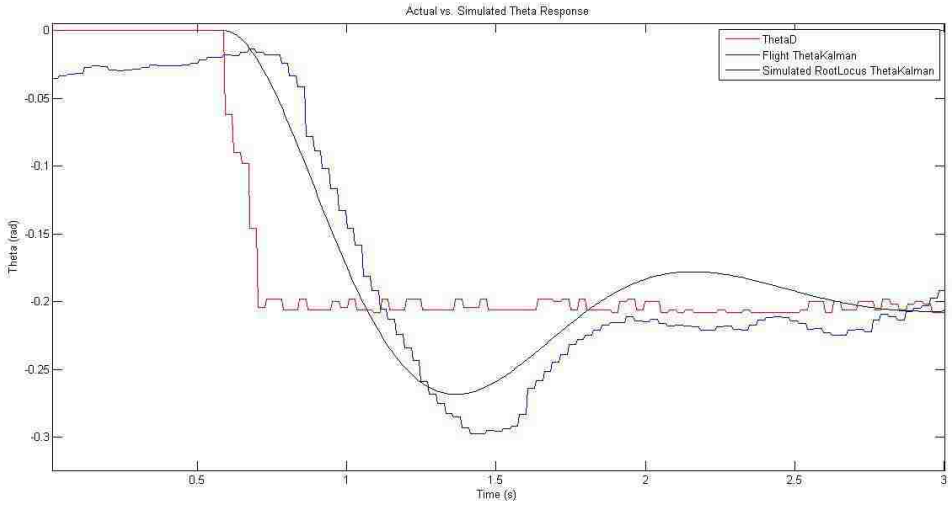


Figure 2-7: Real World Control System Test

2.2.3: Yaw Control

The yaw controller that was implemented was similar to that of the roll and pitch controller, but, instead of a P-PD control scheme, a P-D controller was used. Due to the highly non-linear torque outputted by each actuator, the yaw control was non-robust and not easily stabilized. For the purpose of attitude stabilization, the yaw control was deemed the least important factor. Even though the controller was not robust, the controller was deemed well enough for the purposes of this project.

2.3: Conclusion

By analyzing the previous work on hopper spacecraft simulator design and control system, several key areas of concern where recognized. For starters, the 2nd generation hopper spacecraft simulator was a very large improvement from its previous generation, but there are still two areas that needed to be revisited. The 3rd generation spacecraft simulator must have improved landing stability to reduce the chances of flipping over and causing damage to flight electronics or the propulsion system. Similarly, the next generation simulator must be designed to be easily repaired. Our efforts at Lehigh University is primarily related to GNC development and thus, during early stages of testing, crashes are frequent. Having a system that is easily repairable is necessary to ensure more time is spend on GNC development and not simulator maintenance.

The second area of concern is related to the low level attitude control system. High level GNC systems will only be successful with a robust and fast responding attitude controller. The previous P-PD roll/pitch controller response time it capped between 3-5 seconds. This may be problematic when development GNC systems. Also a major problem was shown in the yaw axis. To have a successful GNC system the yaw axis must be well behaved and easily controllable. Actual flight tests showed that yaw stability was not well accomplished using a P-D control scheme.

Chapter 3: Background Information

The sections contained in this chapter was written as an introduction to both classical set theory, fuzzy set theory, and fuzzy logic control. More complete discourse on these subjects can be found in [2], [3], and [4].

3.1: Classical Set Theory Overview

In real-world engineering processes, engineering elements, stress, strain, energy, etc., may be described by a physical set of non-negative integers. Even though engineering theory often allows for scenarios to exist of non-bounded elements, in most cases there are physical limitations that do not allow such elements to actually occur. To better understand this, let's consider engineering stress. It is theoretically possible to allow the stress of a cylinder, which is loaded axially, to become unbounded if you allow the cross sectional area of the cylinder to decrease until singular. In actuality the real-world phenomenon of failure would occur, thus limiting the maximum stress to the ultimate stress of the material. Therefore, for each material, there is some finite stress universe that can be described from $[0, \sigma_{ultimate}]$, where $\sigma_{ultimate}$ is the ultimate strength of a given material.

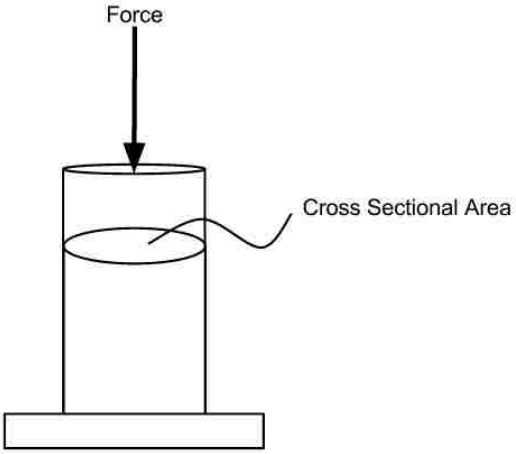


Figure 3-1: Axial Loaded Cylinder

It is often convenient to represent engineering elements belonging to some finite universe mathematically for analytical purpose. Relating elements to sets mathematically can be simply accomplished through the use of element relation operators as:

$x \in X$	Element x belongs to the X universe
$x \notin Y$	Element x does not belongs to the X universe

Table 3-1: Elemental Membership Operators

Note that elements can either belong to or not belong to a set. This is obvious when thinking about stress in terms of the chair leg again. As we discussed previously, each material would undoubtedly have its own unique universe of stress, which intuitively would be bounded between zero and some ultimate stress depended upon material. Similarly, for some set A, which contains the universe of stress elements of a weak material, and set B which contains the universe of stress elements for a stronger material, then it could be said that set A is contained in set B.

It is easy to see how engineering elements may be grouped into sets, and engineering sets may belong to other sets. This may seem obvious, but is a rather powerful realization, specifically when applied to control problems. Mathematically, sets can be defined through relational operators similar to the way in which elements are in Table 3-1. For set relationships the well-known subset and superset relation operators can be used as follows:

$A \subset B$	A is contained in B (if $x \in A$, then $x \in B$,)
$A \subseteq B$	A is contained in or is equivalent to B
$(A \leftrightarrow B)$	$A \subseteq B$ and $B \subseteq A$ (A is equivalent to B)

Table 3-2: Set Relationship Operators

3.1.1: Classical Set Operators

From the foundation of classical elements and sets, more comprehensive set operators can be developed. To start, consider a generalized two-dimensional universal set, X, that contains elements x, and elements y. Like discussed previously, for most engineering applications, it is convenient to group elements together to form sets that correlate to some physical situation. Thus in terms of our universal elements x and y, two generalized subsets A and B can be constructed, see Figure 3-2.

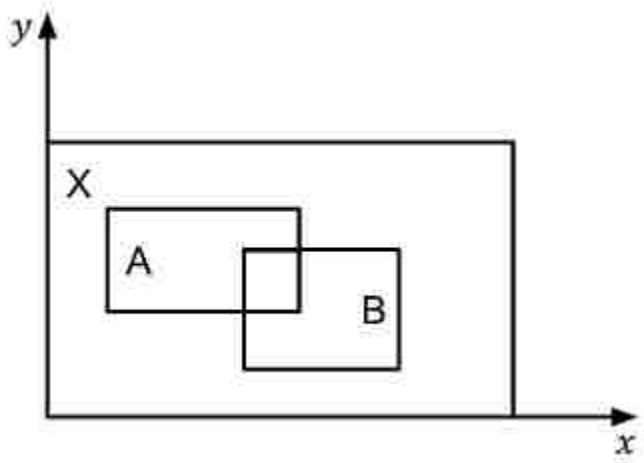


Figure 3-2: Sets, A, B and X

From both an anylitical and practical perspective, it is often convienient to construct new sets from pre-existing sets. To do so, set operators can be used. The most widely used set operators include, union, intersection, complement, and difference.

The union operator is used to define elements that are contained in two or more sets. For example, if we consider a new set C, which is defined through the union of sets A and B

described above, the elements contained in C would be defined by the elements that are contained in set A or set B or in both set A and B; see Figure 3-3a.

Similarly, the intersection operator is used to create new sets from elements shared across multiple sets. In terms of our example, the set C, that is defined by the intersection of sets A and B, is the collection of elements contained in the overlap region, see Figure 3-3b.

The complement operator is different than the pervious two operators discussed because the complement operator is exclusively used to compare a universal set and some subset contained in the universal set. Specifically, the complement, set A, is the elements contiained in the universal set X, exluding the elements contained in A, see Figure 3-3c.

Similar to the complement operator, the difference operator, or relative complement operator, which creates a new set by removing elements that are shared between two sets. In respect to our example, the realtive complement of A in B results in the set of elements that are contained exclusively in A; see Figure 3-3d.

Union	$A \cup B = \{x \mid x \in A \; or \; x \in A\}$
Intersection	$A \cap B = \{x \mid x \in A \; and \; x \in B\}$
Complement	$\bar{A} = \{x \mid x \notin A \; and \; x \in X\}$
Difference	$A \mid B = \{x \mid x \in A \; and \; x \notin B\}$

Table 3-3: Set Operators & Mathematical Notation

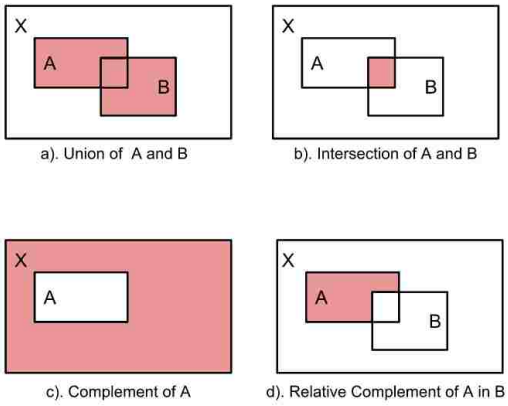


Figure 3-3: Set Operation Results

3.1.2: Properties of Classical Sets

To further mathematically analyze sets, set properties must be define as they pertain to the operators defined in the previous section. This allows for a framework on how classical sets can be mathematically manipulated. Without going into details on each, the following properties are defined in respect to a given set operator:

Commutative:

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

Associative:

$$A \cup (B \cup C) = (A \cup B) \cup C$$

$$A \cap (B \cap C) = (A \cap B) \cap C$$

Distributive:

$$A \cup (B \cup C) = (A \cup B) \cup (A \cup C)$$

$$A \cap (B \cap C) = (A \cap B) \cap (A \cap C)$$

Idempotency:

$$A \cup A = A$$

$$A \cap A = A$$

Identity:

$$A \cup \phi = \mathrm{A}$$

$$A \cap \phi = \phi$$

$$A \cap X = A$$

$$A \cup X = X$$

Transitivity:

$$if\ A\ \subseteq B\ and\ B \subseteq C, then\ A \subseteq C$$

Involution:

$$\bar{\bar{A}} = A$$

3.1.3: Set Mapping

In general, mapping is done to relate two or more sets. The most classic form of mapping is through functional relationships. Where some function is used to map some input space to some output space. For example, consider the most basic function, $f(x) = mx + b$, where the input space X is mapped to the output space Y through the function $f(x)$, which in general is denoted as:

$$f: X \to Y$$

The algebraic expression used to map elements of one set, to elements of another is often referred to as a mapping function. It is import to note that mapping functions are not the only mechanism for mapping sets to one another. Alternative to mapping functions, rule base mapping can be used. Rule based mapping can take many forms, but one form of particular usefulness is IF – THEN style mapping, which will be discussed in detail in Section 3.2.

3.1.4: Indicator Functions

An indicator function, sometimes referenced to as a characteristic function, is a function that maps elements of a given universe to a containing subset. To understand this, let us consider a general one dimensional set A, where $A \subset X$.

As we have discussed previously, in respect to classical logic, some general element x_i, where $x_i \in X$, may either belong or not belong to set A. The indicator function is the tool used to indicate the membership of elements of a subset. This is done through mapping membership in terms of the binary set $\{0,1\}$, where the zero element represents non-membership and the one element represents full membership. Therefore the indicator function for a general set A is:

$$x_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$$

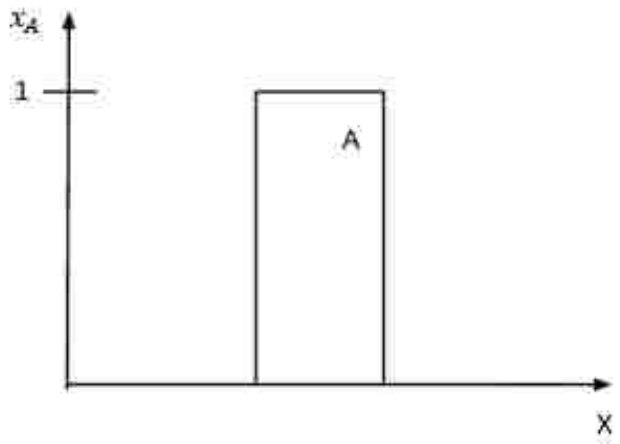


Figure 3-4: Graphical Representation of the Indicator Function of Set A

Earlier, it was shown how sets could be created from two or more sets through the use of set operators, specifically the set operators of union, intersection, and complement. When sets are created from said operators, the correlating indicator function can be determined by analyzing the original indicator functions. If we consider two general sets A and B, which both are subsets of the universal set X, where:

$$x_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$$

$$x_B(x) = \begin{cases} 1, & x \in B \\ 0, & x \notin B \end{cases}$$

Then the indicator function for the new set created through union, intersection or complement can be easily obtained via:

Union (OR operator): $A \cup B \longrightarrow x_{A \cup B}(x) = x_A \vee x_B = \max(x_A(x), x_B(x))$

Intersection (AND operator): $A \cap B \longrightarrow x_{A \cap B}(x) = x_A \wedge x_B = \min(x_A(x), x_B(x))$

Complement: $\bar{A} \longrightarrow x_{\bar{A}}(x) = 1 - x_A(x)$

3.2: Fuzzy Set Theory Overview

In classical set theory, as was just discussed, elements either belong to or they do not belong to a given universe, this is not the case with fuzzy sets. In fuzzy set theory, elements may belong to a fuzzy set with various degrees of membership. Thus there is no longer a well-defined boundary at the end of any universe; instead, the well-defined boundary is replaced with a boundary that allows ambiguity and vagueness. This at first seems not practical, but is actually quite intuitive and human. Unlike classical theory which tries to quantify the world through a binary filter, fuzzy logic tries to break down any predefined boundaries and re-quantifies the world as areas of grey.

Therefore fuzzy sets are sets that contain elements of varying degree of membership. These elements are then mapped to a universe of membership values between the interval of [0,1]. Since we noted before that, for any engineering application, sets are finite in practice, we will only discuss finite fuzzy sets. Thus for a given fuzzy set, the set is defined by its element and its degree of membership or fuzziness. If we consider the fuzzy set A, where the elements of A belong to X and the element's fuzziness is defined by the membership function μ_A (where $\mu_A(x_i) \in [0,1]$). From this the fuzzy set A can be expressed as:

$$A = \left\{\frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \cdots + \frac{\mu_A(n)}{x_n}\right\} = \sum_{n}^{i=1}\left\{\frac{\mu_A(x_i)}{x_i}\right\}$$

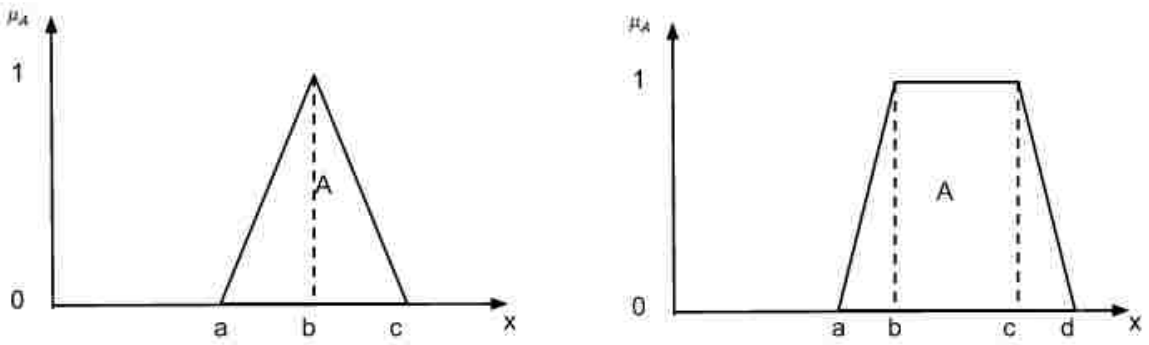
It is important to note in the expression above that the horizontal bar is not signifying division, but rather a way to group individual elements with their corresponding fuzziness. There are classical shapes used to group elements into fuzzy sets; these shapes include triangular, trapezoidal, Gaussian, and generalized bell shapes, to name a few. In most control problems, triangular and trapezoidal are used explicitly. The "indicator functions" for fuzzy sets are

referred to as membership functions. Examples of fuzzy membership functions for the two most widely used sets have been provided for reference:

$$\mu_{Tri}(x) = \begin{cases} 0, & x \leq a \\ \dfrac{x-a}{b-a}, & a \leq x \leq b \\ \dfrac{c-x}{c-b}, & b \leq x \leq c \\ 0, & c \leq x \end{cases}$$

$$\mu_{Trap}(x) = \begin{cases} 0, & x \leq a \\ \dfrac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \dfrac{d-x}{d-c}, & c \leq x \leq d \\ 0, & d \leq x \end{cases}$$

Where a, b, c and d are:



a). Triangular Membership Function b). Trapezoidal Membership Function

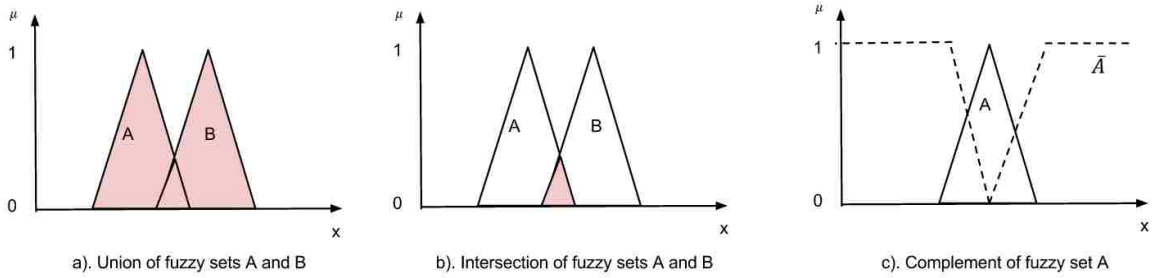
Figure 3-5: Fuzzy Membership Functions

3.2.1: Fuzzy Set Operations

First, three generalized fuzzy sets A, B, and C must be defined. For the purpose of this discussion, assume that fuzzy sets A and B both belong to the universe X, and that they are created via triangular membership functions. From this we can discuss the three set-theoretic operators union, intersection and complement. As discussed earlier, these operators are the same as the operators from classical set theory, but the crisp (non-fuzzy) characteristic indicator will be replaced with a fuzzy membership function.

Union:	$\mu_{A\cup B}(x) = \mu_A \vee \mu_B$
Intersection:	$\mu_{A\cap B}(x) = \mu_A \wedge \mu_B$
Complement:	$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$

This can be seen easily via the Venn diagram method, seen below:



a). Union of fuzzy sets A and B b). Intersection of fuzzy sets A and B c). Complement of fuzzy set A

Figure 3-6: Fuzzy Set Operators

3.2.2: Properties of Fuzzy Sets

Fuzzy sets rely on the same properties that crisp sets use, which is intuitive recalling that crisp sets are a special case of fuzzy set. Thus the properties that are defined in section 3.1.2 are exactly the same for fuzzy sets.

3.3: Fuzzy Control

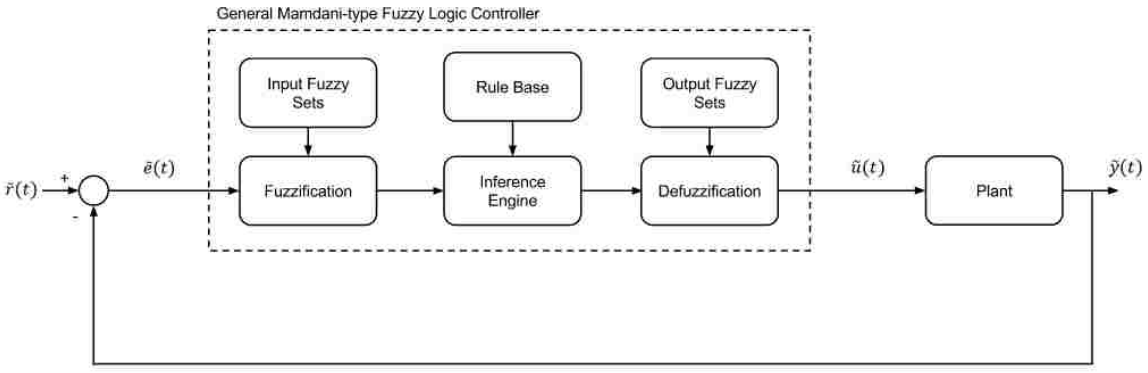


Figure 3-7: Mamdani-type FLC

A fuzzy logic controller (FLC) is a type of controller that is governed by a knowledge base, vague predicators, and an inference mechanism. In general, a FLC can be generalized in the form of a more traditional control law:

$$u(t) = F(e(t), e(t-1), \dots e(t-v), u(t-1), \dots, u(t-v))$$

Where F is a complex fuzzy mapping function which maps a finite set of controller inputs to a finite set of controller outputs. Since it is not reasonable to map all past value of $e(.)$ and $u(.)$ through F, more traditional control laws are often structured in the more classical form:

$$u(t) = F(e(t), \Delta e(t))$$

Where, $e(.)$ is the current system error and $\Delta e(.)$ is the change in error. In this form, F is only mapping the current state of the system to control output set. This form of control reduces the necessity of storing all past states of the system, which is otherwise inefficient and unnecessary.

Fuzzy Logic Control, in this form, was first proposed in 1975 by Mamdani and thus has been named the Mamdani-type Fuzzy Logic Controller (FLC). Mamdani-type FLC's are governed by three distinct mechanisms: Fuzzification, Inference Engine, and Defuzzification. The fuzzification mechanism maps crisp inputs into fuzzy inputs. Then through fuzzy reasoning, those fuzzy inputs are mapped to a fuzzy output space. Through defuzzification, the fuzzy outputs are translated into crisp outputs used to control a plant.

3.3.1: Fuzzification

The fuzzification mechanism is constructed by discretizing each variable universe into sub-universes constructed from fuzzy membership functions named by linguistic variables. Thus, for a generalized input variable x, it is shown in Figure 3-8, how the universe of x can be divided into fuzzy sets. A variable universe may be broken in numerous fuzzy sets, ranging in type, size, and overlap. The combinations are endless and plant specific. It is important to note here that minimizing the number of fuzzy sets used to define a variable is important when computational efficiency is necessary. Once a variable is passed through a fuzzification mechanism the result is a fuzzy variable that is defined by a series of linguistic variables and the associated degree of membership.

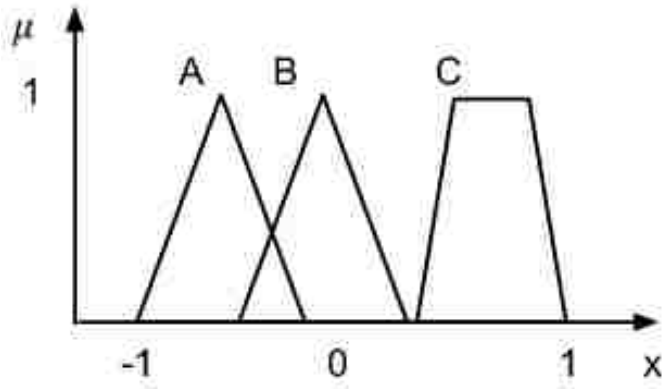


Figure 3-8: Fuzzification Membership Functions

This is often better described through example. Consider the variable, x defined above; if we assume a crisp input of 0, the mapping to an associated fuzzy input produces:

$$x = 0 \longrightarrow \left\{\frac{\mu_A(0)}{0}, \frac{\mu_B(0)}{0}, \frac{\mu_C(0)}{0}\right\} = \left\{\frac{0}{0}, \frac{1}{0}, \frac{0}{0}\right\}$$

Similarly if we assume a crisp input of 0.2, we can map that input to:

$$x = .2 \longrightarrow \left\{\frac{\mu_A(0.2)}{0.2}, \frac{\mu_B(0.2)}{0.2}, \frac{\mu_C(0.2)}{0.2}\right\} = \left\{\frac{.4}{0.2}, \frac{.4}{0.2}, \frac{0}{0.2}\right\}$$

From this example it is easily seen how a crisp value has two components, an element and a degree of membership. It is also critical to understand that an element on a given universe may belong to more than one fuzzy set.

3.3.2: Inference Engine

As discussed earlier, FLC is a complex system for mapping relevant input variables to relevant output variables. The mapping between fuzzy input sets to fuzzy output sets is constructed from conditional rules, which collectively will be referred to as the rule base. The rule base maps input linguistic variables to their associated linguistic output counterpart, while determining the maximum degree of membership for each output membership function. The

output shape and maximum degree of membership are combined through the union operator to construct a new fuzzy output set defined by a fuzzy indication function. To further detail the fuzzy reasoning process, consider another example. First assume we have some simplified generalized rule base consisting of two input variables, $\{x_1, y_1\}$, one output variable, $\{u\}$, and two rules in the form of IF – AND – THEN . Also, let's assume that the input – output universes are defined by triangular fuzzy sets in the form:

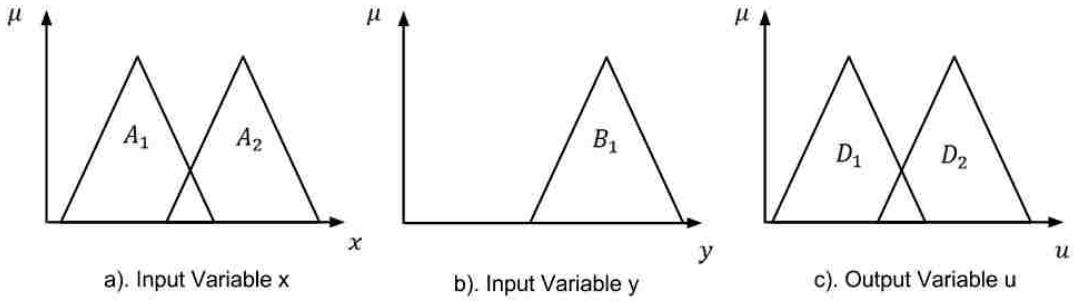


Figure 3-9: Input-Output Fuzzy Sets

Where the rule base is:

$$\text{IF } x_1 \in A_1 \text{ AND } y_1 \in B_1 \text{ THEN } u \in D_1$$

$$\text{IF } x_1 \in A_2 \text{ AND } y_1 \in B_1 \text{ THEN } u \in D_2$$

And for a given input pair, $\{x_1, y_1\}$, where:

$$x_1 \epsilon A_1, A_2 \longrightarrow \left\{\frac{\mu_{A_1}(x_1)}{x_1}, \frac{\mu_{A_2}(x_1)}{x_1}\right\}$$

$$y_1 \epsilon B_1 \longrightarrow \left\{\frac{\mu_{B_1}(y_1)}{y_1}\right\}$$

By plotting x_1 and y_1 on their respective variable universes, their corresponding fuzziness can easily be seen:

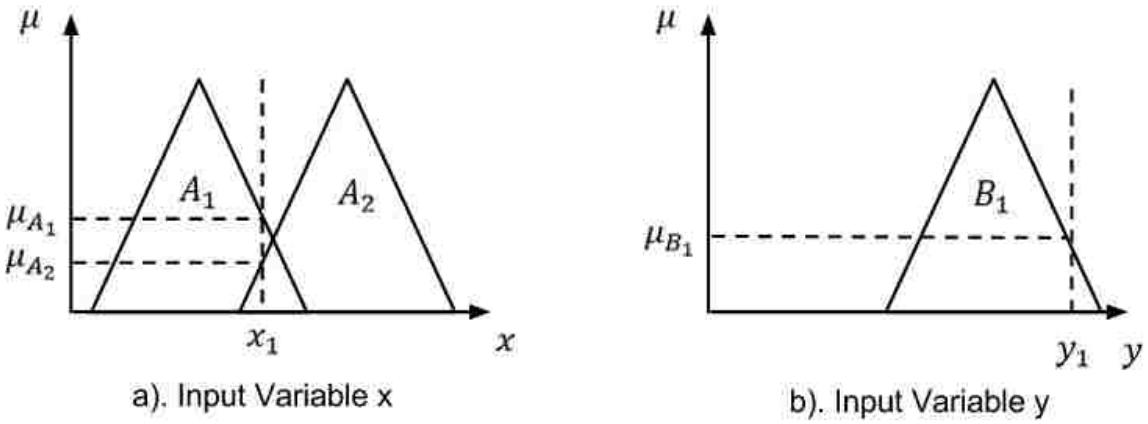


Figure 3-10: Input Fuzzification

From the rule base reasoning mechanism, the output variable must belong to fuzzy sets D_1 and D_2, or similary:

$$u \in D_1, D_2$$

The AND operator (minimum operator) in the rule base, dictates that the fuzzy sets must be "clipped" according to:

$$\mu_{D_1}^{max} = \min\left(\mu_{A_1}(x_1), \mu_{B_1}(y_1)\right) = \mu_{B_1}(y_1)$$

$$\mu_{D_2}^{max} = \min\left(\mu_{A_2}(x_1), \mu_{B_1}(y_1)\right) = \mu_{A_2}$$

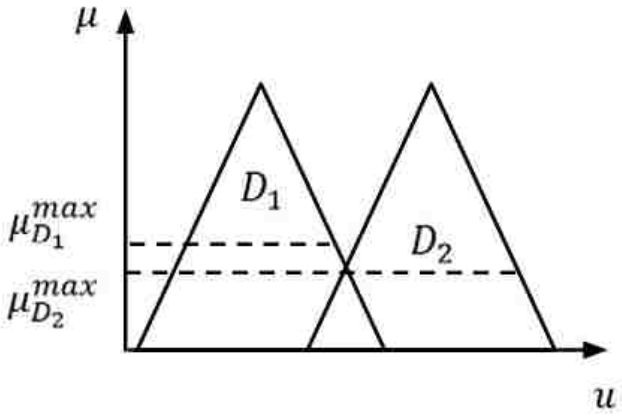


Figure 3-11: Membership Function Clipping

Combining the resultant output membership functions through the union operator and clipping the memberships functions at their maximum degree of membership, yields a piecewise fuzzy membership function in the shape of:

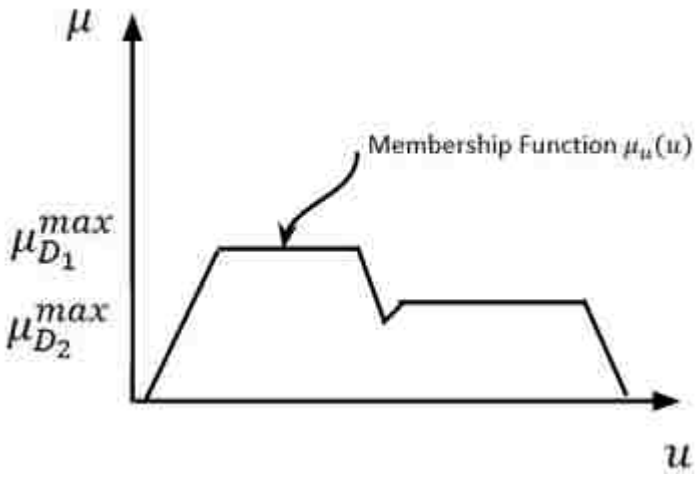


Figure 3-12: Fuzzy Inference Mechanism Result

3.3.3: Defuzzification

Defuzzification is the method used to produce a crisp (non-fuzzy) quantifiable control signal from the fuzzy membership function created through the inference mechanism. The way in which this control signal is computed varies depending on which defuzzification method is used. The five most widely used methods are centroid, bisector, middle, smallest and largest. As it will be shown, depending on which defuzzification method is used, the overall crisp output will vary. For this reason the choice of a defuzzification method will ultimately effect the performance of the fuzzy controller. For the purpose of this analysis let us consider the following output membership function resulting from some generalized inference engine:

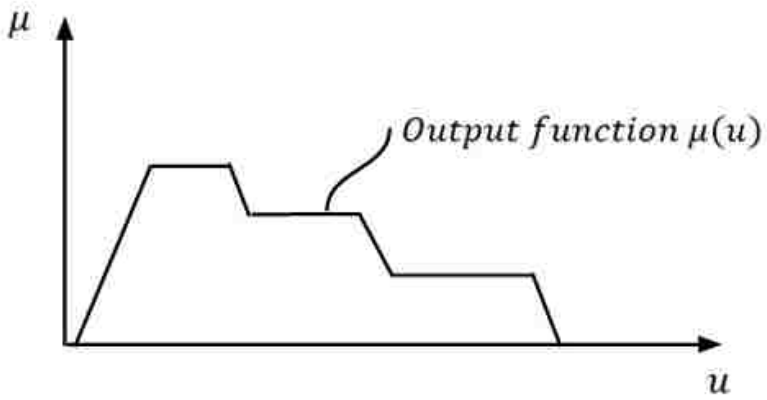


Figure 3-13: Defuzzification Function

Defuzzification Method #1: Centroid Method

The centroid method is arguably the most widely used method for control applications. With that being said, the centroid method is also is the most computationally intensive method. Often this method can be slow, adding an inherent time delay into the system, which can be problematic. The centroid method computes the center of area of the resulting output function via:

$$u = \frac{\int uAdA}{\int AdA}$$

Since for most cases the output function from fuzzy reasoning is piecewise, this computation must be done numerically in the form:

$$u = \frac{\sum_{i=1}^{n} u_i A_i}{\sum_{i=1}^{n} A_i}$$

Thus for the provided example function, the centroid method produces the following crisp output:

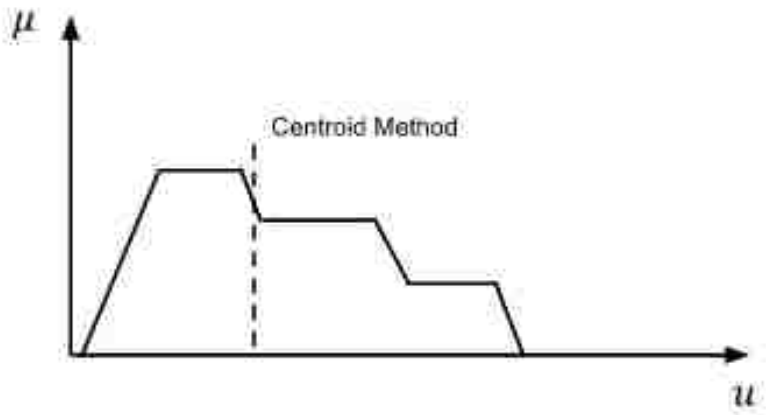


Figure 3-14: Centroid Method Result

Defuzzification Method #2: Bisector Method

The bisector method is very similar to the centroid method in the sense that it determines the crisp output by utilizing the area under the curve. In the bisector method, the crisp output is found in by diving the area under the curve in exactly half. This processes is computed mostly iteratively for complex output functions and again can be relatively slow. In some special simple cases, the bisector defuzzification method will produce the same value as the centroid method. For more complex output functions, the bisector method with produce slightly different results than that of the centroid method. This can be shown through the example provided:

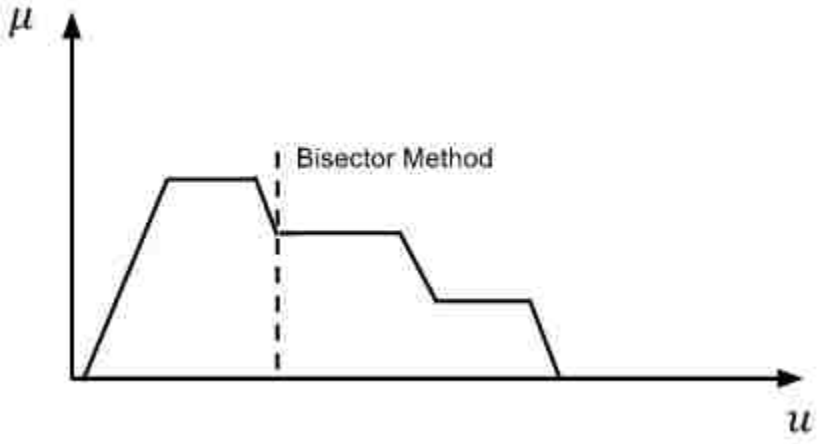


Figure 3-15: Bisector Method Result

Defuzzification Method #3, #4 and #5: Largest, Smallest and Middle of Maximum Methods

Because of their similarity, the remaining three methods will be grouped together. The largest, smallest, and middle of maximum defuzzification methods are the fastest methods and work great for systems with very few output membership functions. Specifically these methods do not require the analysis of the entire output, but rather, only requires the analyses of the region of maximum degree of membership. The largest of maximum method, uses the largest output value associated with the maximum degree of membership. Likewise the small and middle methods use the smallest and middle output value associated with the maximum degree of membership, respectively. This can be shown through the graphic provided:

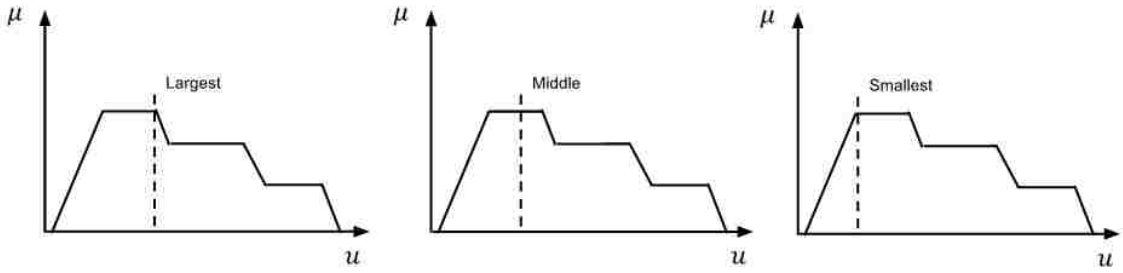


Figure 3-16: LOM, MOM, SOM Results

As stated prior, it is easy seen how the controller output is highly effected by the defuzzification method, a controller's output can vary significantly. For plants that are relatively simple to control, a simple defuzzification method like the largest, middle, smallest method may be sufficient, but these simple defuzzification mechanisms are often not robust in controlling more complex nonlinear systems. In most cases, the latter of the defuzzification methods are completely ignored and better-rounded defuzzification methods like the centroid method are used. To help decrease the time delay associated with the centroid method, much work has been done on centroid estimation algorithms that produces faster results with very little error.

Chapter 4: Third Generation Half-Scale Design

4.1: Frame Design

4.1.1: Material Selection

For the 3rd generation space hopper simulator, a modular frame design was decided. This was done to ensure simple and efficient repairs of structural damage accrued with testing the control systems. To begin the frame design, a material analysis was conducted. It was decided that the ideal material would have a high strength, high stiffness, and a low density. To determine the ideal material that meet these specification, Ashby plots where used. From the Ashby analysis, it was determined that the ideal type of material for this project would be a composite. In general composite materials have a relatively high strength and high stiffness with the added benefit of being extremely light compared to materials with similar values of strength and stiffness.

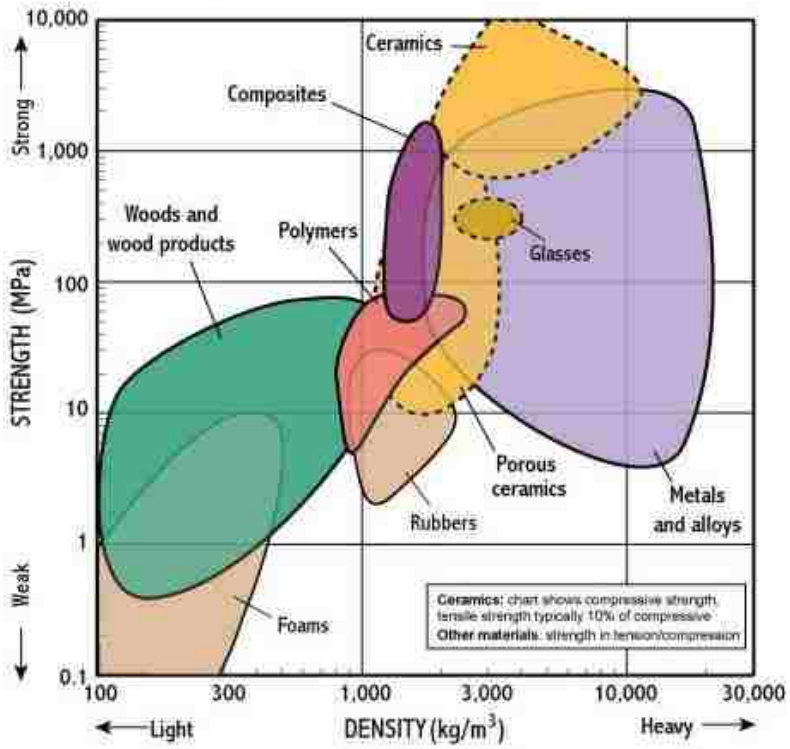


Figure 4-1: Ashby Density Verse Strength [5]

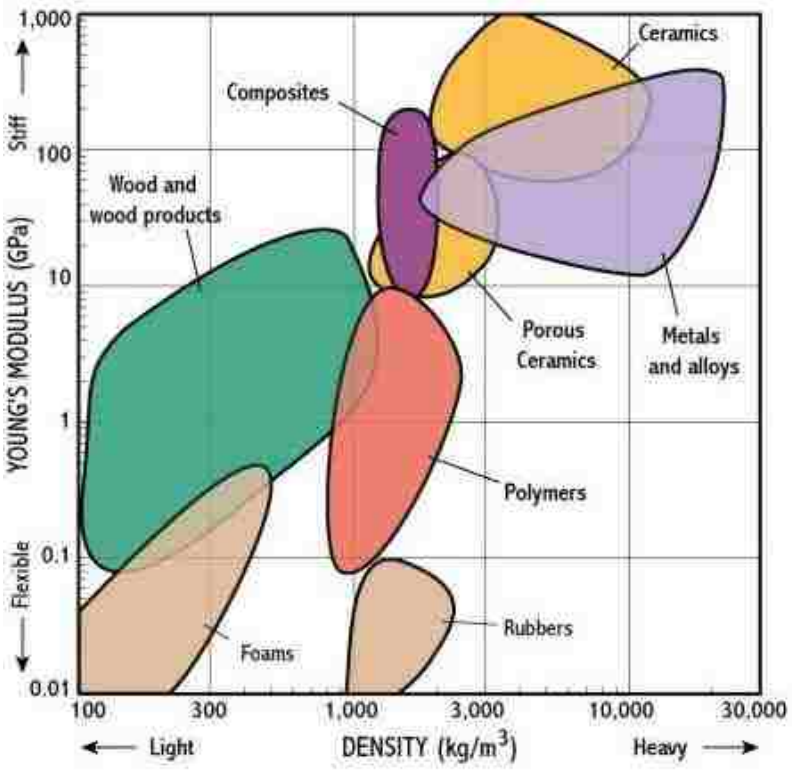


Figure 4-2: Ashby Density Verse Stiffness [5]

4.1.2: Design

The need for a module design became clear through the last two iterations of the hopper spacecraft simulator. Keeping this design constraint in mind, a simple and module frame design was developed.

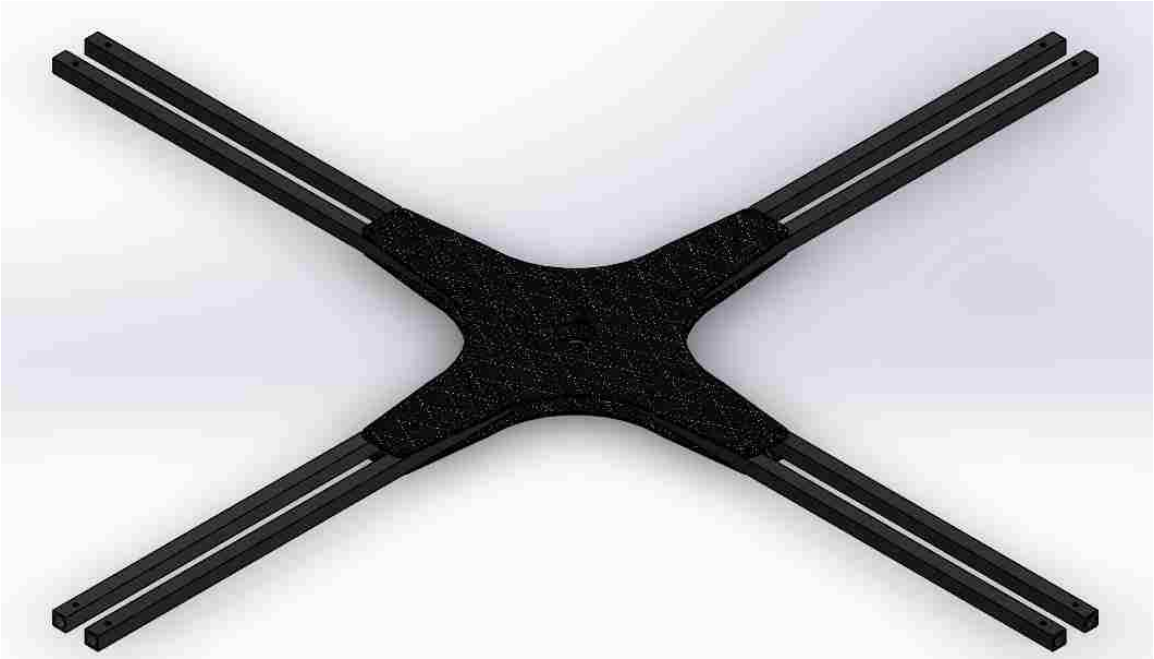


Figure 4-3: Frame Design

4.2: Propulsion System

For the propulsion system, it was decided to use brushless hobby grade DC motors as in the 2nd generation simulator. This decision was primarily driven by the mass availability of these actuators and their ease of use.

During the initial design phase, a comprehensive analysis was done on readily available brushless motors. Many motors were compared against one another, the major design concerns where efficiency, torque, and power consumption. Ultimately, it was decided to use the Antigravity MT2814 motor by T-Motor.

Figure 4-4: Propulsion System Actuator

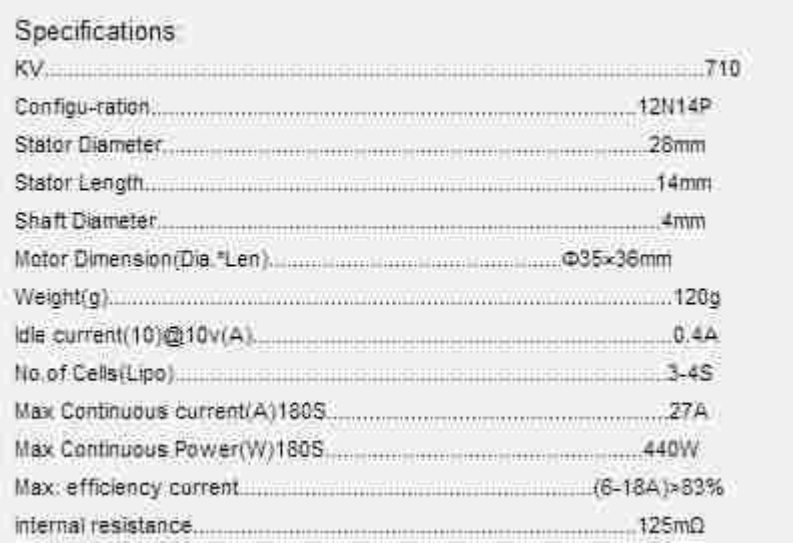


Figure 4-5: Propulsion Actuator Specifications [6]

Once the actuators were available, preliminary tests were conducted to determine the max thrust generated by the propulsion actuators based on propeller size and power consumption. During this initial testing phase three propeller sizes where considered 11x3.7, 12x 4, and 13x4.4.

Utilizing the thrust data collect, it was decided to use a 12 x 4 propeller size. This decision was driven by the estimated weight of the simulator and the desire to be able to achieve a max thrust to weight ratio of 3:1.

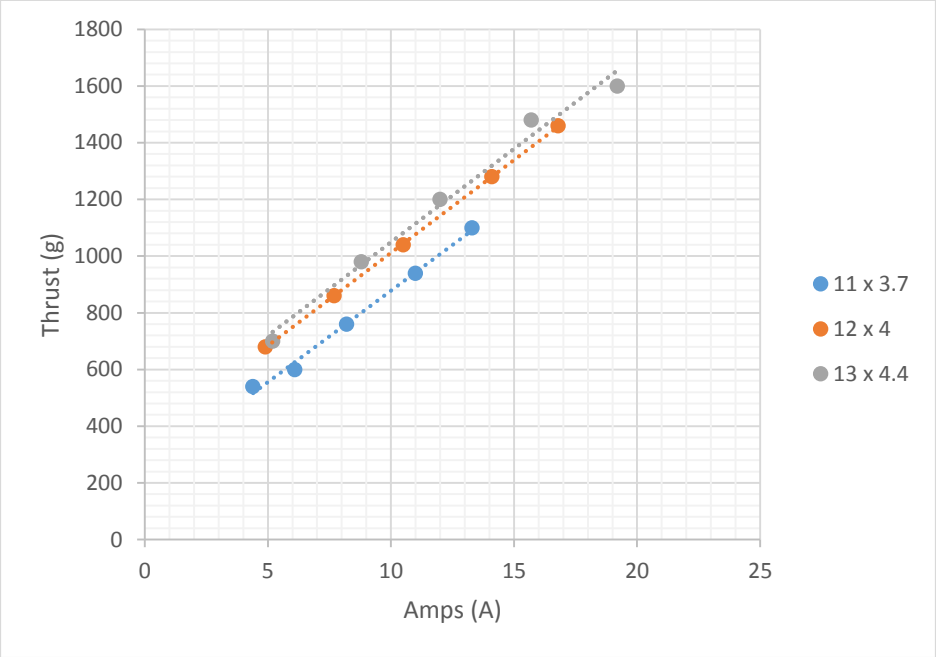


Figure 4-6: Amperage Verse Thrust

Figure 4-7: Brushless Motor Thrust/Torque Test Stand [1]

4.3: Power System

The power system design was driven by the actuator power requirements and the desired flight time. Like in the previous simulators, lithium-ion polymer (LIPO) batteries were chosen as the power source due to their high power density and discharge rates

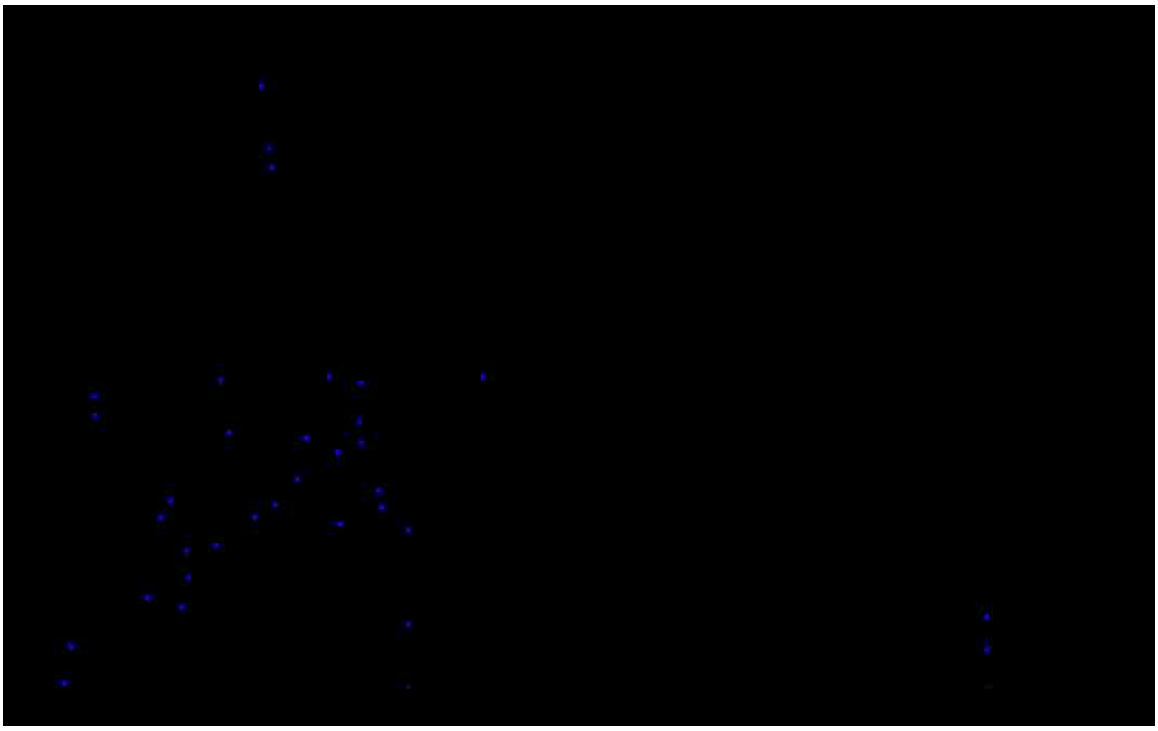


Figure 4-8: Power Density Chart [7]

For LIPO batteries, battery life can be calculated via:

$$Battery\ Life = \frac{Battery\ Capacity\ [mA\text{h}]}{Load\ Current\ [mA\text{h}]} \tag{4.1}$$

If it were assumed that the propulsion system will operate at a nominal load current of 6000mAh and 10 minute flight time was required, then the battery capacity can be found as:

$$Battery\ Capacity\ = 4000\ [mA\text{h}]$$

To help distribute the weight of the battery more evenly it was decide to use two batteries, from this design choice, the required battery specifications were calculated as:

Max Discharge [mA]	Battery Life [Min]	Battery Capacity [mAh]	Supply Voltage [V]
16000	10	2000	14.8

Table 4-1: LIPO Battery Specifications for a 2 Battery Power System

From the design specifications shown in Table 4-1 and through a detailed analysis of the best LIPO batteries on the market, it was decided to choose the ThunderPower 2100mAh 4-Cell G8 Pro Lite LIPO batteries for the simulators power system.

Figure 4-9: LIPO Battery Specifications

4.4: Landing Gear

From the knowledge learned from previous generations of the hopper spacecraft simulator, it was important to consider a landing gear system that is both low and flexible. This is to reduce any chances of capsizing during landing. To have a truly flexible landing gear system, it was decided to design a set of custom leaf springs and feet to be used as a landing gear systems.

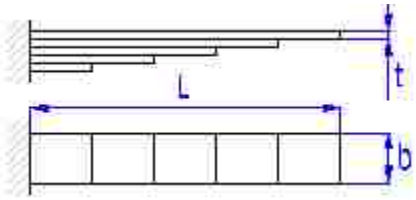


Figure 4-10: Leaf Spring Design

From an introduction course of Machine Design, the stress in a leaf spring can be found by:

$$\sigma = \frac{6 \cdot F \cdot L}{n \cdot b \cdot t} \tag{4.2}$$

And the spring deflection can be found by:

$$s = \left(\frac{3}{2 + \frac{n' + 1}{n}} \right) \cdot \left(\frac{4 \cdot F \cdot L}{E \cdot n \cdot b \cdot t^3} \right) \tag{4.3}$$

Where n is the total number of springs, n' is the number of extra full length springs, F is force applied to the spring, E is the module of elasticity.

From Equations 4.2 and 4.3, a parameter study was conducted. Given a material, a number of springs, a desired deflection, and a assumed load, the landing gear thickness t and width b were

varied to find sets of parameter's that match the assumed conditions without failure; see Figure 4-11.

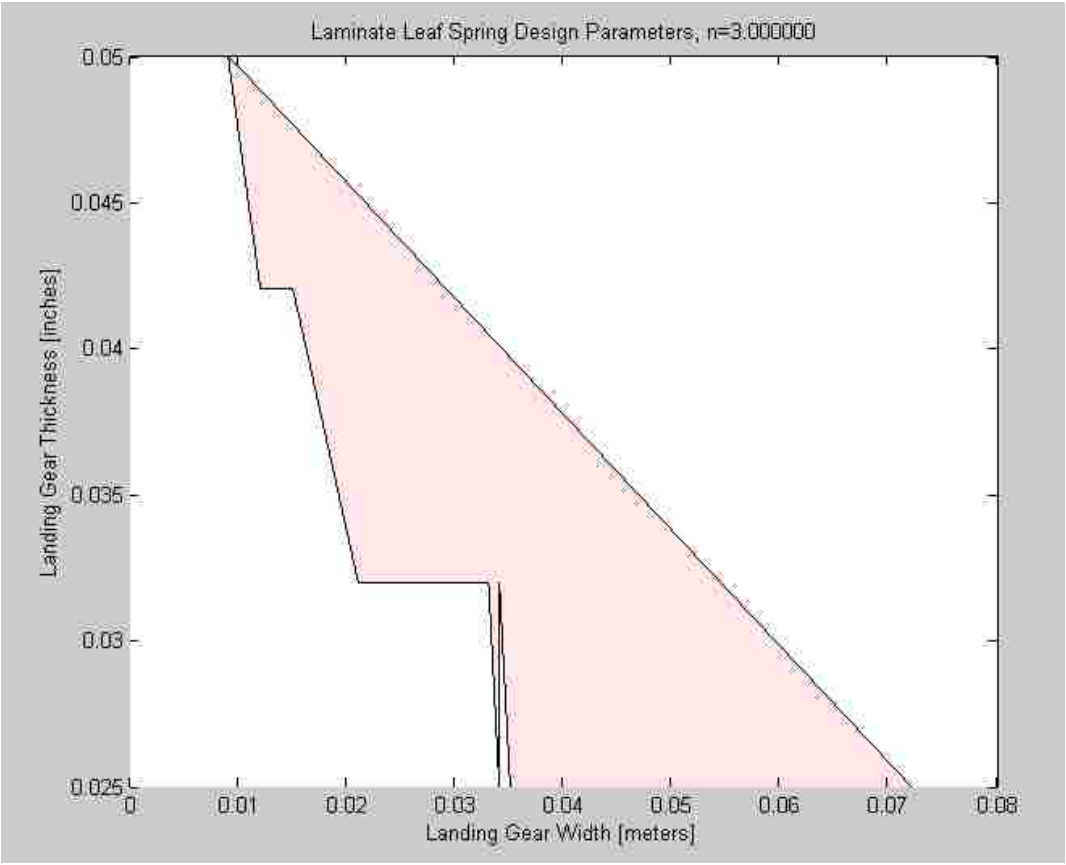


Figure 4-11: Example Parametric Study

From the parametric studies that where conducted the following design parameters where chosen as an initial design starting point:

Material: Spring Steel (1095)	
Parameter	Value
n	3
n'	0
L	12 cm
t	.12 cm
b	2 cm
E	200 Gpa

Table 4-2: Leaf Spring Design Parameters

Through the use of CAD software, the design was refined and finite element analysis was conducted to ensure design validated. The finalized landing gear design can be seen in the figure below.

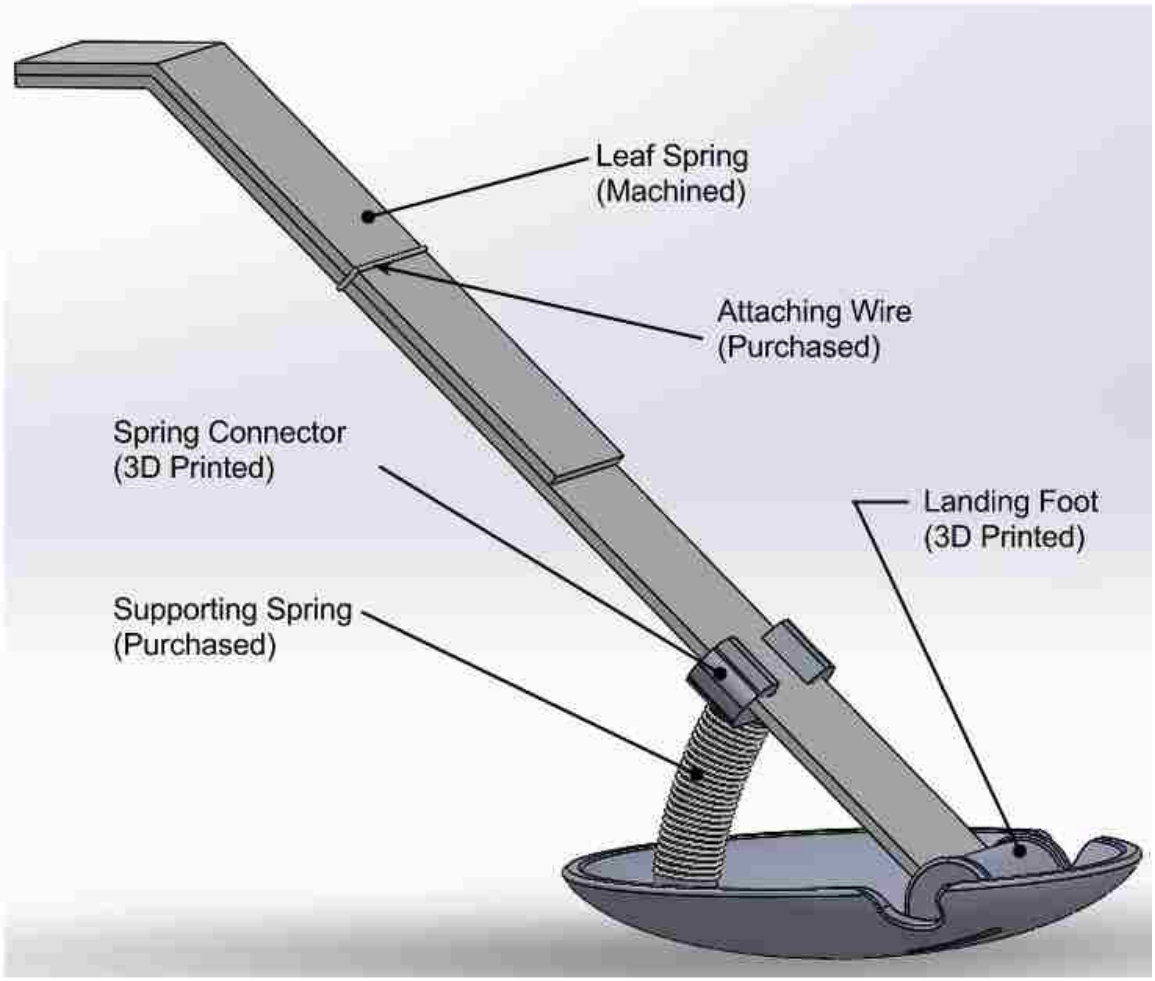


Figure 4-12: Landing Gear Design

4.5: Control System electronics

Below is a tabulated list of the flight electronic used for control system implementation?

Electronics Overview	
Flight Computer:	Beaglebone Black
IMU "Orientation Sensor":	UM6
Electronic Speed Controller:	QBrain
Wireless Module:	Xbee

Table 4-3: Flight Electronics

4.6: 3rd Generation Half-Scale Final Design

See Appendix A and Appendix B for detailed BOM and manufacturing drawings.

Figure 4-13: Finalized CAD Model

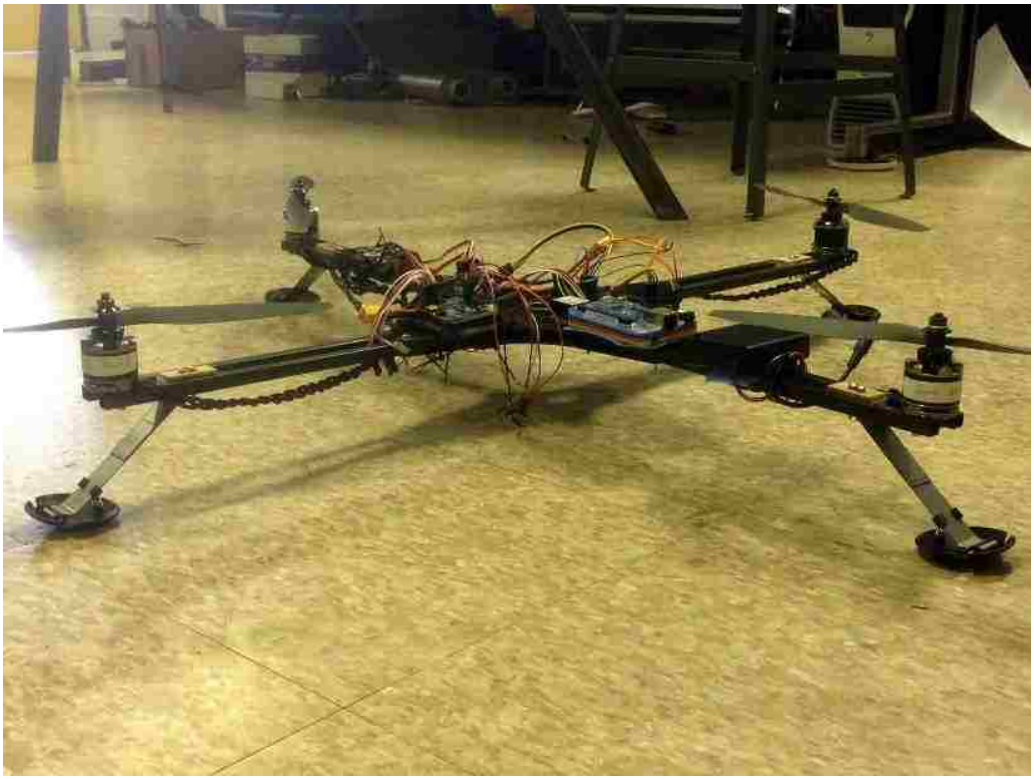


Figure 4-14: 3rd Generation Simulator

Chapter 5: Kinetics and Dynamics

5.1: Equations of Motion Derivation

Before work can begin developing control systems, the dynamics of the system must be understood. To ensure continuity of the equations of motion, it is convenient to set up frames of reference. This is particularly important when dealing with systems that will be utilizing all special dimensions. To simplify things, only two frames of reference will be used. Specifically, we will be utilizing the planetary, inertial fixed frame, and the vehicle fixed frame as seen below:

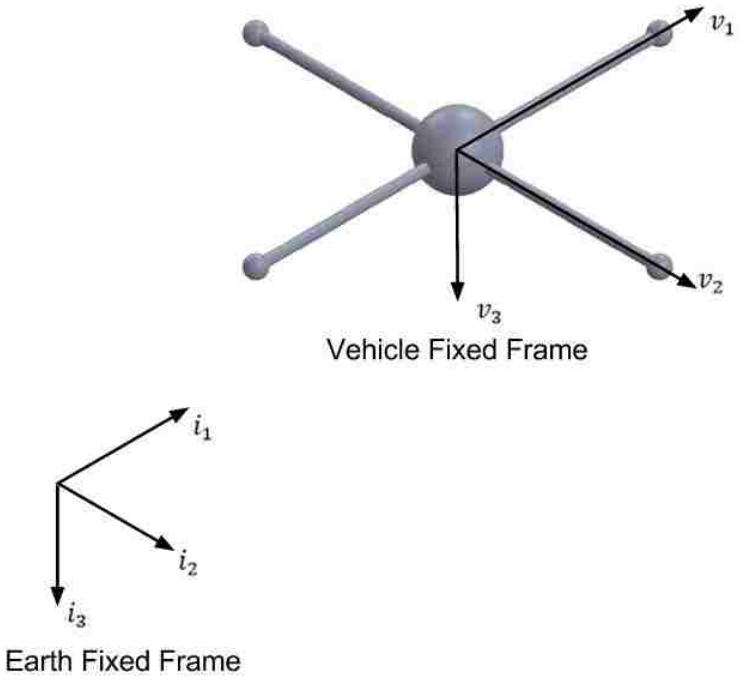


Figure 5-1: Frames of Reference

The Euler angles which will describe the vehicles rotation in the earth fixed frame where the pitch axis is v_1, the roll axis is, v_2 and the yaw axis is v_3. The Euler angles will be referred to as follows:

$$roll \equiv \phi$$

$$pitch \equiv \theta$$

$$yaw \equiv \psi$$

Based on the design of our system, the following FBD was determined:

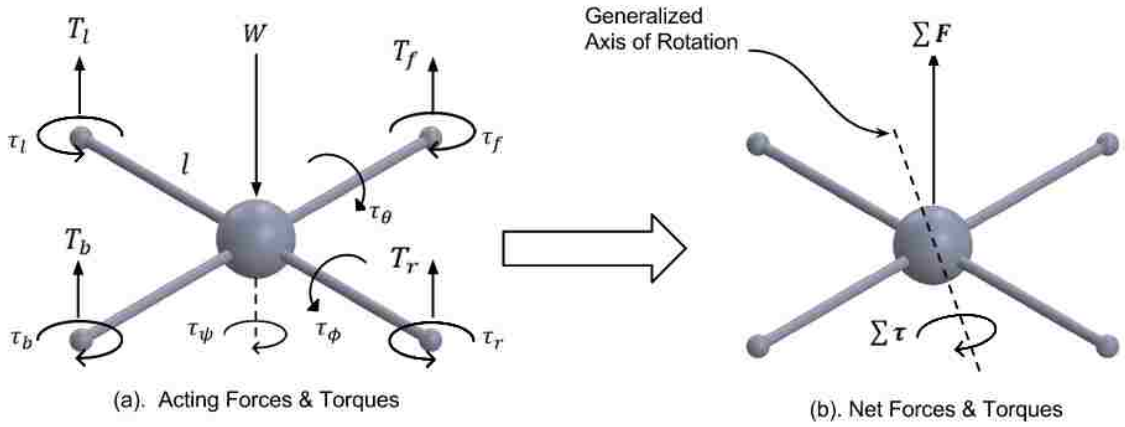


Figure 5-2: Free Body Diagram

By evaluating the provided FBD using the well-known Newton-Euler formulation, the correlating system of dynamic equations can be found. The Newton-Euler formalization will allow us the ability to describe both the translation and rotational dynamics of the systems in terms of the vehicle frame. We will begin the derivation from:

$$\begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{\tau} \end{bmatrix}_v = \begin{bmatrix} m\mathbf{1} & 0 \\ 0 & \boldsymbol{I_{cm}} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{\zeta}}_{\boldsymbol{cm}} \\ \ddot{\boldsymbol{\eta}} \end{bmatrix}_v + \begin{bmatrix} \mathbf{0} \\ \dot{\boldsymbol{\eta}} \times \mathbf{I_{cm}} \dot{\boldsymbol{\eta}} \end{bmatrix}_v \tag{5.1}$$

Where, $\boldsymbol{F}$ is the total force acting on the system, $\boldsymbol{\tau}$ is the total torque acting on the system, and m is the total mass of the system. Also:

$$\mathbf{1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

$$\mathbf{0} = [0 \quad 0 \quad 0]^T \tag{5.3}$$

$$\boldsymbol{I_{cm}} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{5.4}$$

$$\boldsymbol{\zeta} = [v_1 \quad v_2 \quad v_3] \tag{5.5}$$

$$\boldsymbol{\eta} = [\phi \quad \theta \quad \psi] \tag{5.6}$$

The forces acting on the simulator can be simply obtained by:

$$\boldsymbol{F} = [\boldsymbol{F}]_v + [\boldsymbol{G}]_v$$

Where $[\boldsymbol{F}]_v$ and $[\boldsymbol{G}]_v$ are the net actuator forces and net gravitation forces acting on the system in respect to the vehicle frame. The net gravitational force in respect to the vehicle frame, $[\boldsymbol{G}]_v$, can be obtained by applying a rotation matrix to the net gravitation force in respect to the inertial frame, thus:

$$[\boldsymbol{G}]_v = R^{-1}[\boldsymbol{G}]_i$$

And R is the rotation matrix from the inertial frame to the vehicle frame, note that $c\beta$ and $s\beta$ is short hand for $\cos(\beta)$ and $\sin(\beta)$ respectively:

$$R = \begin{bmatrix} c\theta c\psi & c\psi s\theta s\phi - c\phi s\psi & s\psi s\phi + c\psi c\phi s\theta \\ c\theta s\psi & c\psi c\phi + s\psi s\theta s\phi & c\phi s\psi s\theta - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}$$

The gravitational force acting on the spacecraft simulator in respect to the inertial frame is simply:

$$[\boldsymbol{G}]_{\boldsymbol{i}} = [0 \quad 0 \quad mg]^T$$

Since the actuators are fixed to the vehicle frame, the total actuator force $[\boldsymbol{F}]_v$ is defined as:

$$[\boldsymbol{F}]_v = [0 \quad 0 \quad -T]^T$$

Therefore $\boldsymbol{F}$ can be easily computed as,

$$\boldsymbol{F} = [-mgs\theta \quad mgc\theta s\phi \quad mgc\theta c\phi - T]^T \tag{5.8}$$

Similarly the total toque acting on the system is found as:

$$\boldsymbol{\tau} = [\tau_\phi \quad \tau_\theta \quad \tau_\psi]^T \tag{5.8}$$

Where τ_ψ is the total actuator toque, and τ_θ and τ_ϕ are the total pitching and rolling torque respectively. Therefore $\tau_\psi, \tau_\theta, \tau_\phi$ can be determined from:

$$\tau_\psi = \tau_f + \tau_b + \tau_l + \tau_r \tag{5.9}$$

$$\tau_\theta = l\left(T_f - T_b\right) \tag{5.10}$$

$$\tau_\phi = l(T_l - T_r) \tag{5.11}$$

Substituting equations 5.1-5.8 into equation 5.1 and simplifying yields the following set of attitude dynamics:

$$\boxed{\begin{bmatrix}\ddot{\psi}\\ \ddot{\theta}\\ \ddot{\phi}\end{bmatrix}_v = \begin{bmatrix}\frac{\tau_\psi}{I_{zz}} - \frac{\left(I_{xx} - I_{yy}\right)}{I_{zz}}\dot{\theta}\dot{\phi}\\ \frac{\tau_\theta}{I_{yy}} - \frac{(I_{zz} - I_{xx})}{I_{yy}}\dot{\phi}\dot{\psi}\\ \frac{\tau_\phi}{I_{xx}} - \frac{\left(I_{yy} - I_{zz}\right)}{I_{xx}}\dot{\theta}\dot{\psi}\end{bmatrix}_v} \tag{5.12}$$

While the translational dynamics where found to be,

$$\begin{bmatrix}\ddot{v}_1\\ \ddot{v}_2\\ \ddot{v}_3\end{bmatrix}_v = \begin{bmatrix}-gsin(\theta)\\ gcos(\theta)\sin(\phi)\\ gcos(\theta)\cos(\phi) - T/m\end{bmatrix}_v$$

By utilizing the rotation matrix R, the translation dynamics can be expressed in respect to the more useful inertial frame of reference through:

$$\begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ -\ddot{Z} \end{bmatrix} = R \begin{bmatrix} \ddot{v}_1 \\ \ddot{v}_2 \\ -\ddot{v}_3 \end{bmatrix}_v \tag{5.13}$$

From equations 5.12 and 5.13, a Simulink mode can be constructed to simulate the vehicles nonlinear dynamics. This model will be of importance for preliminary proof of concept testing the guidance, navigation and control systems that will be discussed in the following section.

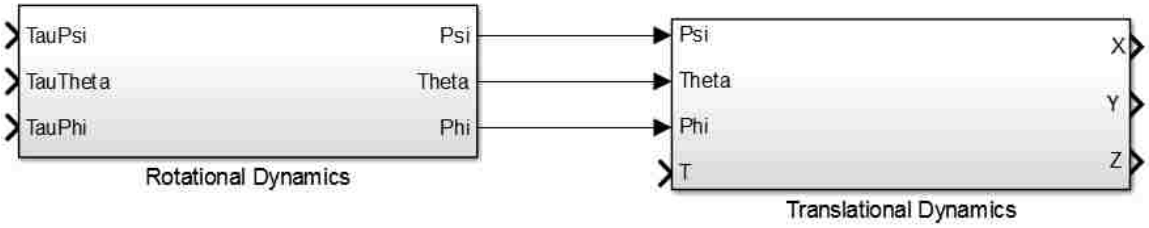


Figure 5-3: Simulink Nonlinear Dynamic Model

5.2: Motor Dynamics (System Identification)

To ensure the virtual model of the system is complete, the motor dynamics were experimentally modeled. To do so, a system identification approach was done. System identification is a general method which uses statistics to match input-output data to a mathematical model. This is convenient when trying to determine the dynamics of real world systems. To collect input-output sets, the thrust/torque test stand shown in Figure 4-7 was used. Each motor-propeller combination was tested and data was collected. Using the System Identification Tool Box built into MATLAB, the data was fitted to several different models. In general, it was found that a linear transfer function offered a good fit for the thrust dynamics, and a look up table worked the best for the torque dynamics.

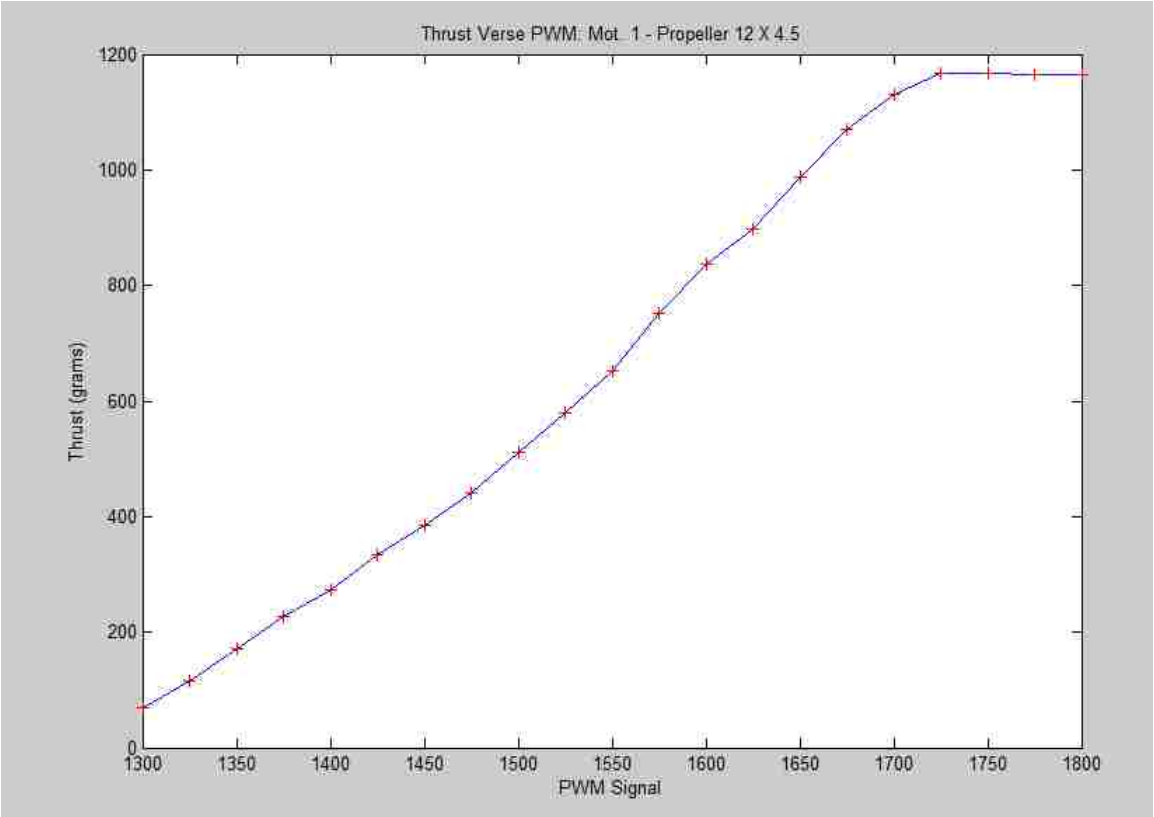


Figure 5-4: PWM Verse Thrust Data

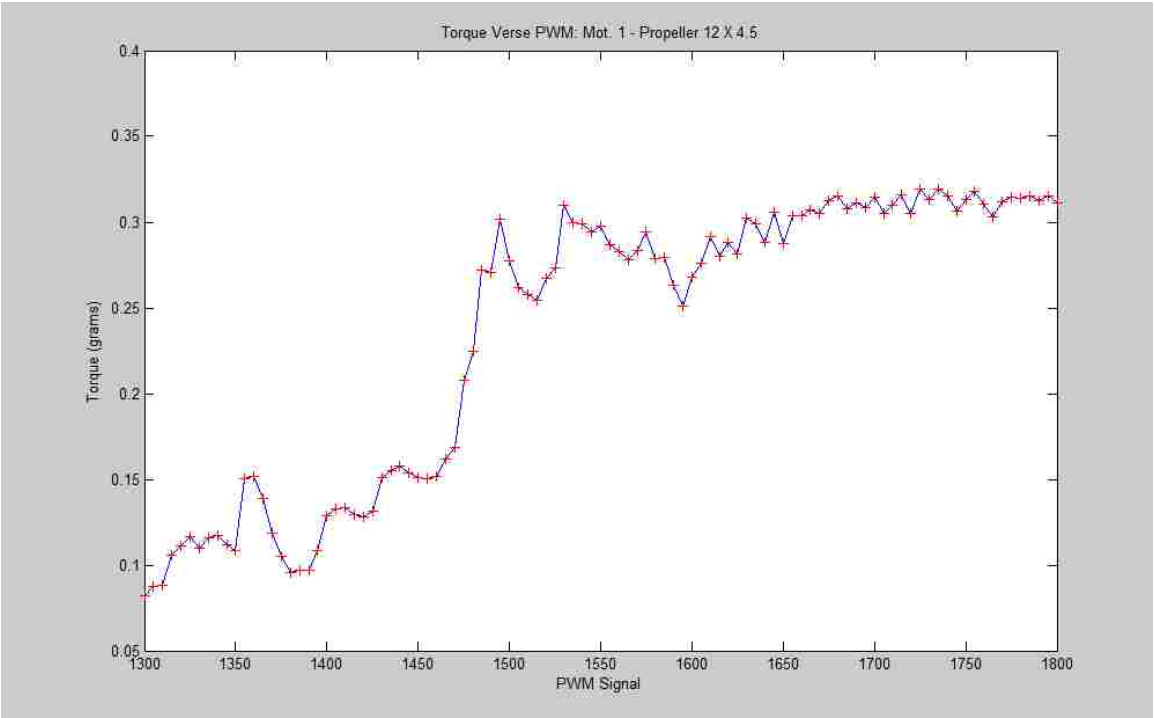


Figure 5-5: PWM Verse Torque Data

Chapter 6: GNC Controller Design

6.1: Numerical Attitude Benchmark Analysis

Before developing a nonlinear attitude Mamdani-type Fuzzy Control for the hopper spacecraft, a simple linear PID attitude stabilization controller was designed and implemented for comparison. A PID control scheme was chosen due to its ease of implementation and successful track record for systems of this type [10]. The general structure of a PID controller can be seen in Figure 6-1.

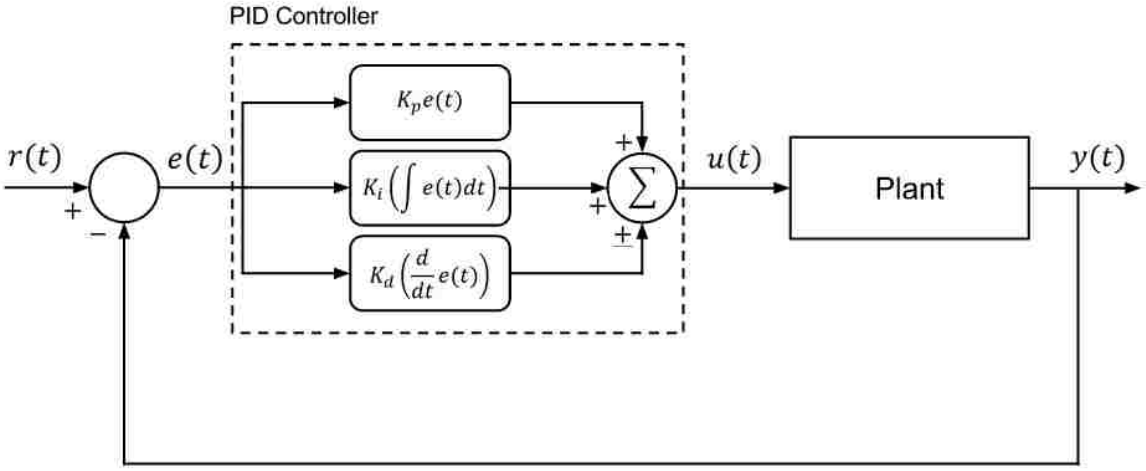


Figure 6-1: Single Input – Single Output PID Block Diagram

For the hopper spacecraft simulator, three independent PID controllers are needed for attitude stabilization, one for each attitude axis. Thus the associated system of control laws take the form:

$$u_1(t) = K_p\tilde{\theta}(t) + K_i\int \tilde{\theta}(t)dt + K_d\left(\frac{d}{dt}\tilde{\theta}(t)\right) \tag{6.1}$$

$$u_2(t) = K_p\tilde{\varphi}(t) + K_i\int \tilde{\varphi}(t)dt + K_d\left(\frac{d}{dt}\tilde{\varphi}(t)\right) \tag{6.2}$$

$$u_3(t) = K_p\tilde{\psi}(t) + K_i\int \tilde{\psi}(t)dt + K_d\left(\frac{d}{dt}\tilde{\psi}(t)\right) \tag{6.3}$$

Where:

$$\tilde{\theta}(t) = \theta_r(t) - \theta(t) \tag{6.4}$$

$$\tilde{\varphi}(t) = \varphi_r(t) - \varphi(t) \tag{6.5}$$

$$\tilde{\psi}(t) = \psi_r(t) - \psi(t) \tag{6.6}$$

The controller's outputs are then "mixed" with the vehicles throttle signal to compute the PWM signal that is sent to each actuator. The mixing process was developed from the vehicle dynamics described in Chapter 5. The following control mixing algorithm was developed:

$$PWM_{M_f} = Throttle(t) + u_1(t) + u_3(t) \tag{6.7}$$

$$PWM_{M_b} = Throttle - u_1(t) + u_3(t) \tag{6.8}$$

$$PWM_{M_l} = Throttle + u_2(t) - u_3(t) \tag{6.9}$$

$$PWM_{M_r} = Throttle - u_2(t) - u_3(t) \tag{6.10}$$

Where the subscript M_f, M_b, M_l, and M_r are reference to the front, back, left, and right actuators respectively. The correlating signal mixing diagram is then:

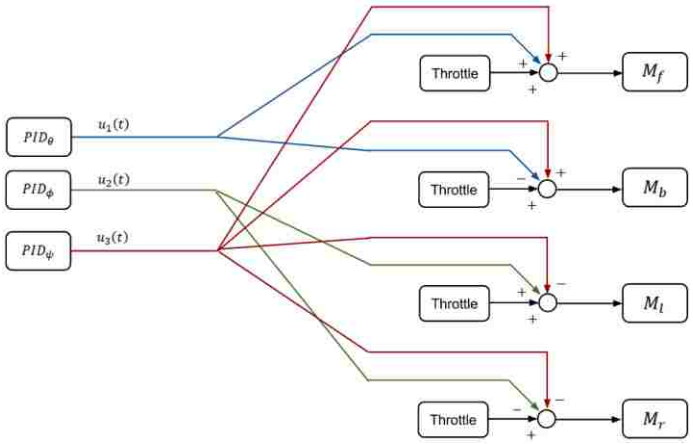


Figure 6-2: PWM Signal Mixing

By applying the control laws described in equations 6.1 – 6.6, the following feedback controller can be developed:

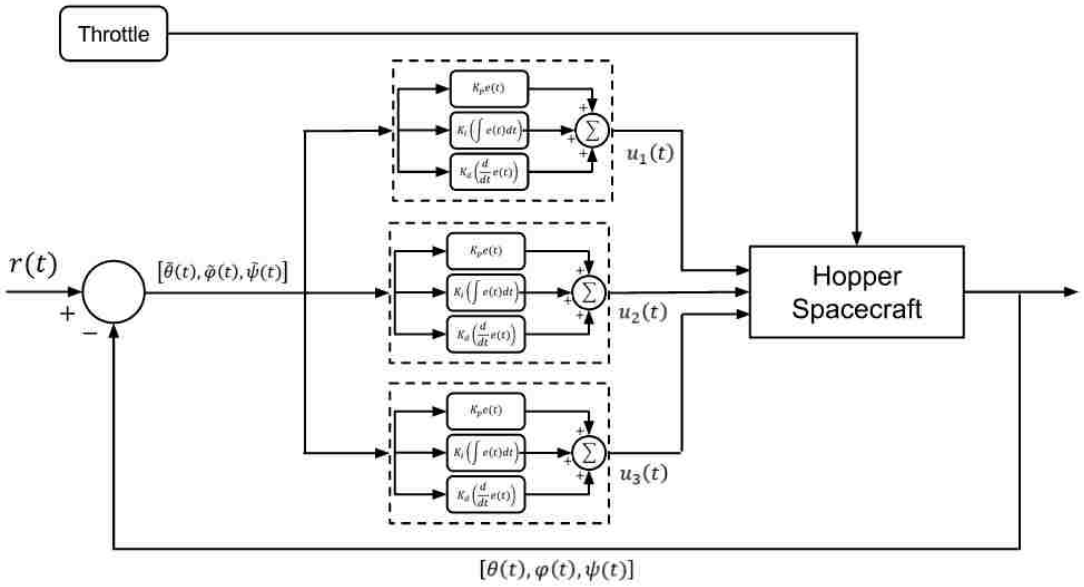


Figure 6-3: PID Attitude Controller

To simulate the controller's performance on the attitude dynamics described in Chapter 5, a Simulink model was constructed. By utilizing Simulink's PID toolbox, with respect to the desired controller performance, a set of PID gains were found. The desired performance of the close-loop system is as follows:

1. Settling Time < 4 Seconds
2. Rise Time < 1 Second
3. % Overshoot < 50

The PID gains that successfully minimized the three design criteria list above was found to be:

	Roll, $\boldsymbol{\theta}$	Pitch, $\boldsymbol{\Phi}$	Yaw, $\boldsymbol{\psi}$
$\boldsymbol{K_P}$	70.78	70.78	7456.70
$\boldsymbol{K_I}$	16.07	16.07	634.89
$\boldsymbol{K_D}$	62.24	62.24	2276.86
N	7.24	7.24	10

Table 6-1: Optimized PID Gains

Assuming an arbitrary reference angle of 0.1 radians, the following dynamic step performance was determined.

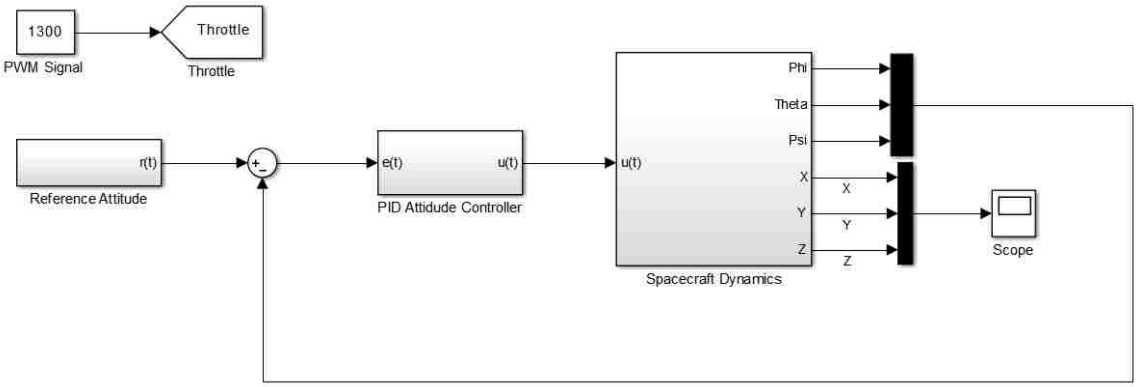


Figure 6-4: PID Simulink Model

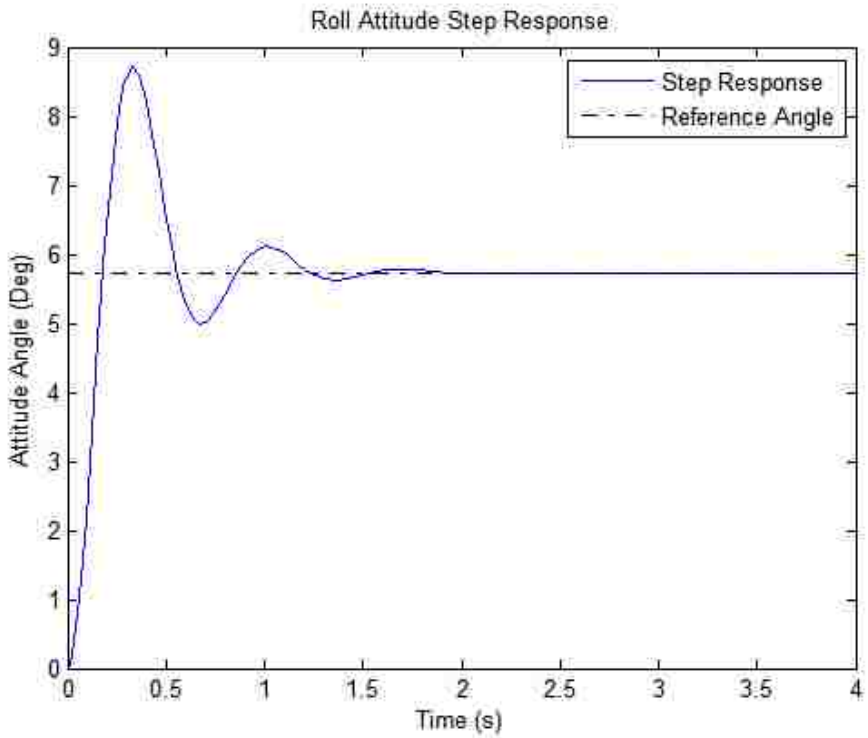


Figure 6-5: PID Roll Response

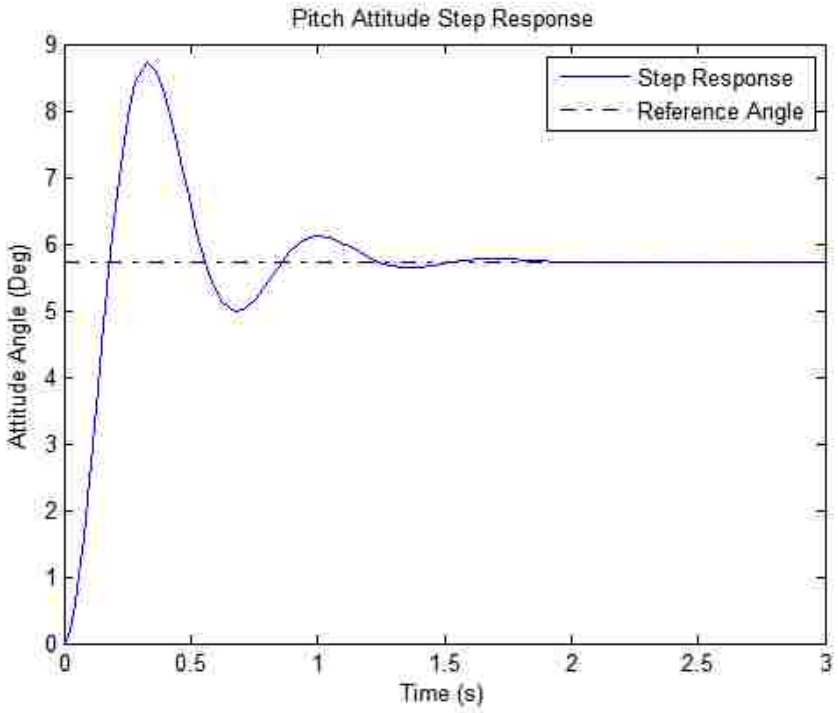


Figure 6-6: PID Pitch Response

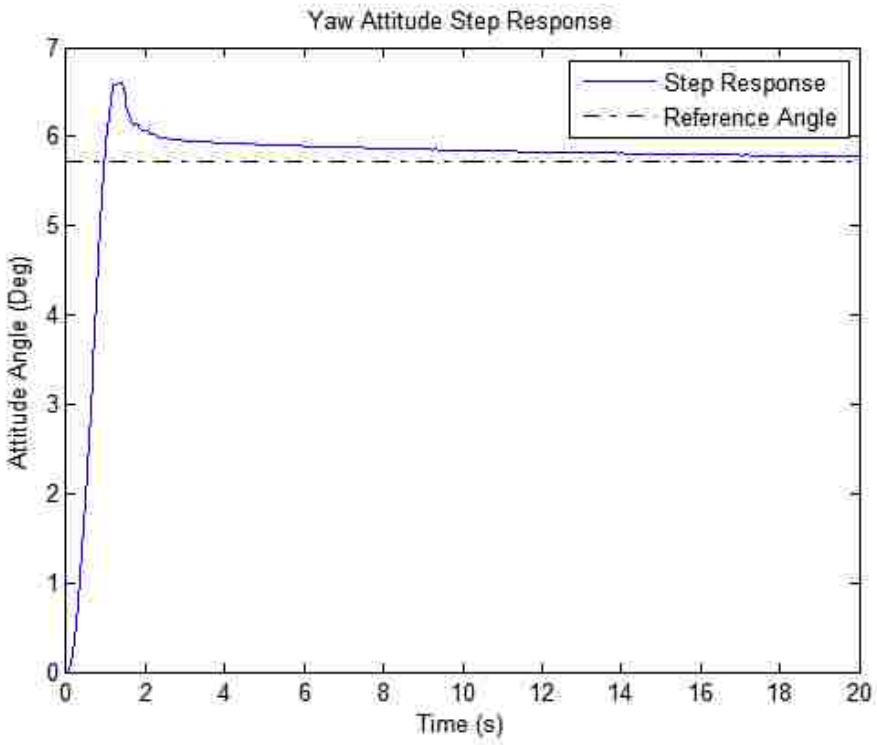


Figure 6-7: PID Yaw Response

From the simulation results, the following benchmarking data was tabulated:

	Roll, $\boldsymbol{\theta}$	Pitch, $\boldsymbol{\Phi}$	Yaw, $\boldsymbol{\psi}$
Step Angle (Rad)	0.1	0.1	0.1
Rise Time (sec)	.25	.25	1
Settling Time (sec)	2	2	20
% Over Shoot	46	46	1

Table 6-2: Attitude Benchmark Analysis

6.2: Nonlinear Fuzzy Attitude Controller Design and Analysis

Now that benchmarking data has been established, a more sophisticated nonlinear Mamdani-Type FLC was applied to the hopper spacecraft to increase performance. Similar to the design of the linear PID controller in section 6.1, three independent Mamdani-type FLCs are used to stabilize the hopper spacecraft's attitude. To reduce complexity, only two inputs will be used, position error and position error rate. From this the general control law can be constructed in the form:

$$u_1(t) = FLC_\theta\left(\tilde{\theta}(t), \frac{d\tilde{\theta}(t)}{dt}\right) \tag{6.11}$$

$$u_2(t) = FLC_\varphi\left(\tilde{\varphi}(t), \frac{d\tilde{\varphi}(t)}{dt}\right) \tag{6.12}$$

$$u_3(t) = FLC_\psi\left(\tilde{\psi}(t), \frac{d\tilde{\psi}(t)}{dt}\right) \tag{6.13}$$

Where, $\tilde{\theta}(t)$, $\tilde{\varphi}(t)$ and $\tilde{\psi}(t)$ are defined as in Equation 6.4 – 6.6. As discussed in Chapter 3, the Mamdani-type FLC has three internal mechanisms that make up the control law, the three mechanisms are fuzzification, inference engine, and defuzzification. The interaction between these mechanisms can be seen in Figure 3.8. Thus, replacing the PID controllers in Figure 6-3 yields our new attitude controller in the form:

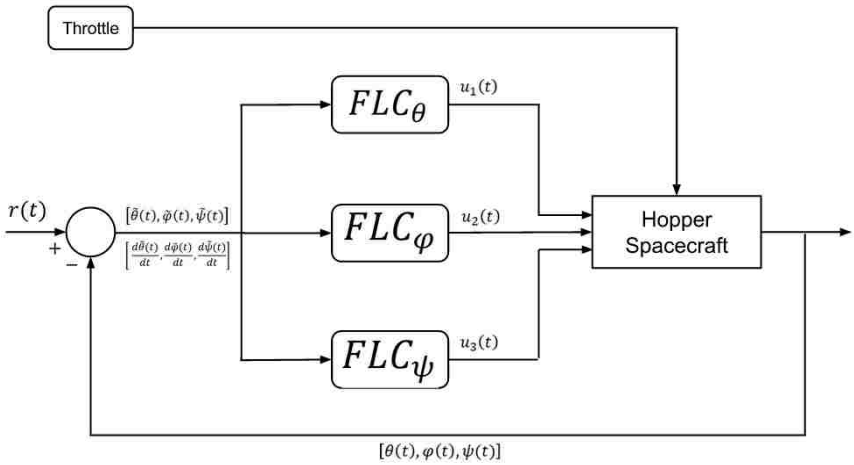


Figure 6-8: Fuzzy Attitude Controller

For the design of the fuzzification mechanism, a few initial assumptions were made to simplify the tuning process and to reduce computational complexity. These design assumptions are:

1. Restrict the input variables to a normalized input universe of [-1, +1].
2. Use only 3 fuzzy sub universes (Negative, Zero, Positive) per input variable.
3. Use only simple and symmetric membership functions.

To be consistent, the defuzzification mechanism was created based on the same initial design assumptions for the establishment of the fuzzy sub universes of the output space. It was decided that the defuzzification mechanism would use the centroid defuzzification method for computing the crisp output. This method was chosen over the others because of reasons discussed in section 3.3.4.

Based on the controller requirements described above it was intuitive to choose triangular and trapezoidal membership functions to describe the input-output space. These membership functions are the simplest to analyze and often are very successful when applied to nonlinear systems. From these design assumptions an initial fuzzy variable space was constructed:

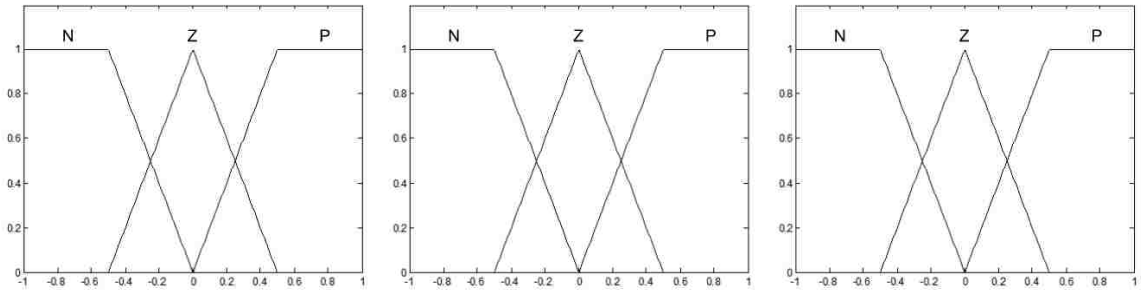


Figure 6-9: Initial Attitude FLC Membership Functions

The associated rule base that was developed for this system is comprised of 9 rules constructed in the form of IF-AND-THEN described in section 3.3.3. In this form, each pair of inputs is uniquely pared with an output state. The rules were constructed based human experience and intuition on the dynamics of this system.

$\dot{e}$ \ e	N	Z	P
N	N	N	Z
Z	N	Z	P
P	Z	P	P

Table 6-3: Attitude FLC Rule Base

Because the input and output variable spaces were restricted to a normalized universe of [1,+1], an additional pre-processing and post-processing component had to be added to the generalized FLC control law. The pre-processing and post-processing components are simply normalizing gains that map the input variables to the normalized input space and map the output variables to the output control single space. These gains are beneficial in the sense they reduce some of complexity of tuning the controller. These tuning gains are hidden from the view by Figure 3-7, but they can be seen in Figure 6-10 below.

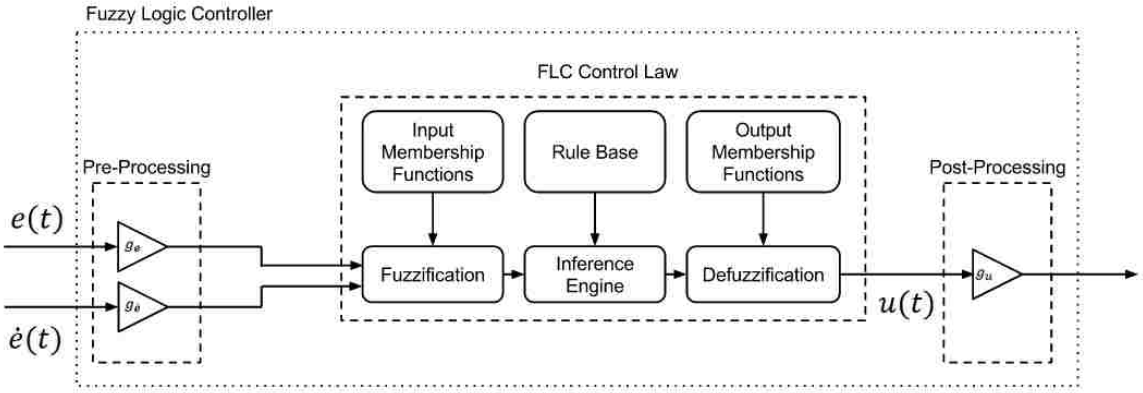
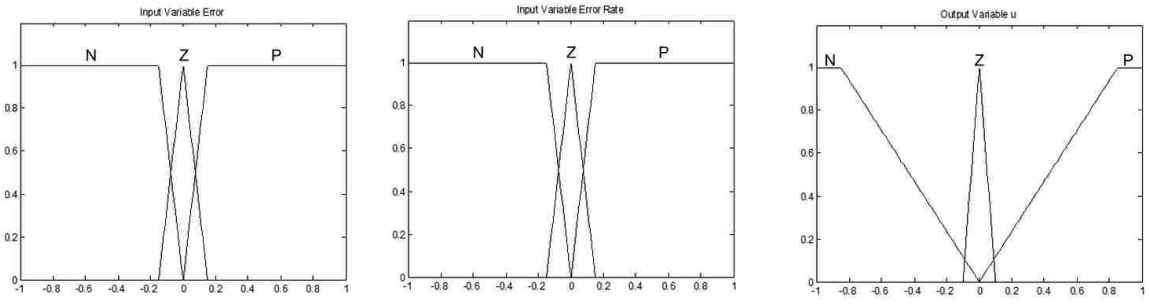


Figure 6-10: FLC with Pre & Post Processing Components

Tuning membership functions for optimized performance can be challenging. Many smart tuning process like the ones in [11], [12], [13] have been developed to efficiently tune FLC's. Because the membership structure of this controller was left minimal, such complex tuning methods are not needed. Instead, for tuning, it was assumed that the membership functions would be left symmetric and the negative and positive membership functions would be left as mirrored images of each other. This simplifies the tuning process and thus only requires the shapes to be "stretched" or "compressed". Thus, after applying this tuning method to the input output membership functions, the following tuned membership functions where found:



(a). Input Variable Error (b). Input Variable Error Rate (c). Output Variable u

Figure 6-11: Tuned Attitude FLC Membership Functions

Also a stochastic tuning process was done to tune both the pre-processing and post-processing normalization gains. The choice of gains are tabulated below:

	Roll	Pitch	Yaw
g_e	2.5	2.5	2
$g_{\dot{e}}$	3.33	3.33	10
g_u	80	80	300

Table 6-4: Fuzzy Normalizing Gains

The tuned FLC was then tested on the simulated hopper spacecraft's dynamics, the results are shown in Figure 6-12 – 6-14.

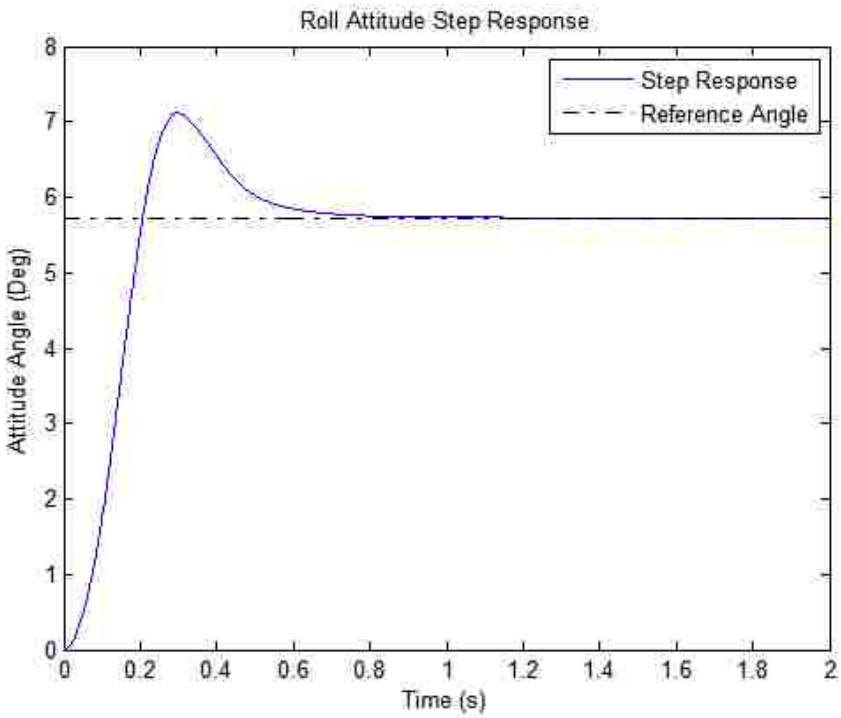


Figure 6-12: Fuzzy Roll Response

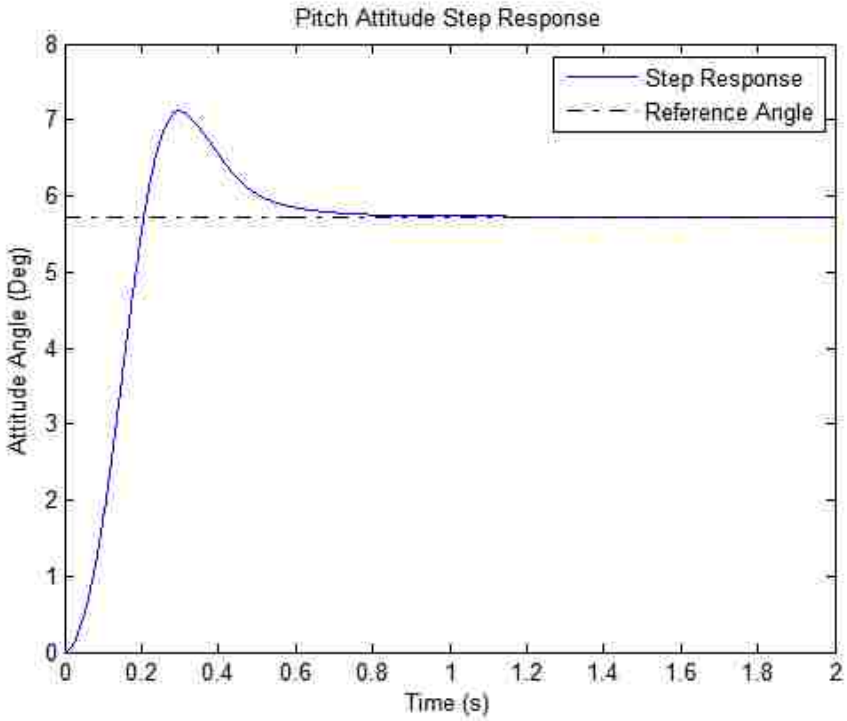


Figure 6-13: Fuzzy Pitch Response

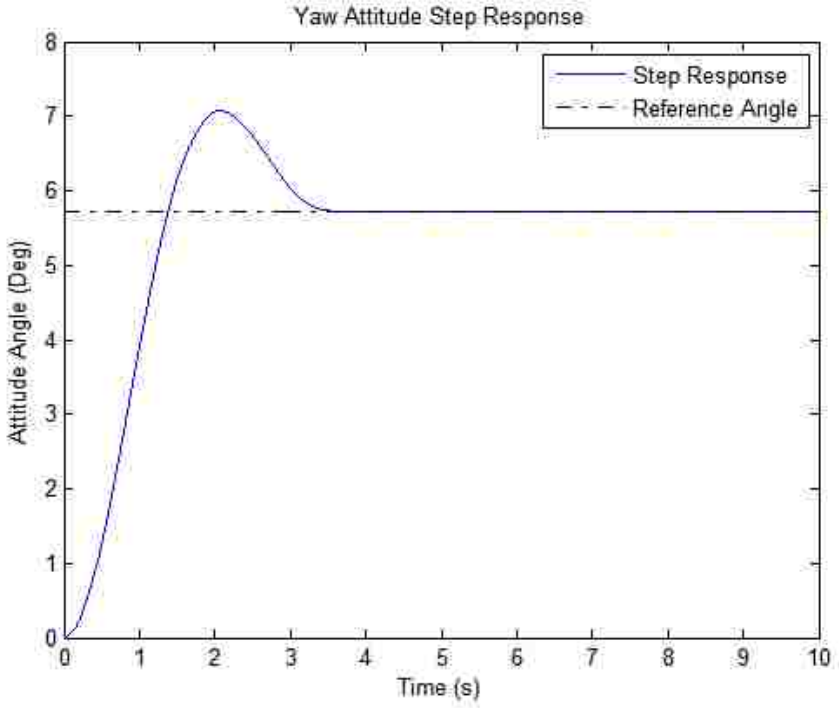


Figure 6-14: Fuzzy Yaw Response

From the fuzzy step response simulation data, fuzzy benchmark data was collected and tabulated.

	Roll, $\boldsymbol{\theta}$	Pitch, $\boldsymbol{\Phi}$	Yaw, $\boldsymbol{\psi}$
Reference Angle (Rad)	0.1	0.1	0.1
Rise Time (sec)	.20	.20	1.1
Settling Time (sec)	1.2	1.2	4
% Over Shoot	2	2	2

Table 6-5: Fuzzy Benchmark Analysis

Compared to the PID attitude controller, the fuzzy attitude controller offers significant performance improvements in all three of the controller design requirements. In terms of roll and pitch, the fuzzy controller decreased the rise time by 105%, decreased the settling time by 66% and reduced the % over shoot by 210%. In terms of yaw, the fuzzy controller had a slight increase in rise time and % overshoot of about 10%, but offered significant improvements decreasing the settling time by over 400%. From the simulation results, it is clear that properly implementing a fuzzy logic attitude controller will offer significant improvements over the classical PID style controller that was implemented previously. These results have found consistent with the work that has been done in respect to fuzzy controllers applied to similar plants [14], [15].

6.4: Fuzzy Attitude Implementation Analysis and Results

The simulation benchmark analysis has proven the Mamdani-type FLC is theoretically very effective in the control of the spacecraft simulators dynamics as defined in Chapter 5. To further analysis the robustness of the controller that was developed, two real world design issues, sensor noise and time delay, must be taken into consideration.

Sensor noise, is always a fundamental concern with any control system. An abundance of sensor noise can lead to major design issue, including increased rise time, increased settling time, and ultimately can drive the system unstable. Even though sensor noise is problematic, there are ways to work around sensor noise, these methods include state estimation and filtering.

Adding time-delay to a system, is often the easiest way to drive a system unstable. From the introduction discourse of linear control system, it is well known that time-delay impacts stability by increasing phase delay, and excessive phase delay leads to instability. For the purposes of implementation, the system has several sources of time-delay. The major areas of concern is senor sampling lag, computational lag, and actuator signal lag. If the total time-delay is too large, the system will become unstable.

Ultimately, the amount of time-delay and sensor noise that is acceptable is dependent upon the system. Simulation analysis will be conducted to determine design requirements for the flight electronics to ensure the attitude controller remains robust.

6.4.1: Time Delay Analysis

As a rule of thumb, the amount of time-delay that is acceptable to a system is often approximated for first iterations as:

$$T_d = \frac{1}{20 * \omega_n} \tag{6.14}$$

Where ω_n is the fastest natural frequency of the closed-loop system.

Since FLC controllers are a nonlinear mapping between sets with no analytical form, determining the natural frequency of the closed loop system directly is impossible. To overcome this issue, system identification can be done to approximate the close-loop transfer function from the systems step response.

For the closed-loop transfer function approximation, the system was assumed to be linear, and thus the yaw and roll axis were assumed to be stable, thus:

$$\ddot{\phi} = \dot{\phi} = \phi = 0$$

$$\ddot{\psi} = \dot{\psi} = \psi = 0$$

From these linearization assumptions the simplified angular dynamics become:

$$\ddot{\theta}(t) = \frac{\tau_\theta(t)}{I_{yy}}$$

Applying the FLC attitude controller designed in Chapter 6.1, with a reference angle of 0.3 radians the following step response was simulated:

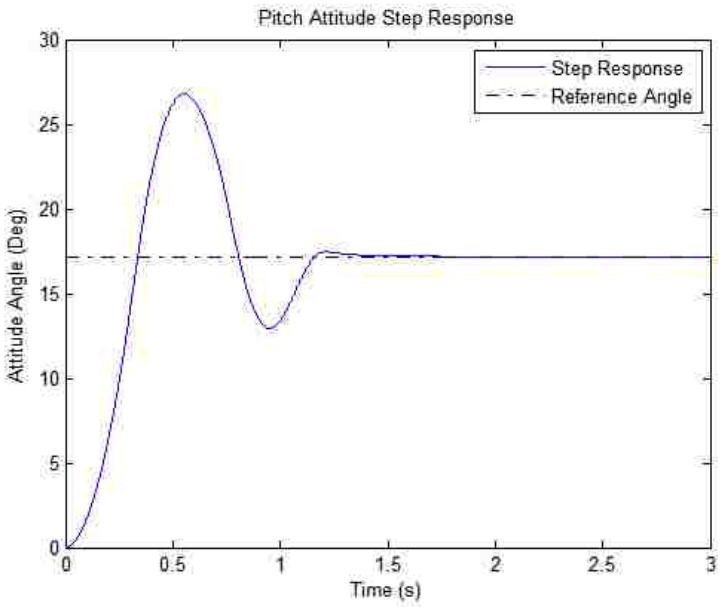


Figure 6-15: System Identification Reference Performance

By analyzing the input reference angle with respect to the systems angular position, the following approximated linear transfer function was determined for the closed loop response:

$$\frac{\Theta(s)}{\Theta_r(s)} = \frac{504.8s^3 + 3387s^2 + 7.843E^4s + 6.362E^5}{s^6 + 12.94s^5 + 369.2s^4 + 3046s^3 + 3.432E^4s^2 + 1.399E^5s + 6.372E^5}$$

Where:

$$E^n = 10^n$$

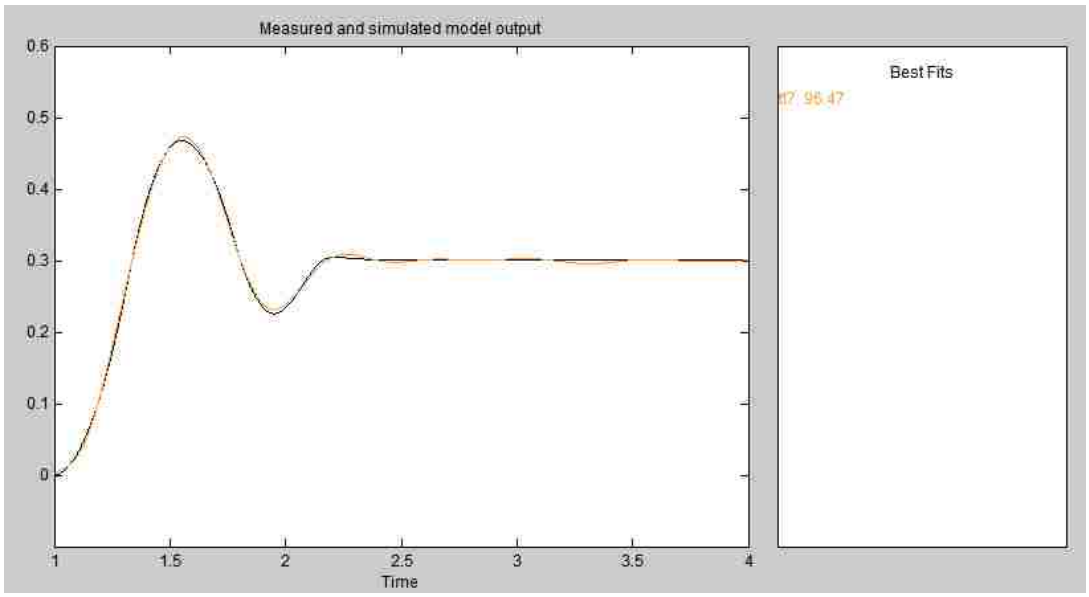


Figure 6-16: System Identification Fit Results

From the transfer function, the range of natural frequencies were determined as:

$$\max(\omega_n) = 13.59$$

$$\min(\omega_n) = 6.0921$$

From equation 6-14, the corresponding range of acceptable time delay is thus:

$$\max(T_d) = 0.008 = 125\ Hz$$

$$\min(\omega_d) = 0.004 = 250\ Hz$$

To prove the range of time delays are acceptable, simulations where conducted in which a unit delay was added to the system to simulate the effects of IMU sampling rate lag. From the results in Figure 6-17 and Figure 6-18, it is clear that the first round iterations of acceptable time delay yields results approximately equivalent results to the bench mark analysis shown in Figures 6-15.

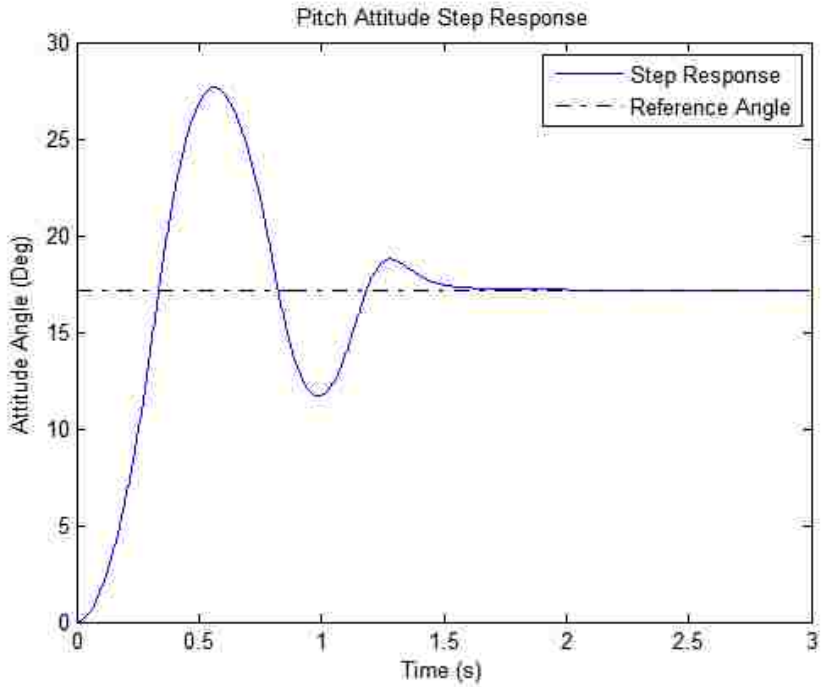


Figure 6-17: Close-Loop Response with Td=0.004

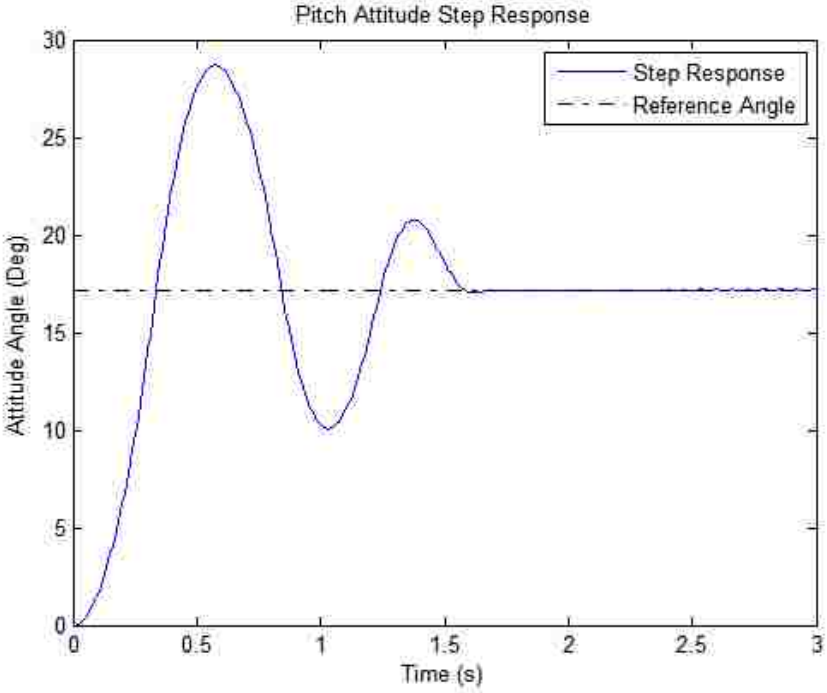


Figure 6-18: Close-Loop Response with Td=.008

6.4.2: Sensor Noise Analysis & Time-Delay

Sensor noise is not as analytically easy to analyze as time delay; therefore, the analysis must be done through simulation to determine requirements to ensure stability. As a starting point, almost all natural sensor noise is Gaussian and thus will be assumed as so in simulation. To assess the effects of sensor noise on the closed-loop system, a randomized Gaussian distributed noise was added to the angular position "measurement". The amplitude of the high frequency noise was increased until instability was reached. From simulation the acceptable range of high frequency sensor noise is below ± 0.5 without suffering significant performance degradation. For the simulation experiments, it was assumed that the system time delay and sampling rate was 0.004.

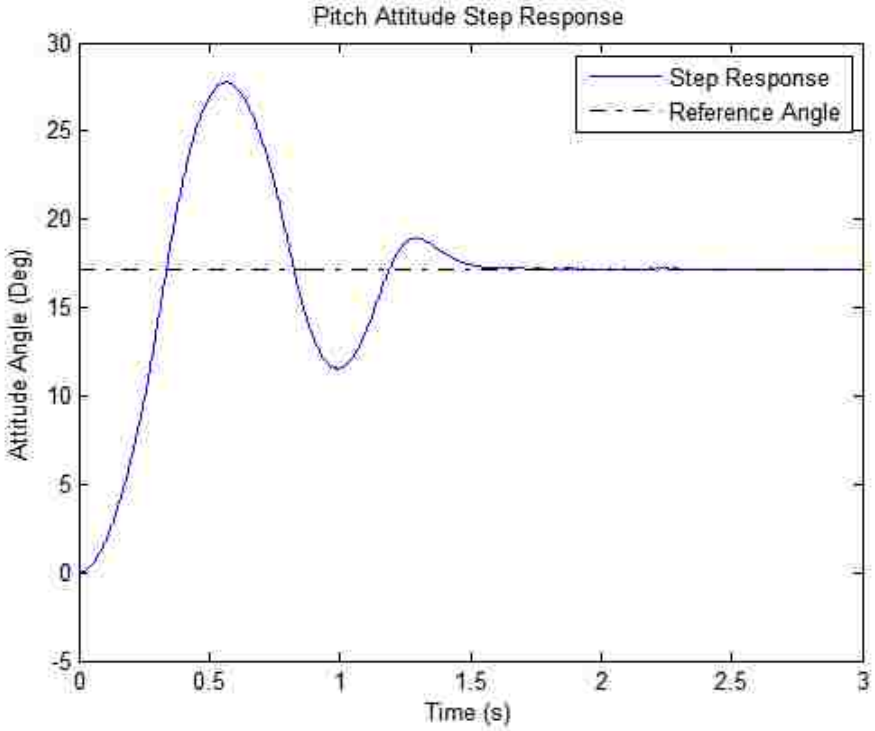


Figure 6-19: Sensor Noise Simulation (Max Noise $\pm 0.01\ deg$)

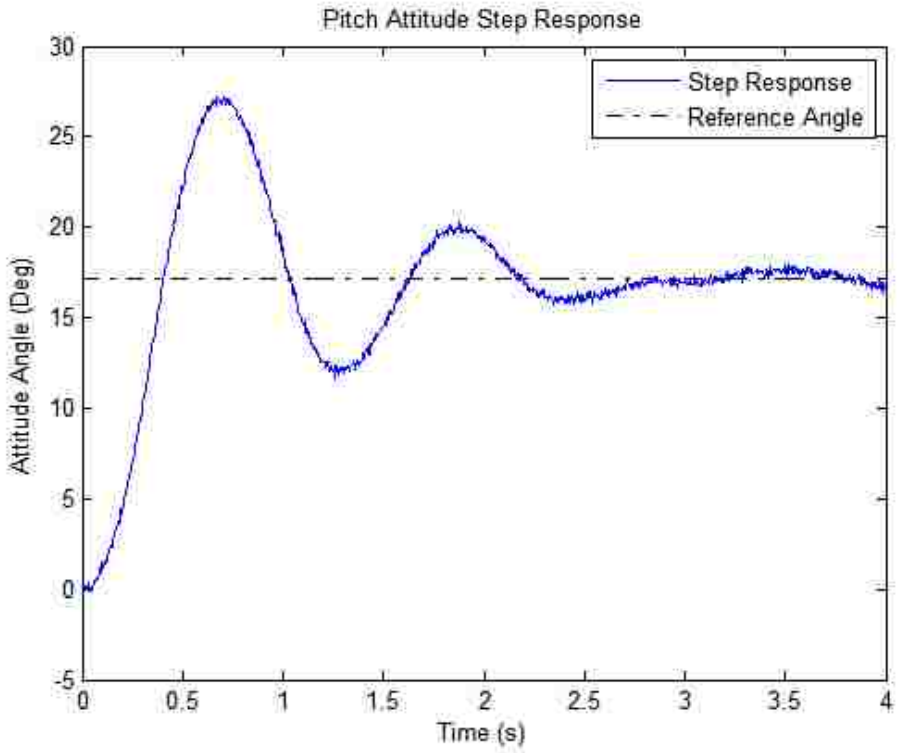


Figure 6-20: Sensor Noise Simulation (Max Noise $\pm 0.5\ deg$)

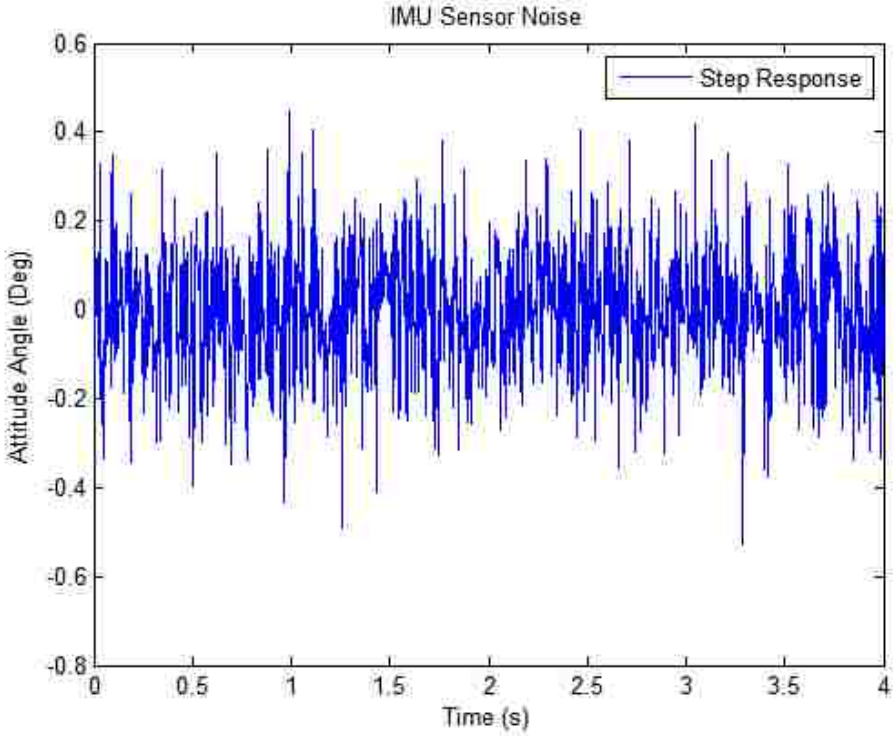


Figure 6-21: Gaussian Sensor Noise (Max Noise $\pm 0.5\ deg$)

6.4.3: Fuzzy Attitude Stabilization Implementation Results

From the results of Chapter 6.4.1 and Chapter 6.4.2, critical design specs were determined to ensure controller robustness. These design specifications ultimately will determine if the chosen flight hardware will be effective in controlling the system. The hardware design specifications are tabulated below:

Parameter	Max Spec.
Total Time Delay (sec)	0.008
IMU Noise (Deg)	±0.5

Table 6-6: Flight Hardware Design Specifications

The programming language chosen for the flight controller was Python. In Appendix C, the python code can be seen for the Mamdani-type FLC controller developed in Chapter 6.1. Careful analysis of the flight hardware and software revealed that the total time delay of the system was significantly over the design specifications shown in Table 6-6. The total observed time delay of the flight electronics was 0.02. With this amount of time delay in the system, the simulation predicts the closed-loop system will be unsuccessful at converging to some desired reference angle. To prove that simulations accuracy, the real closed-loop system was tested and compared to the simulation.

Due to safety concerns of an unstable flying system, it was decided to constrain the 6 DOF hopper spacecraft simulator system and test only a single axis. The simple test rig can be seen in Figure 6-22.

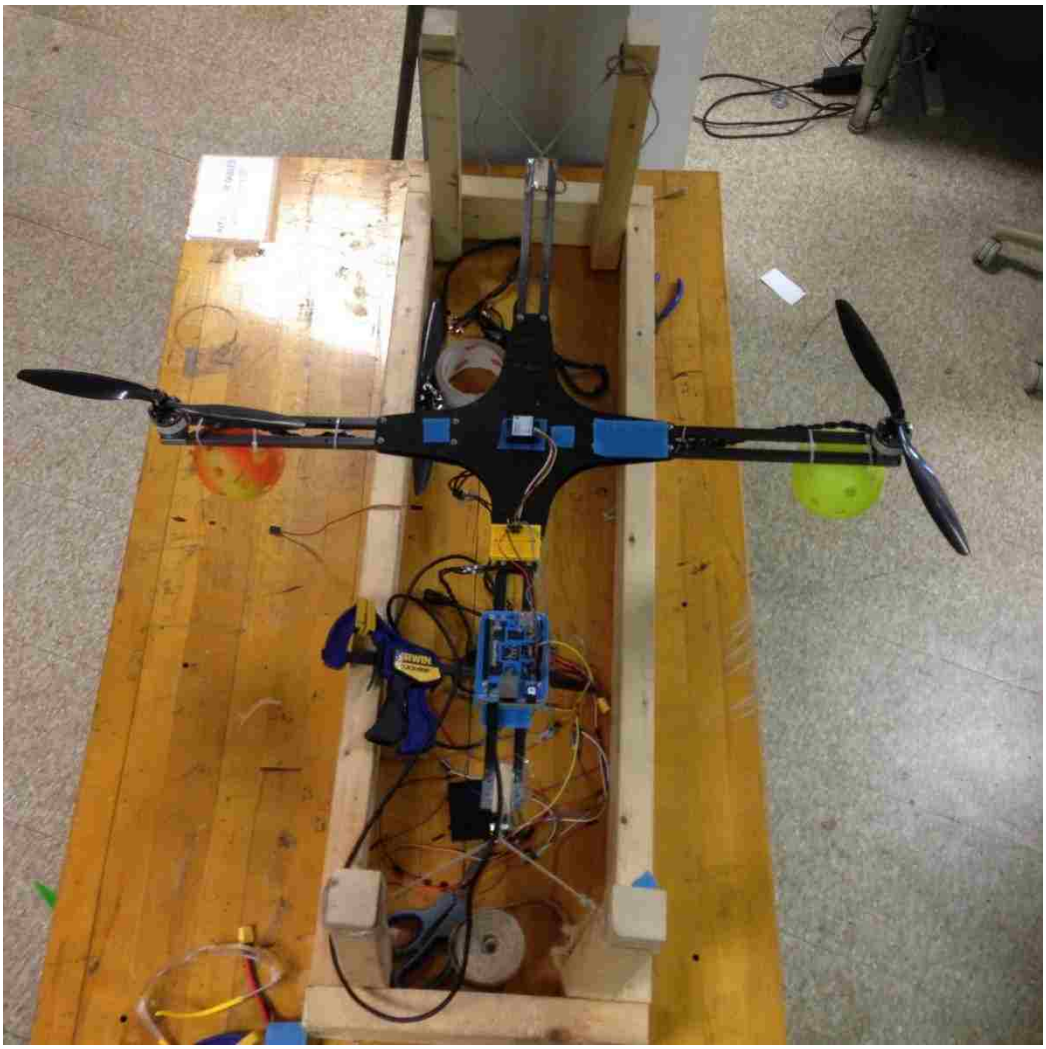


Figure 6-22: Single Axis Test Stand

By comparing the actual hopper simulator to the simulation with 0.02 second time delay, the simulation results were determined accurate. The results of the experiment can be seen in Figure 6-23. In the figure provided, the green data is for the simulation results, and the blue data set is for the actual simulator, the dotted line is the desired reference angle.

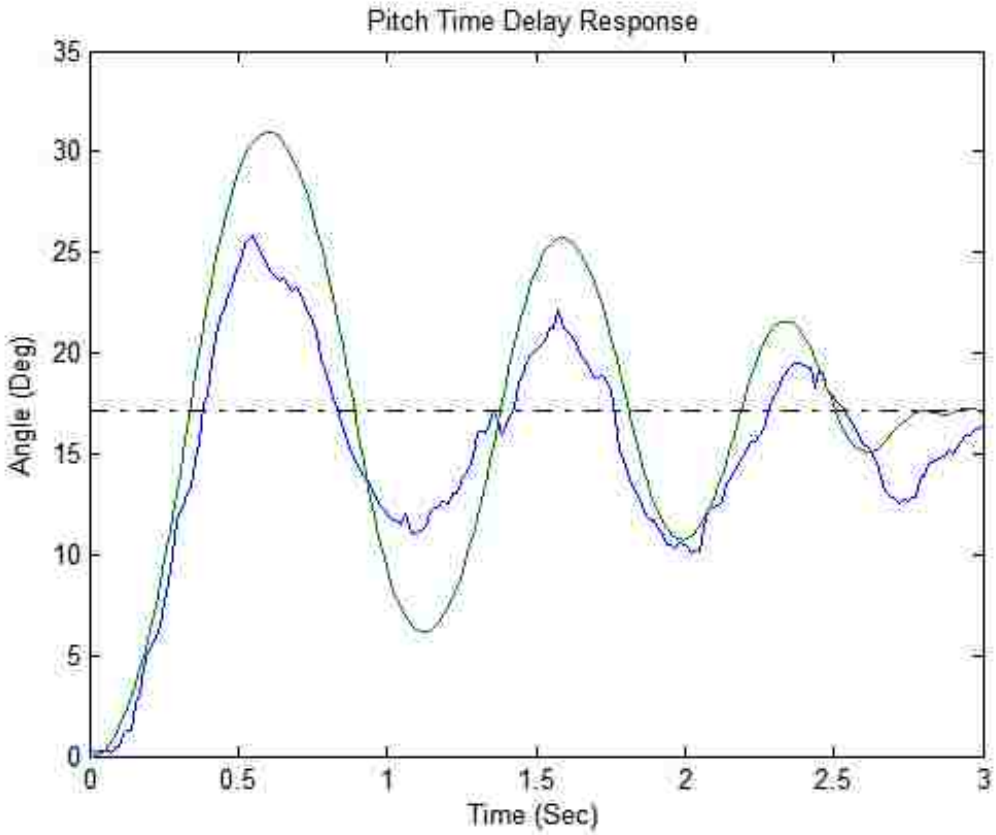


Figure 6-23: Simulation & Actual System Response

Chapter 7: Future Work

7.1: Trajectory Control

Due to the effectiveness of the fuzzy attitude controller, work began in developing a high level guidance and navigation system capable of moving the hopper spacecraft simulator between two points A and B, given some predefined trajectory; see Figure 7-1.

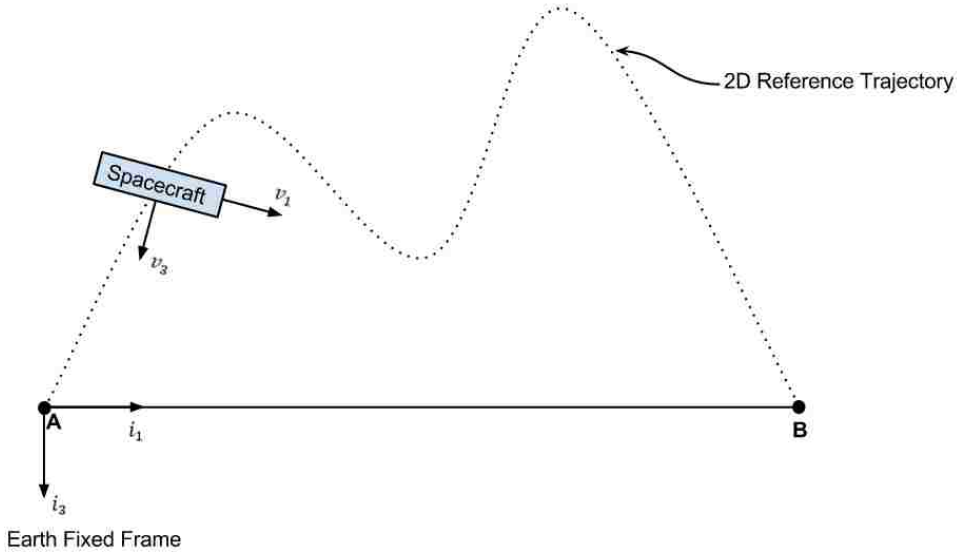


Figure 7-1: 2-D Hop Trajectory Example

To begin the developing of a trajectory controller, a "cascade" Mamdani-type FLC is proposed. A cascade control law is used to send a reference angle command to the attitude controller developed previously while controlling the vehicles altitude through throttle control.

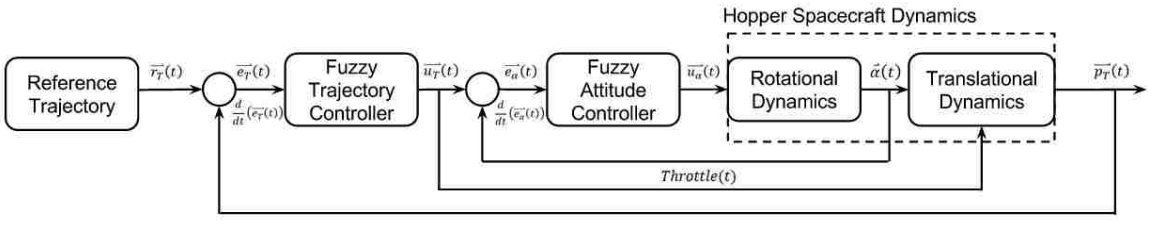


Figure 7-2: Cascade Fuzzy Trajectory Controller

Like the attitude controller, the cascade trajectory controller requires a single controller per translational coordinate (X,Y,Z). For the purpose of developing a simplified control law for testing, a 2-D trajectory controller was developed. The 2-D trajectory controller assumes that the hopper spacecraft will first rotate to a reference yaw angle, thus the vehicle is always in the direction of the desired new location. This style of controller is similar to the controllers used by aircrafts, where the vehicle's heading is kept pointing to the direction of its desired location. The signals shown in Figure 7-2 are defined as,

Where:

$$\overrightarrow{r_T}(t) = \begin{bmatrix} r_x(t) & r_y(t) & r_z(t) \end{bmatrix}^T$$

$$\overrightarrow{p_T}(t) = [x(t) \quad y(t) \quad z(t)]^T$$

$$\overrightarrow{e_T}(t) = [r_x(t) - x(t) \quad r_y(t) - y(t) \quad r_z(t) - z(t)]^T$$

$$\overrightarrow{u_T}(t) = \begin{bmatrix} r_\phi(t) & r_\theta(t) & r_\theta(t) \end{bmatrix}^T$$

$$\vec{\alpha}(t) = [\phi(t) \quad \theta(t) \quad \psi(t)]^T$$

$$\overrightarrow{e_\alpha}(t) = [r_\phi(t) - \phi(t) \quad r_\theta(t) - \theta(t) \quad r_\psi(t) - \psi(t)]^T$$

$$\overrightarrow{u_\alpha}(t) = [u_\phi(t) \quad u_\theta(t) \quad u_\psi]^T$$

Where the simple fuzzy trajectory controller is in the form:

$$r_\theta(t) = FLC_X\left(\tilde{y}(t) * g_y, \frac{d\tilde{\theta}(t)}{dt} * g_{\dot{y}}\right) * g_{r_\theta}$$

$$r_\psi(t) = \sin^{-1}\left(\frac{B_y - A_y}{B_x - A_x}\right)$$

$$Throttle(t) = FLC_Z\left(\tilde{y}(t) * g_Z, \frac{d\tilde{\theta}(t)}{dt} * g_{\dot{Z}}\right) * g_{Throttle}$$

And the pre-processing and post-processing gains were chosen as:

	X	Y	Z
g_e	3	3	3
$g_{\dot{e}}$	8	8	8
g_u	-0.1	-0.1	500

From the cascade controller the following simulation results were obtained:

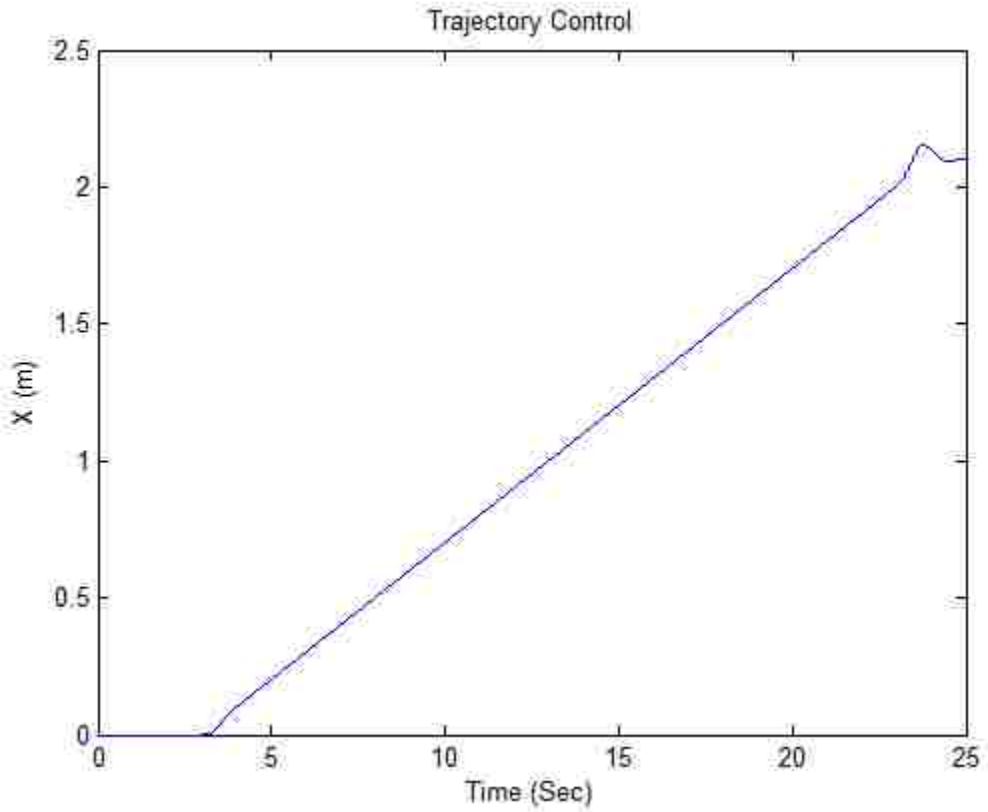


Figure 7-3: Trajectory Control Simulation, Time Verse X

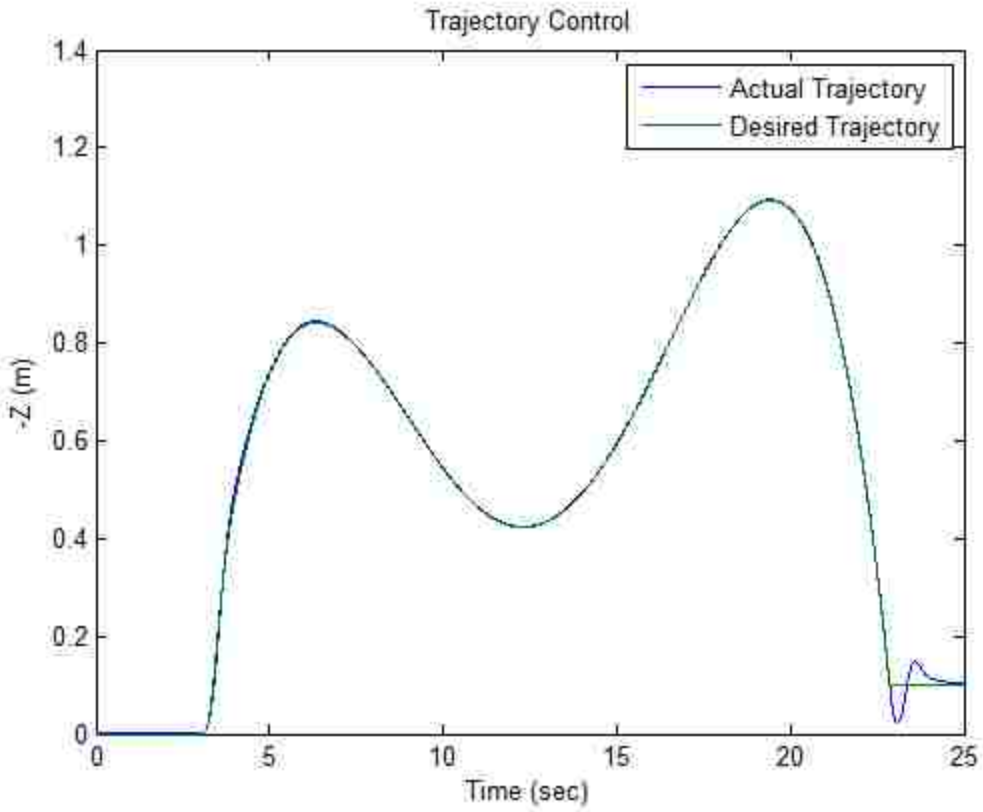


Figure 7-4: Trajectory Control Simulation, Time Verse –Z

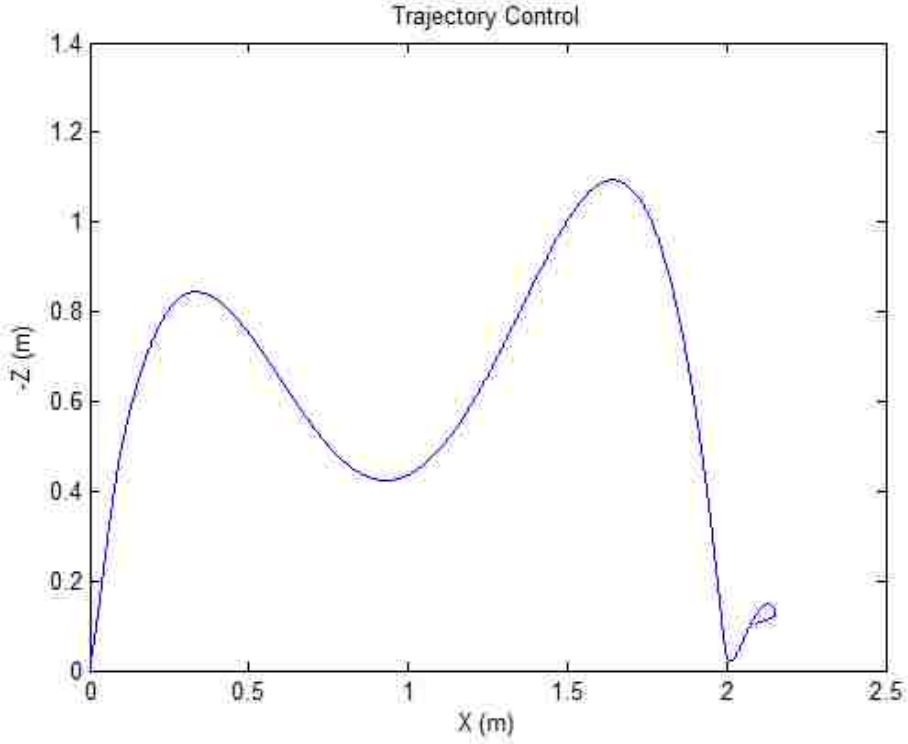


Figure 7-5: Trajectory Controller Stimulation, X Verse -Z

To similarly test the trajectory controller's robustness, other trajectories were simulated. For a purely parabolic trajectory, the following simulation results were obtained:

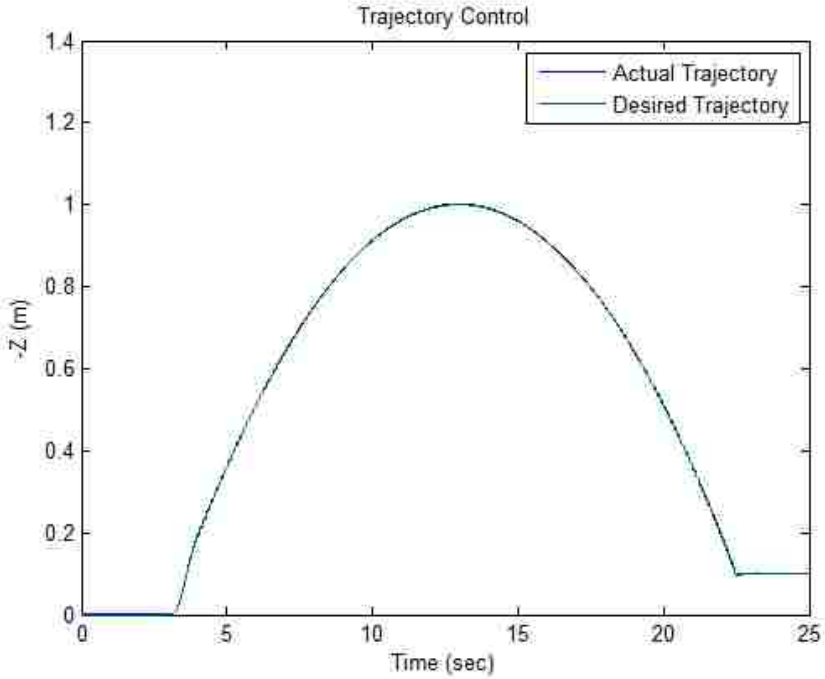


Figure 7-6: Parabolic Trajectory Time Verse –Z

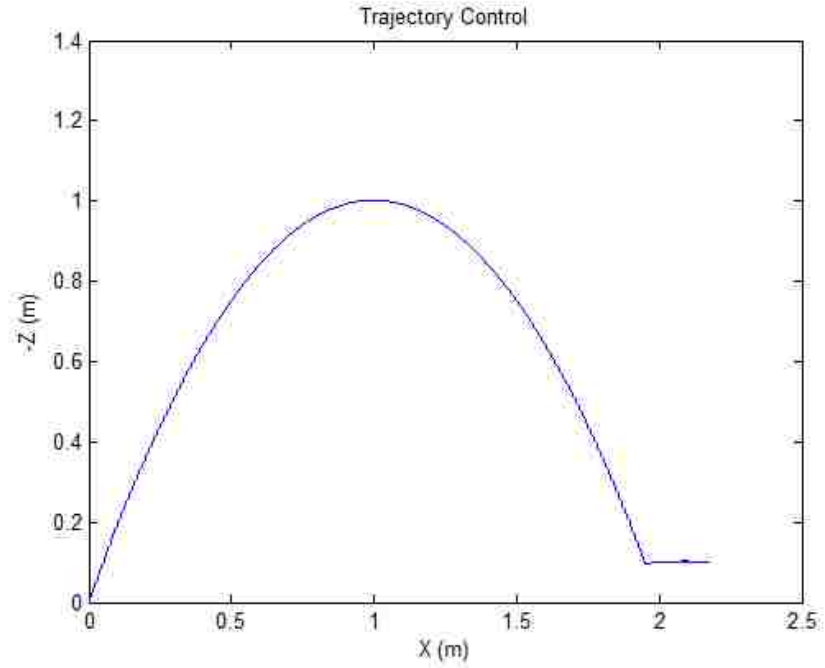


Figure 7-7: Parabolic Trajectory X verse -Z

7.2: Trajectory Control Implementation

To be able to successful implement the cascade trajectory controller described in the previous section, the flight electronics must be revisited. As seen in Chapter 6, the design specifications required to ensure attitude stability were not met by the initial selection of flight electronics. Ultimately, without flight electronics capable of robustly controlling the vehicles attitude, a full GNC system will not be possible. To ensure the required design specs are met, the flight control board should be replaced with a real-time control board. One such board which seems ideal for this application is the C2000 real-time control device by Texas Instruments. The C2000 is a 32-bit microcontroller based on industry leading C28x CPU. Not only is the C2000 control board designed for real time computing, it also can be programmed via Simulink directly. Thus all the initial design work that was done could be directly imported and used as the control software. The C2000 will ultimately be able to effectively control the vehicles attitude by reducing the time delay, and through the use of Simulink, the amount of time required for control software development will also be decreased.

Figure 7-8: C2000 LaunchPad

References

[1] Mucasey, E. P. K. (2013). Design, Fabrication, and Testing of a Hopper Spacecraft Simulator: Lehigh University.

[2] Ross, T. J. (2009). Fuzzy logic with engineering applications: John Wiley & Sons.

[3] Yager, R. R., & Filev, D. P. (1994). Essentials of Fuzzy Modeling and Control: Wiley.

[4] Nguyen, H. T., & Prasad, N. R. (1999). Fuzzy Modeling and Control: Selected Works of Sugeno: Taylor & Francis.

[5] Material Selection Charts. (2014). From http://www-materials.eng.cam.ac.uk/mpsite/interactive_charts/

[6] 'Antigravity MT2814 - Antigravity Motors - T-MOTOR. (2014). From http://www.rctigermotor.com/html/2013/Antigravity_Motors_0916/77.html

[7] Power density - Wikipedia, the free encyclopedia. (2014). from http://en.wikipedia.org/wiki/Power_density

[8] TP2100-4SPP25. (2014). From http://www.thunderpowerrc.com/Products/2700-mAh_5/TP2100-4SPP25

[9] Spring Calculation (2014). From http://www.mitcalc.com/doc/springs/help/en/springs.htm

[10] Luukkonen, T. (2011). Modelling and control of quadcopter. *Independent research project in applied mathematics, Espoo*.

[11] Herrera, F., Lozano, M., & Verdegay, J. L. (1995). Tuning fuzzy logic controllers by genetic algorithms. *International Journal of Approximate Reasoning, 12*(3), 299-315.

[12] Chung, H.-Y., Chen, B.-C., & Lin, J.-J. (1998). A PI-type fuzzy controller with self-tuning scaling factors. *Fuzzy Sets and Systems, 93*(1), 23-28.

[13] Mudi, R. K., & Pal, N. R. (2000). A self-tuning fuzzy PI controller. *Fuzzy Sets and Systems, 115*(2), 327-338.

[14] Bhatkhande, P. S. (2014). REAL TIME FUZZY CONTROLLER FOR QUADROTOR STABILITY CONTROL.

[15] Raza, S. A., & Gueaieb, W. (2010). Intelligent Flight Control of an Autonomous Quadrotor. *InTech*.

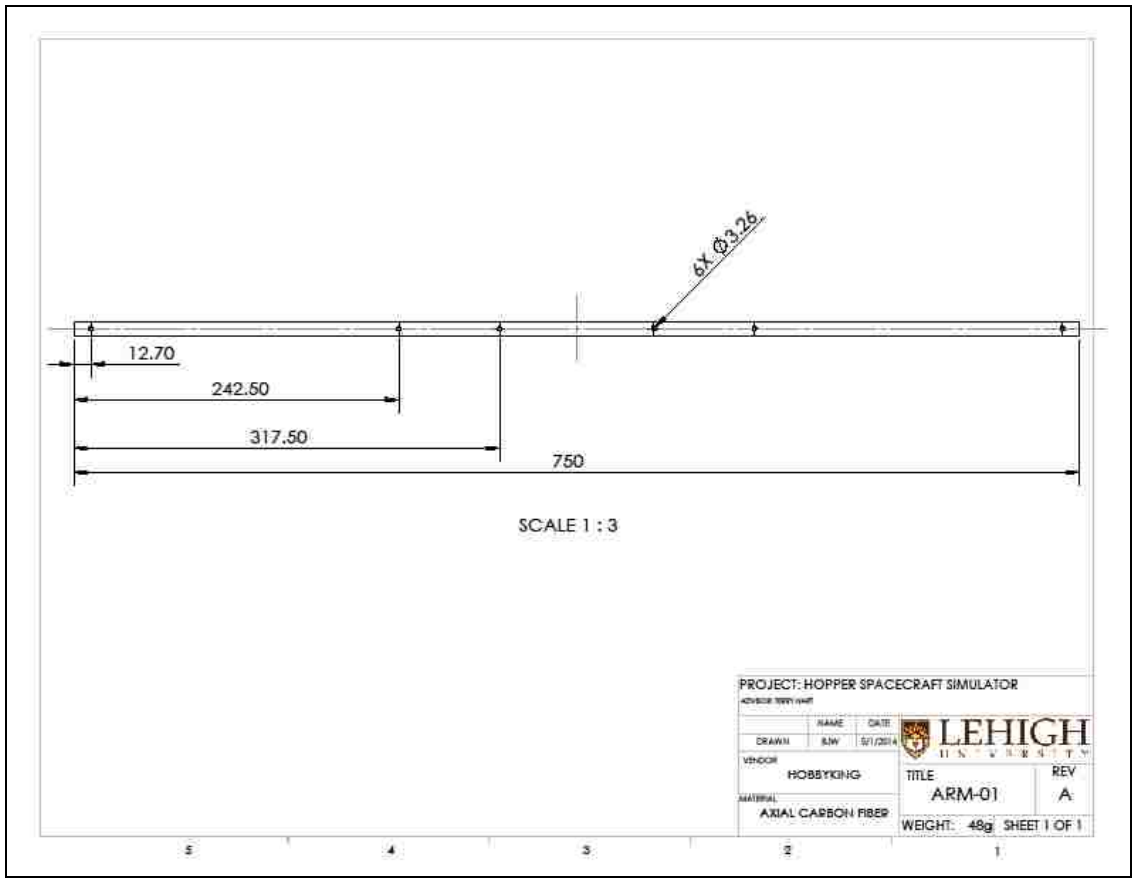
Appendix A: Bill of Materials

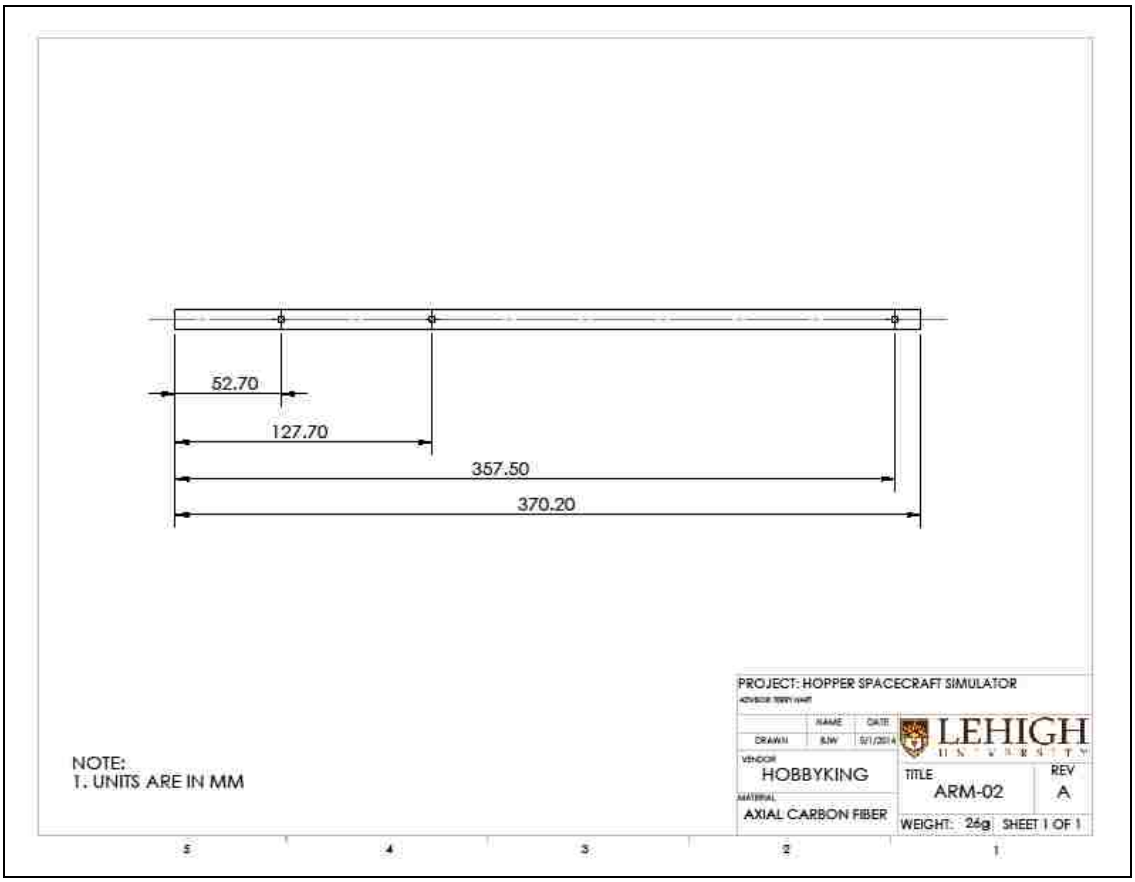
Comprehensive Bill of Material & Mass Estimate				
Description	Vendor	QTY	Cost/Unit	Mass (g)
Frame*	See Frame BOM	1	$151.17	300
MT2814	T-Motor	4	$65.90	120
IMU	Pololu	1	$130.00	6
Qbrain	HobbyKing	1	$29.92	112
Pololu Board	Pololu	1	$6.00	5
Beaglebone Black	Sparkfun	1	$45.00	80
4s 25C Lipo	ThunderPower	2	$62.99	197
Total Cost	**$751.67**			
Total Mass (g)	**1377**			

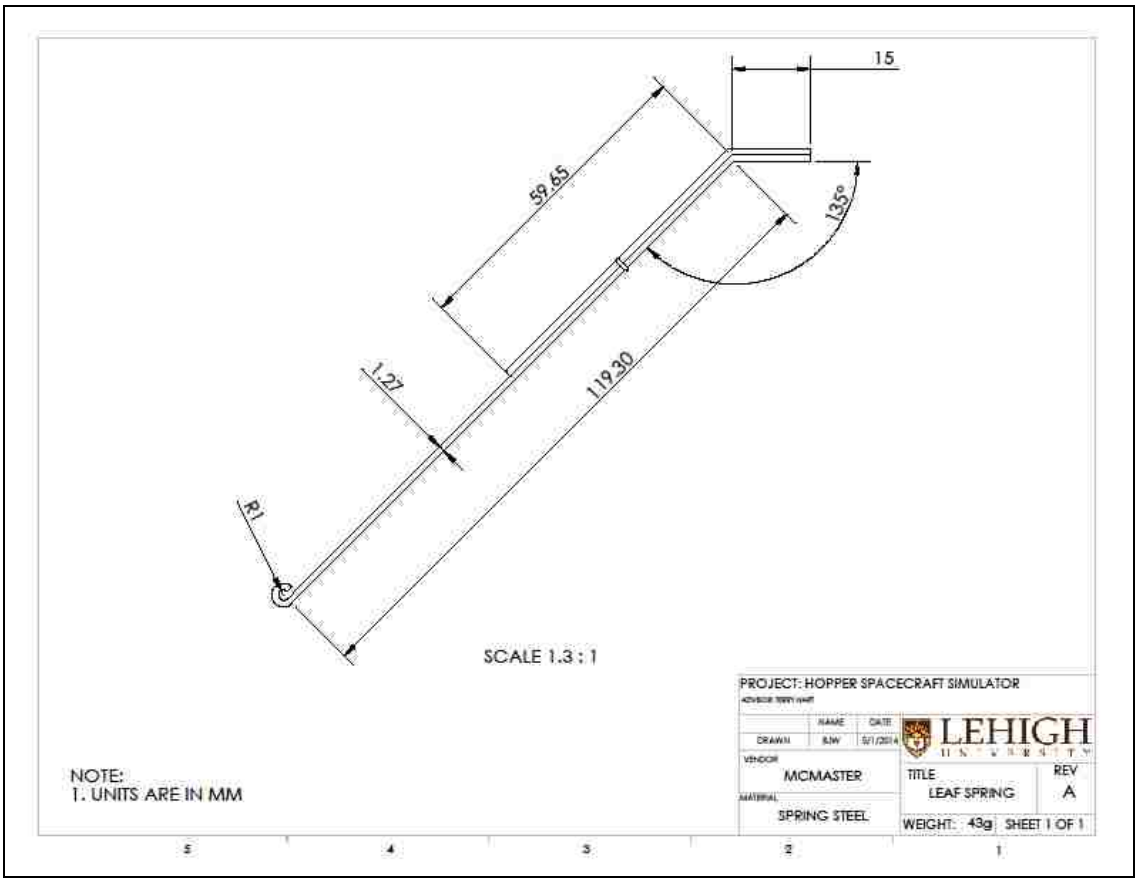
*From Solidworks

Frame Bill of Material				
Description	Vendor	QTY	Cost/Unit	Mass (g)
CF Square Tubing (750x6mm)	HobbyKing	4	3.12	48
Machined CF Plate	Mcmaster	2	57.37	80
M3 x 22 Cap Screw	Mcmaster	1	6.59	-
M3 Nut	Mcmaster	1	13.98	-
M3 Washer	Mcmaster	1	1.76	-
M3 Lock Washer	Mcmaster	1	1.62	-
Total	**151.17**			

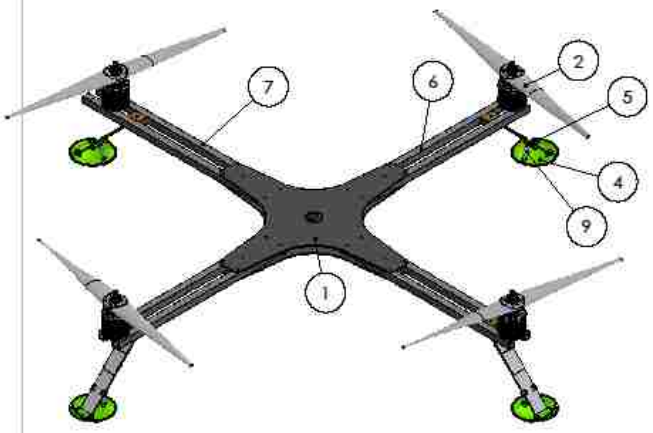
Appendix B: Manufacturing Drawings




52.70
127.70
357.50
370.20
NOTE:
1. UNITS ARE IN MM
PROJECT: HOPPER SPACECRAFT SIMULATOR
NAME
DATE
DRAWN
BJW
5/1/2014
VENDOR
HOBBYKING
MATERIAL
AXIAL CARBON FIBER
LEHIGH UNIVERSITY
TITLE
ARM-02
REV
A
WEIGHT: 24g
SHEET 1 OF 1


15
59.65
135°
1.27
119.30
R1
SCALE 1.3 : 1
PROJECT: HOPPER SPACECRAFT SIMULATOR
NAME
DATE
DRAWN
BJW
5/1/2014
VENDOR
MCMASTER
MATERIAL
SPRING STEEL
LEHIGH UNIVERSITY
TITLE
LEAF SPRING
REV
A
WEIGHT: 43g
SHEET 1 OF 1
NOTE:
1. UNITS ARE IN MM
5
4
3
2
1

ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	PLATE	CARBON FIBER PLATFORM (1/16IN THICKNESS)	2
2	ACTUATOR	T-MOTOR MT281-11 WITH 12X4.5IN PROPELLER	4
3	LANDING GEAR	STAINLESS STEEL LEAF SPRING LANDING GEAR SEE DWG	4
4	FEET	3D PRINTED FROM STL FILE	4
5	SPRING CLIP	3D PRINTED FROM STL FILE	4
6	SHORT ARM	SEE DWG (ARM-02) FOR DETAILS	4
7	LONG ARM	SEE DWG (ARM-01) FOR DETAILS	2
8	LANDING GEAR MOUNT	CUT TO FIT FROM BALSA WOOD	4
9	STAINLESS STEEL SPRING	MCMASTER PART# 9657K269	4



ASSEMBLY NOTES:
1. 3MM HARDWARE NOT SHOWN
2. LANDING GEAR MOUNT SHOULD BE EPOXIED IN PLACE

PROJECT: HOPPER SPACECRAFT SIMULATOR

	NAME	DATE
DRAWN	BJW	5/1/2014

LEHIGH UNIVERSITY

VENDOR: N/A

MATERIAL: N/A

TITLE: SIM. ASY

REV: B

WEIGHT:

SHEET 1 OF 1

5 4 3 2 1

Appendix C: Python Code

C.1: Fuzzification Module

```
# Brian Wisniewski
# 5/14/2014
# Fuzzification Module V-2
class Fuzzify:
    def __init__(self):
        pass
    def Fuzzy(self,e,ed,end_e,end_ed):
        def eTrapN(self,x):
            if x <= -.15:
                return 1
            elif x > -.15 and x < 0:
                return x/float(-.15)
            else:
                return 0

        def eTrapP(self,x):
            if x >= .15:
                return 1
            elif x > 0 and x < .15:
                return x/float(.15)
            else:
                return 0

        def eTri(self,x,end_e):
            #Assumes a symetric Trianguler memebership function
            end_e=.15/1
            if x >= -end_e and x < 0:
                return 1+x/float(end_e)
            elif x >= 0 and x <= end_e:
                return 1-x/float(end_e)
            else:
                return 0
        eFuzzy=[eTrapN(self,e),eTri(self,e,end_e),eTrapP(self,e)]


        def edTrapN(self,x):
            if x <= -1.5:
                return 1
            elif x > -1.5 and x < 0:
                return x/float(-1.5)
            else:
                return 0

        def edTrapP(self,x):
            if x >= 1.5:
                return 1
            elif x > 0 and x < 1.5:
                return x/float(1.5)
            else:
                return 0

        def edTri(self,x,end_ed):
            #Assumes a symetric Trianguler memebership function
            end_ed=1.5/1
```

```
            if x >= -end_ed and x < 0:
                return 1+x/float(end_ed)
            elif x >= 0 and x <= end_ed:
                return 1-x/float(end_ed)
            else:
                return 0
        edFuzzy=[edTrapN(self,ed),edTri(self,ed,end_ed),edTrapP(self,ed)]

        return eFuzzy, edFuzzy
```

C.2: Inference Module

```
1.  # Brian Wisniewski
2.  # 5/16/2014
3.  # Inference Engine Module V-1
4.  from numpy import zeros as ze
5.  from math import ceil
6.  class Inference:
7.      def __init__(self):
8.          pass
9.
10.     def Mnd(self,error,error_d): #Manmadi Inference engine
11.         RULES = [[0,0,1],[0,1,2],[1,2,2]]
12.         out=ze((4,2))
13.         ind=0
14.         Zer=[0]*4
15.         Pos=[0]*4
16.         Neg=[0]*4
17.
18.
19.         for ii in range(0,len(error_d)):
20.             if error_d[ii] > 0:
21.                 for jj in range(0,len(error)):
22.                     if error[jj] > 0:
23.                         out[ind,0] = min(error_d[ii],error[jj])
24.                         out[ind,1] = RULES[ii][jj]
25.                         ind=ind+1
26.
27.
28.
29.         for ii in range(0,4):
30.             if out[ii,1]==0:
31.                 Neg[ii]=out[ii,0]
32.             if out[ii,1]==1:
33.                 Zer[ii]=out[ii,0]
34.             if out[ii,1]==2:
35.                 Pos[ii]=out[ii,0]
36.         OUT=[ceil(max(Neg)*10000)/10000,ceil(max(Zer)*10000)/10000,ceil(max(Pos)
    *10000)/10000]
37.         return OUT
```

C.3: Defuzzification Module

```
1.  # Brian Wisniewski
2.
3.  # Defuzzification Module V-1
4.  class Defuzzify:
5.      def __init__(self):
6.          pass
7.
8.      def Defuzz(self,out):
9.          N=out[0]
10.         Z=out[1]
11.         P=out[2]
12.         if N>0:
13.             a=-(-1-N*(-.85))
14.             b=1
15.             An=.5*N*(a+b)
16.             xn=-1+(a**2+b**2+a*b)/(3*(a+b))
17.         else:
18.             An=xn=0
19.         if Z>0:
20.             a=2*((Z-1)*-.1)
21.             b=.2
22.             Az=.5*Z*(a+b)
23.             xz=0
24.         else:
25.             xz=Az=0
26.         if P>0:
27.             a=(1-P*(.85))
28.             b=1
29.             Ap=.5*P*(a+b)
30.             xp=1-(a**2+b**2+a*b)/(3*(a+b))
31.         else:
32.             xp=Ap=0
33.
34.
35.         X=(xn*An+xp*Ap)/(An+Az+Ap)
36.
37.         return X
```

C.3: Fuzzy Control Processing Module

```
from FUZZIFY_V2 import Fuzzify
from INFERENCE import Inference as inf
from DEFUZZIFY import Defuzzify
from math import exp

class Fuzzycontroller():
    def __init__(self,desired):
        #Normalize Desired Angle
        self.desiredRoll   = .016*desired[0][0]
        self.desiredPitch  = .016*desired[0][1]
        self.desiredYaw    = .00555*desired[0][2]
        self.desiredRolld  = .005*desired[1][0]
        self.desiredPitchd = .005*desired[1][1]
        self.desiredYawd   = .005*desired[1][2]
        self.defuzz=Defuzzify()
        self.fuzzy=Fuzzify()
        self.inf=inf()


    def control(self,data,throttle):
        per_control_pitch=.01
        per_control_roll=.01
        per_control_yaw=0
        offback=8
        offleft=8
        Blimit=1100


        #Normalize IMU Data Angle
        roll  =.016*data.roll
        pitch =.016*data.pitch
        yaw   =.0055555556*data.yaw

        rolld  =.005*data.roll_d
        pitchd =.005*data.pitch_d
        yawd   =.005*data.yaw_d
        if data.roll_d > 600 or data.roll_d < -600:
            if data.roll_d > 600:
                rolld=3
            else:
                rolld=-3
        if data.pitch_d > 600 or data.pitch_d < -600:
            if data.pitch_d > 600:
                pitchd=3
            else:
                pitchd=-3
        if data.yaw_d > 600 or data.yaw_d < -600:
            if data.yaw_d > 600:
                yawd=3
            else:
                yawd=-3


        roll_e  = self.desiredRoll-roll
        pitch_e = self.desiredPitch-pitch
        yaw_e   = self.desiredYaw-yaw
```

```
59.
60.         roll_ed  = self.desiredRolld-rolld
61.         pitch_ed = self.desiredPitchd-pitchd
62.         yaw_ed   = self.desiredYawd-yawd
63.
64.
65.         eFuzzyRoll ,edFuzzyRoll  =self.fuzzy.Fuzzy(roll_e,roll_ed,.15,1.5)
66.         eFuzzyPitch,edFuzzyPitch =self.fuzzy.Fuzzy(pitch_e,pitch_ed,.15,1.5)
67.         eFuzzyYaw  ,edFuzzyYaw   =self.fuzzy.Fuzzy(yaw_e,yaw_ed,.15,1.5)
68.         OUT=[0]*3
69.         OUT[0]=self.defuzz.Defuzz(self.inf.Mnd(eFuzzyRoll,edFuzzyRoll))
70.         OUT[1]=self.defuzz.Defuzz(self.inf.Mnd(eFuzzyPitch,edFuzzyPitch))
71.         OUT[2]=self.defuzz.Defuzz(self.inf.Mnd(eFuzzyYaw,edFuzzyYaw))
72.
73.         error=[roll_e,pitch_e,yaw_e]
74.         errorRate=[roll_ed,pitch_ed,yaw_ed]
75.         # This Defines the PWM OUTPUT that will be sent to the motors
76.         yaw=int(round(600*OUT[2]*per_control_yaw))
77.
78.         #Associated PWM Signal Sent to Motors
79.         front=+int(amp*(round(per_control_pitch*600*OUT[1])))+yaw+throttle+Blimi
    t-offback
80.         back=-int(amp*(round(per_control_pitch*600*OUT[1])))-
    yaw+throttle+Blimit+offback
81.         left=+int(amp*(round(per_control_roll*600*OUT[0])))-
    yaw+throttle+Blimit+offleft
82.         right=-
    int(amp*(round(per_control_roll*600*OUT[0])))+yaw+throttle+Blimit-offleft
83.
84.         PWMmot=[left,right,front,back]
85.
86.         return error, errorRate, OUT,PWMmot
```

C.4: Main Loop

```
from fuzzyCONTROLLER import Fuzzycontroller
from DATA import Data
import time
from UM6 import Imu
from MOTOR import Motor
import Adafruit_BBIO.UART as UART
from DATA import Data
from BLACK_BOX import Black_Box
# Initialize Black Box
bb = Black_Box()
data = Data()
bb.resetTime()

desired=[[0,-.3,0],[0,0,0]]
throttlemin=1000 #MAX throttle PWM signal 420
throttlemax=1300
data=Data()
controller=Fuzzycontroller(desired)

UART.setup("UART1")
print 'A'
imu=Imu('/dev/ttyO1')
print 'B'
imu.calibrate()
UART.setup("UART2")
motor = Motor('/dev/ttyO2')
motor.arm()
TEND=25
start=time.time()
flight=1
mode='B'
throttle=throttlemax
start=time.time()
Left  =1
Right =5
Front =3
Back =2
dtold=0
e2=0;
t2=0
while flight ==1:
    t1=t2
    e1=e2
    imu.stateUpdate(data)
    t2=time.time()
    bb.logData(data)
    dt=t2-t1
    error,errorRate,OUT,PWMmot=controller.control(data,throttle,e1,dt)
    e2=error[1]

    print error[1], errorRate[1]
    #print data.pitch_d
    #dtav=(dt+dtold)/2
    #dtold=dtav
    # Write PWM signal to proper channel
    # motor.writeChannel(Left, PWMmot[0])
    # motor.writeChannel(Right,PWMmot[1])
    motor.writeChannel(Front,PWMmot[2])
```

```
59.     motor.writeChannel(Back, PWMmot[3])
60.
61.
62.
63.     if time.time()-start>TEND:
64.         flight=0
65.
66. #print dtav
67. motor.disarm()
68.
69. # Close Device
70. motor.close()
71.
72. # Close Black Box
73. bb.close()
74.
75. # Plot Graph
76. #bb.plotGraph()
```

Vita

Growing up in a small town in Berks County, he was an active member of his community through his involvement with the Boy Scouts of America, where he received his Eagle Scout Award in the summer of 2008. As a student at Temple University in Philadelphia, he was active in many school organizations and was inducted into Pi Tau Sigma International Mechanical Engineering Honor Society. He was also instrumental in the development and remolding of the Introduction to Engineering course, where he provided three semesters of hands-on in-the-classroom teaching experience and was award the Diamond Peer Teaching Award. In 2013, Brian graduated from Temple University, where he received a B.S. in Mechanical Engineering and was awarded the Theodore P. Vassallo Award for outstanding and signification contributions to student activities in the College of Engineering.

During his undergraduate studies, Brian was involved in many engineering projects, including an interdisciplinary project to develop a lunar mining excavation system for NASA's Lunabotics Design competition, along with the development of the design project for Introduction to Engineering and a water powered vehicle for an ASME student design competition. With Brian's passion for engineering design, leadership, and service, Brian was a Student Government Representative for Temple Universities ASME student chapter. During his internship with Kingsbury Inc., Brian was a member of the internal support and development team, where he worked with machinists and engineers to ensure all product lines were up-to-date and accurate.

While at Lehigh University, Brian worked for Professor Terry Hart as a teaching assistant for systems engineering and control systems while he worked towards his thesis. Brian also received the NASA student research scholarship through the Pennsylvania State Grant Consortium for his graduate work.