University of South Florida
Digital Commons @ University of South Florida

USF Tampa Graduate Theses and Dissertations | USF Graduate Theses and Dissertations

June 2019

CubeSat Constellation Design for Intersatellite Linking

Michael T. White
University of South Florida, michaelwhite325@yahoo.com

Follow this and additional works at: https://digitalcommons.usf.edu/etd

Part of the Electrical and Computer Engineering Commons

Scholar Commons Citation

White, Michael T., "CubeSat Constellation Design for Intersatellite Linking" (2019). *USF Tampa Graduate Theses and Dissertations.*
https://digitalcommons.usf.edu/etd/7987

This Thesis is brought to you for free and open access by the USF Graduate Theses and Dissertations at Digital Commons @ University of South Florida. It has been accepted for inclusion in USF Tampa Graduate Theses and Dissertations by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact digitalcommons@usf.edu.

CubeSat Constellation Design for Intersatellite Linking

by

Michael T. White

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Robert Bishop, Ph.D.
Chung Seop Jeong, Ph.D.
Wilfrido Moreno, Ph.D.

Date of Approval:
June 14, 2019

Keywords: fuzzy control, MIMO control, Kalman filter, topology, MATLAB, Simulink

Copyright © 2019, Michael T. White

ACKNOWLEDGMENTS

I would like to acknowledge my loving friends and family for their support over the years. This opportunity to study and learn was made possible by their support through their words of encouragement and guidance.

I have had the distinct advantage in my undergrad years to be inspired by great engineers. My mentor Dean Bishop inspired me to pursue higher education. His teaching methods and knowledge fascinated me with the control theory discipline. I would like to thank Dr. Leffew for inspiring me to want to serve as an engineer for the U.S. Armed Forces; his stories and character have shown me how important well engineered systems are for the safety and betterment of our great nation. I would like to thank Dr. Fehr for inspiring me to pursue a P.E. license. Moreover, all the professors in the electrical engineering department, along with my peers, have proved to me that I made a good call joining the ranks of electrical engineers to help solve the world's problem.

TABLE OF CONTENTS

LIST OF TABLES iii

LIST OF FIGURES iv

ABSTRACT v

CHAPTER 1: INTRODUCTION 1
- 1.1 Overview 1
- 1.2 Objectives 2
- 1.3 Organization 4

CHAPTER 2: BACKGROUND AND PROBLEM SET-UP 5
- 2.1 Star Tracker 5
- 2.2 Extended Kalman Filter (EKF) 6
- 2.3 Satellite Constellation Formations and Attitude Control 7
- 2.4 MIMO Controller 10
  - 2.4.1 Fuzzy Control 11
  - 2.4.2 Sliding Mode Control (SMC) 12

CHAPTER 3: MATHEMATICAL MODELS 14
- 3.1 Star Tracker Plant System (STS) 14
  - 3.1.1 STS Sensor Model 16
  - 3.1.2 Hardware 18
    - 3.1.2.1 Features 18
    - 3.1.2.2 Product Properties 18
    - 3.1.2.3 Performance 18
- 3.2 Extended Kalman Filter (EKF) 19
  - 3.2.1 Propagation Cycle 20
  - 3.2.2 Update Cycle 22
- 3.3 Satellite System Model Plant 23
  - 3.3.1 Overview 23
  - 3.3.2 Cube-Sat ACS Hardware 23
    - 3.3.2.1 Attitude Sensors 23
    - 3.3.2.2 Magnetorquer Rod Nctr-Moo2 25
    - 3.3.2.3 MSS Magnetometer 27
  - 3.3.3 Electronic Configuration 28
- 3.4 System Modeling 28
  - 3.4.1 Attitude Kinematics 28
  - 3.4.2 Magnetic Field and Actuator Models 29

3.4.2.1 Magnetic Dipole Moment 29
3.4.2.2 Magnetic Field 30
3.4.2.3 Control Torque 31

CHAPTER 4: SIMULATION 32
4.1 Overview 32
4.2 Actuators 33
4.2.1 Actuator Torque Limit 33
4.2.1.1 Torque Limit for Each Axis 33
4.3 Extended Kalman Filter (EKF) 35
4.4 Plant 36
4.5 Controller 38

CHAPTER 5: RESULTS AND CONCLUSION 44
5.1 Results 44
5.2 Future Work 51
5.3 Conclusion 52

REFERENCES 53

APPENDICES 57
Appendix A: EKF MATLAB Code 58
Appendix B: Fuzzy Control Concepts 61
Appendix C: Quaternion Concepts 63
Appendix D: Max Torque for Actuators Code 65

LIST OF TABLES

Table 3.1: Reference frames for star tracker system 16

Table 3.2: Properties of Cube-Sat ACS hardware 24

Table 3.3: Testing results of the Cube-Sat hardware 25

LIST OF FIGURES

Figure 1.1: CubeSat constellation communication and corrected topology 2

Figure 1.2: Novel controller..... 3

Figure 4.1: Torque limit in Simulink 34

Figure 4.2: EKF in Simulink 35

Figure 4.3: Simulink attitude kinematics..... 37

Figure 4.4: Simulink Omega model 38

Figure 4.5: Basic PID 40

Figure 4.6: SMC architecture 41

Figure 4.7: SMC with fuzzy logic 42

Figure 4.8: Normalized quaternion plot 43

Figure 5.1: MIMO Simulink diagram 45

Figure 5.2: Plant #1 quaternion plot 46

Figure 5.3: Plant #2 quaternion plot 47

Figure 5.4: Plant #3 quaternion plot 48

Figure 5.5: Angular velocity estimates..... 49

Figure 5.6: Attitude error..... 50

Figure 5.7: Velocity error 51

Figure A.1: MATLAB Fuzzy Toolbox interface 61

Figure A.2: Membership function plot..... 62

Figure A.3: Surface viewer 62

ABSTRACT

This thesis investigates the concept of controlling a CubeSat constellation in low-Earth orbit. Low-Earth orbits are considered because the torque used for satellite control is supplied with magnetorquers, and the closer the satellite is to Earth's magnetic field the more control gain can be supplied. Also, this is the expected orbit altitude of future CubeSat constellations to enable communications.

Controlling a CubeSat relies on attitude determination. This means being able to estimate its attitude relative to a given reference frame. To determine the attitude, we propose to use a star tracker and a Kalman filter. A star tracker scans the stars in the satellite's view, correlates the object to a database, to return an attitude measurement. The measurement is then processed using the Kalman filter. The attitude estimate is then used as the reference input for the controller.

Once the attitude of the satellites is determined, a controller can be implemented; assuming the system is controllable and observable. These parameters are verified by adding enough actuators and sensors, respectively.

The novelty of this thesis is constructing a controller that will take three satellites and their attitude estimates and arrange them broadside to a target. For simplicity, the arrangement will be a linear formation, and the target and satellite constellation will all be near-field communication. The goal is to place the satellite constellation in an attitude for an intersatellite link to be established. This is a proposed solution to better budget power and computational constraints associated with CubeSats. In addition to adjusting the topology of the system, a communication method must be considered for the data to be distributed across the system

requiring an antenna design to implement the communication method. Both issues are discussed in the thesis; however, the focus is the controller design for attitude control. The control approach is a multi-input multi-output (MIMO) sliding fuzzy controller. The focus of the analysis is attitude control for communication while maintaining the constellation in a linear formation. The results shown this controller to be a valid proof of concept.

CHAPTER 1:

INTRODUCTION

1.1 Overview

The focus of this thesis is control of a constellation of three CubeSats. Attitude measurements are generated from an onboard star tracker sensor. An Extended Kalman Filter (EKF) processes the measurements to determine an estimate of the CubeSat's attitude. All three satellite's attitude will be actively controlled to optimize communication. The communication is an intersatellite link (ISL) between the given constellation and another system; be it a constellation or a single (larger) standard satellite.

There are many benefits CubeSats can provide. One major drawback however, can be their limited power. With a limited power device, developing an ISL with large bandwidth capability can be challenging. With the future of CubeSats being able to connect to the internet and become part of the 5G technology family, this constraint must be addressed. By having the CubeSats assembled in a constellation and equipped with a cognitive radio network (CRN), the power can be budgeted optimally [1]. A CRN can be implemented with a Raspberry Pi and a Universal Software Radio Peripheral (USRP) [2]. The problem still arises for system communications outside the constellation network. This is where a digital controller whose mission is to reconfigure the constellation such that minimal losses and high bandwidth are achieved for the ISL. Figure 1.1 illustrates the system topology. The constellation is such that a linear and equally spaced phased array system is established. The controller is shown first

obtaining a desired attitude signal which represents facing the target. Aligning the system broadside to the controller will produce an optimal scan angle for broadband connection [6].

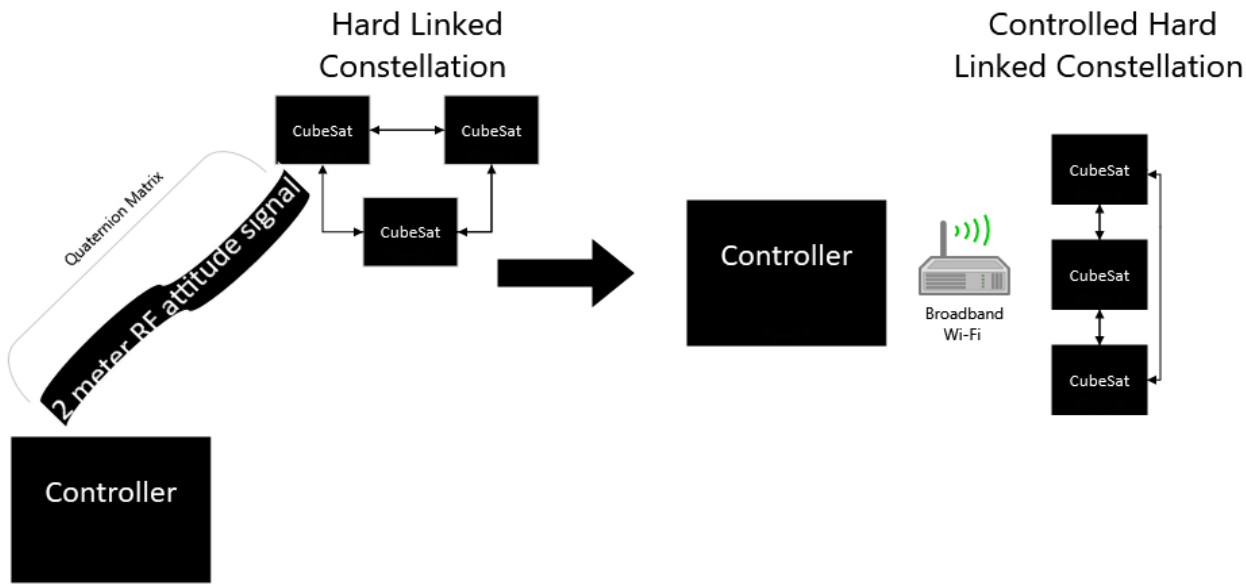

Figure 1.1: CubeSat constellation communication and corrected topology.

1.2 Objectives

For the ISL to be established, a phased array antenna system is proposed. It is proposed that the antenna aperture be computer controlled utilizing a Raspberry Pi, and each satellite considered as one element for the overall three element antenna system. A narrowband signal containing the satellites coordinates will initiate the controller to optimize the constellation for wideband communication. The EKF will provide the attitude estimates. The referencing input to the controller is the target constellation desired attitude. A fuzzy controller similar to [3] - [5] will provide the desired control signals for each satellite.

The theme of this thesis is the application of control theory and not RF communications, hence there are key assumptions. These assumptions will be stated more thoroughly in Chapter 4, but are summarized here:

- The main assumption for communication is that the satellites are hard-linked to avoid additional wireless noise and signal loss
- The main assumption for the antenna is that it receives and transmits corrupted only by addictive white gaussian noise (AWGN)
- The main assumption for the controller is that it is focused strictly on network topology, in this case a linear constellation, implying that pointing at the target will remain out of scope

It is also important to note that all solutions to this problem are presented in a heuristic manner; meaning a more optimized approach is feasible and the simulated work in Chapter 4 is solely for proof of concept. Figure 1.2 illustrates the block diagram of the novel controller. This is a proposed controller for attitude adjustments. The plant will contain dynamics of the constellation. The target position equates to having two axes equal to each other, and one axis with a small, equally spaced, variance. This can be observed in target position one. Target position two illustrates the definition of broadside. Basically, by placing the constellation at a desired position in one axis we can achieve broadside alignment.

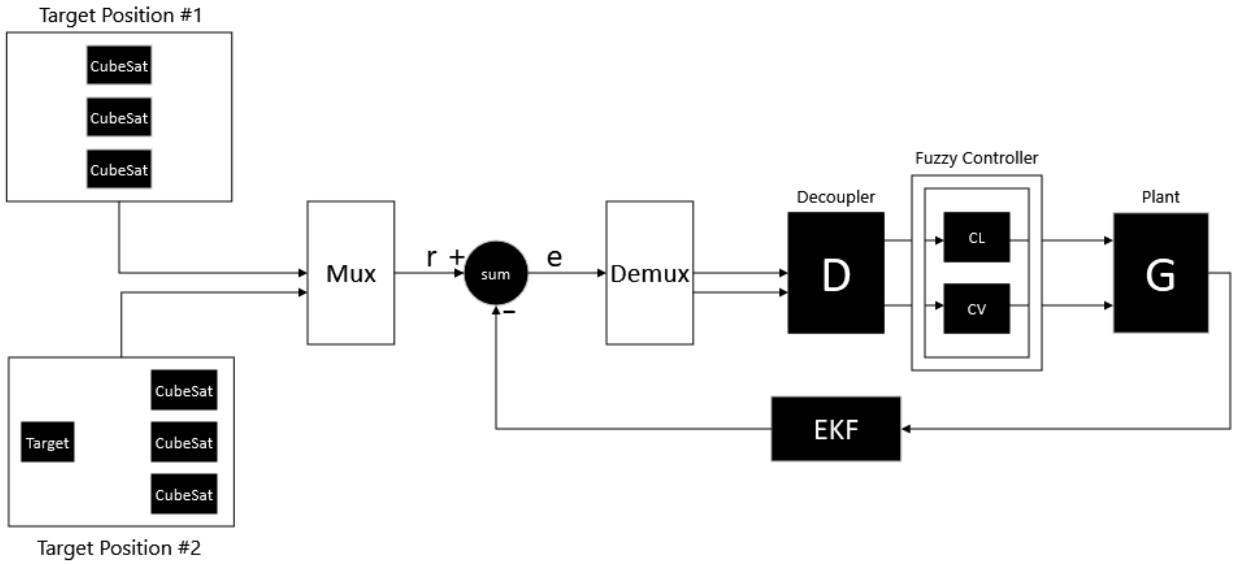

Figure 1.2: Novel controller.

1.3 Organization

In Chapter 2, the basic components and theory are discussed. The literature survey constructed in Chapter 2 serves as the background to any readers who are not familiar with the technology and terminology. In Chapter 3, the theory and mathematical models are discussed to build the foundation of the simulation. Chapter 3 also contains information about mass-produced CubeSat components to provide validly as a proof of concept. In Chapter 4, we simulate and produce the main results. In Chapter 4, the methodology for constructing and testing the controller is also discussed. The results and conclusions are presented in Chapter 5.

CHAPTER 2:

BACKGROUND AND PROBLEM SET-UP

2.1 Star Tracker

A challenge in astrophotography is to record as much starlight as possible before the stars make trails in the image. Star-trails pose an issue with processing data to map the night sky. The motion of the Earth imposes a limit on sharpness and detail an image can provide. The camera needs to use a brisk, wide point central focus and high ISO characteristics to image details. A mechanical assembly exists to address this issue and reveal higher resolution images of the stars. The principal tool astrophotographers use is an equatorial mount, also known as a star tracker. These devices are created to empower telescopes to autonomously track stars in the night sky without having the photographer physically move the telescope system to keep the objects in frame. These motorized mounts work by knowing the speed at which Earth is rotating and provide a control signal to counteract the Earth's rotation. Normally these trackers are large so they can function on a myriad of telescopes. There are companies like Sky Watcher, Vixen, and iOptron who have made more compact star trackers that can actively track the night sky with camera equipment alone, i.e. no heavy telescope equipment is needed [7]. With modern technology providing a smaller and more compact design, this tool becomes a valid candidate to be used as a sensor for a CubeSat.

The function of a star tracker on a CubeSat is to provide a measurement of attitude relative to a given reference frame. This is achieved by obtaining an image of the sky and mapping star constellations. These constellations are then correlated to known positions in a

database and an attitude quaternion is generated. Appendix C provides information quaternions for attitude representations. The accuracy of a star tracker can be increased with redundancy, i.e. using two cameras for imaging and incorporating a complimentary filter to ensure the attitude measurement is near the true attitude [8].

2.2 Extended Kalman Filter (EKF)

The Kalman Filter (KF) has been proven as a robust system for state estimation and providing stability in stochastic systems [9]. The KF can also be interpreted as a linear quadratic estimator (LQE). It is an algorithm that makes use of measurements and state estimates that are observed over a period of time. The EKF is a nonlinear implementation of the linear Kalman filter. The KF (and EKF) are sensor fusion algorithms that provides an optimal state estimate utilizing external measurements and an internal model of the external environment. The EKF optimally balances the information from the measurements and the state estimates at the time of the measurement to provide a more accurate state estimate. In this thesis, the external measurements are provided by the star tracker and the attitude estimate is provided by the EKF.

The response time of the EKF to reach an accurate state estimate is directly correlated with the quality of sensors and the accuracy of the mathematical models. This means if the sensor is behaving more ideally and introducing low noise into the system, then the amount of measurements needed is reduced. Considering computational costs, it is better to have a high-quality sensor. In the case of CubeSats, using a higher quality sensor would lower the time to reach an accurate attitude estimate.

2.3 Satellite Constellation Formations and Attitude Control

The coordination amongst precisely-controlled spacecraft, belonging to a more-than-one platform (formation or constellation), is a primary requirement in important missions, especially those using radars or optical interferometers. It is also an open challenge to develop integrated control systems for federated spacecraft systems. Different techniques are applicable to create and control coordinated mindset constellations [13]. The traditional control strategy is a chief-follower structure in which all spacecraft rely on the behavior of a control signal sent from a chief satellite. Alternatively, the multi-behavioral method is characterized by each satellite independently transmitting their location to a shared network where they can jointly decide how to arrange the constellation. This is known as forming a swarm and has been demonstrated with autonomous drone technology [14].

A software defined program in [15] proposes a unified remedy of those principles by means of a few fundamental definitions of the consensus dynamics and cooperative control. The convergence to the centered configuration is addressed analytically through the usage of Lyapunov stability requirements and numerically via numerical simulations. The mindset requirements and constraints are highlighted and a solution to manage a set of rules regarding continuous actuators on each platform is developed. A comparative analysis of the best suited control techniques is demonstrated using the Linear Quadratic Regulation (LQR) and the State Dependent Riccati Equation (SDRE). Satellite constellations and formations provide several advantages when compared to a single spacecraft. Some of those benefits are [16]:

- Mission flexibility
- Increased mission reliability by having redundancy in the system
- Reduction in overall costs

- Improved ability to mitigate disasters by having assignment exchanges amongst members of the constellation

The number of applications related to using more than one spacecraft, both in recent times or under development programs, confirms the interest of further development for this kind of control concept. A challenge that arises in executing adaptive formation swarming is developing rule-based strategies that allow the spacecraft to autonomously accomplish the obligations of steering, navigation, and being adaptive in an environment with many uncertainties, all while maintaining low power consumption [17].

The problems concerning the control of relative positions and attitude among spacecraft were widely investigated in the past. The difficulty of mindset coordination (swarm formation) among spacecraft is more current because the actual implementation requires controlling its networking with enormous amounts of data. As mentioned in Chapter 1, a cognitive radio network (CRN) can handle the communication requirement, but an adaptive controller is also desired to maintain robustness. Attitude coordination is pinnacle for a constellation to maintain a desired formation or hold a required position. In [18], some novel techniques for the self-sustaining coordination of spacecraft are presented.

One of those conventional approaches is the so-called leader follower structure [19], in which every spacecraft must track the mindset of a designated leader. Coordination challenges can be reduced by tracking known problems and logging them as *a priori* of the system. This known *priori* can then be used to calculate the probability of an error to occur, thereby assisting in reducing uncertainties. Within such an approach, however, no information from the followers are utilized by the chief satellite, and the chief becomes a system that cannot fail. Alternatively, a non-unique solution is given by the control law of the situational-based selection of the chief

satellite in which the control adjusts based on a utility cost between each spacecraft, and cooperatively they set control gains by determining the minimal control effort needed when taking into account all systems. In other words, having all satellites positions known, a controller will be optimized with information from all satellites and not solely on a single, fixed chief satellite. This allows for better versatility of the formation that can adapt itself to precise responsibilities providing more tuning possibilities. However, this non-unique performance does take more computational effort and traditionally would need to be computed onboard every spacecraft. This traditional technique would have each system considered as an independent agent, and in terms of a communique hyperlink, would be continuously switching information from/to every satellite in the formation. A novel approach to address this issue is to design a control scheme similar to [20], however instead of mitigating actuators it would be distributing the overall control signals first and then optimizing the control gains to the individual actuators based upon the positions of the other satellite-systems. This method is primarily based on the definition of a digital form, which is now structured on modern-day servers. More research is needed to investigate the efficiency; in addition to, the benefits and the drawbacks of having decentralized as opposed to centralized selection agents. Specifically, these studies would need to analyze the opportunity of having a single global (common-sense) unit for computing every control signal and then assign a satellite specific control signal to every platform of the formation; not to mention taking into account redundancy, via a distribution of the choice authority among some or all spacecraft in order to benefit the non-unique solution and favor a cognative configuration. Basically, how to operate and optimize a MIMO controller within a cognitive network of CubeSats.

2.4 MIMO Controller

The following section assumes the reader has knowledge of classical control theory knowledge. Consider the following MIMO model as a concatenation of SISO models. Using transfer functions ‘tf’ and Laplace notation ‘s’. Consider H(s) as a single-input, two-output system [21]

$$H(s) = \begin{bmatrix} \frac{s-1}{s+1} \\ \frac{s+2}{s^2+4s+5} \end{bmatrix}.$$

You can depict H(s) by connection of its SISO transfer functions. For instance,

```
h11 = tf ([1 - 1], [1 1]);

h21 = tf ([1 2], [1 4 5]);
```

or, similarly,

```
s = tf ('s')

h11 = (s-1)/(s+1);

h21 = (s+2)/(s^2+4*s+5);
```

this can be connected to

```
H = [h11; h21].
```

This dialect structure has the benefit of being readable to those familiar with the notation of transfer functions. There is another format to represent MIMO systems and it represents transfer functions with cell arrays arguments. To portray MIMO systems, it requires two cell groups; N and D are commonly used to depict the numerator and denominator polynomials. Considering the same system H(s) mentioned above; the two cell shows N and D will contain the depictions of the numerator/denominator polynomial entries: N(s) = [(s−1)/(s+2)], D(s)

=[(s+1)/(s^2+4s+5)]. You can write this MIMO transfer function H(s) by declaring the following,

N = {[1 - 1]; [1 2]};

D = {[1 1]; [1 4 5]};

H = tf (N, D)

this will yield two transfer functions,

$$\frac{s-1}{s+2}$$

$$\frac{s+1}{s^2+4s+5}$$

For a block diagram representation of a MIMO system please see [22]. A block diagram is helpful for visualizing what a matrix of transfer functions looks like in Simulink.

2.4.1 Fuzzy Control

There are many challenges when attempting to control a MIMO system. One of the main challenges in the case study depicted in this thesis is how to assign control gains for the first few measurements. This initialization is what makes simple feedback controllers go unstable or have undesirable characteristics. A fuzzy controller can address this challenge by analyzing inputs and based on boundary conditions (rules) being satisfied within the programmable controller to then program each case into a desired range of outputs. In other words, the fuzzy logic will saturate the controller output range based on inputs. Consider the following example where a controller output ‘y’ has programmed thresholds that are case specific for some input ‘x’:

1.) if x >= 100, y = Max value

2.) if x <= 100, y = Min value

3.) if 100 < x < 100, y = u

where

Max value – is the least upper stability boundary (supremum) of the controller actuator

Min value – is the greatest lower stability boundary (infimum) of the controller actuator

u – is a control output of the feedback controller.

This can be realized with a python script implementing if-then statements designated as rules for a controller.

What makes a fuzzy decision rule unique from other controllers is that it can be programmed to assign weights other than just 1 or 0 to if-then statements. “In fuzzy logic, the truth of any statement becomes a matter of degree” [23]. A fuzzy decision has overlapping rule-boundaries and the weight of the output-decision, for a specific rule or case, decreases when the input is close to the max or min boundary of that particular rule or case. See Appendix B for more information on fuzzy controller architectures being created in MATLAB.

2.4.2 Sliding Mode Control (SMC)

Sliding mode control is a technique used to remove non-linearities in a control system [24]. The main advantages for implementing a sliding mode controller are:

- Order reduction
- Decoupling design procedures
- Increased robustness
- Disturbance rejection
- Simple implementation by means of conventional power converters

SMC’s are also good for suppressing chattering (digital noise). Chattering is introduced into the system as a downside of using a fuzzy controller.

When implementing a SMC into the overall control design, the control law becomes dynamic and changes due to pre-defined rules. The rules are defined as switching functions and

depending on the plant, a switching gain will be resolved. The switching gains are designed such that for a given input, the system will reach a sliding surface, that is designed to be linear, and maintain close vicinity to said surface for future inputs [25]. This process can be summarized into two phases where the first phase is reaching the sliding surface, and the second phase is sliding either up or down until the system appears to have linear characteristics. The downside is having observed chatter when sliding along the surface, however these oscillations should decrease once steady state is achieved. Chattering-free SMC is possible with additional tuning, and a disturbance observer can improve performance requirements, such as faster settling times. It is worth noting that uncertainties and disturbances are assumed to be bounded for SMC to maintain stability, and the overall system must be reachable. In order to be reachable, the system must have its noncontrollable components of the system's state observed as stable. A reachable system will satisfy the criteria to be controllable. Recalling, controllability issues can be solved by adding actuators to control components that are observed in an undesirable state.

CHAPTER 3:

MATHEMATICAL MODELS

3.1 Star Tracker Plant System (STS)

One of the most fundamental subsystems on any spacecraft is its attitude determination system (ADS), including the sensor (star tracker), the Kalman filter, and the CubeSat itself. First, we present the attitudes estimation techniques, relying on star tracker measurements to provide attitude measurements in the form of quaternions, using the quaternion measurement to estimate the spacecraft attitude in the EKF, and then control this attitude to the desired attitude. Other methods may include the use of gyros, but in [26] we see the cons of using a gyro, namely the drifting factor with respect to time, i.e. as the passage of time increases so does the measurement error. To estimate the spacecraft altitude without high computational cost, we utilize only the star tracker and the EKF. First, it's important to understand the dynamic and kinematic equations of satellite motion.

To determine the attitude dynamics of a spacecraft we use Euler's equation of motion

$$\mathbf{I}\dot{\boldsymbol{\omega}}_B^I = \mathbf{n}_{GG} + \mathrm{N}_D + \mathbf{n}_{MT} - \boldsymbol{\omega}_B^I \times (\mathbf{I}\boldsymbol{\omega}_B^I) \tag{1}$$

where

$\boldsymbol{\omega}_B^I$ – Is the angular velocity vector of the body relative to inertial space

$\mathbf{I}$ – Is the inertia matrix of the spacecraft

$\mathbf{n}_{GG}$ – Gravity gradient torque vector in the body frame

$\mathbf{n}_{MT}$ – Applied magnetorquer control torque in the body frame

N_D – Disturbance torque in the body frame.

The kinematics equation of spacecraft in terms of quaternions is given by [42]

$$\dot{\overline{\boldsymbol{q}}} = \frac{1}{2}\overline{\boldsymbol{\omega}} \otimes \overline{\mathbf{q}} \tag{2}$$

with

$$\overline{\boldsymbol{\omega}} = \begin{bmatrix} \boldsymbol{\omega} \\ 0 \end{bmatrix},$$

here the bold with an overbar signifies a quaternion, i.e. $\overline{\boldsymbol{\omega}}$ is seen as the pure quaternion with vector component $\boldsymbol{\omega}$ and a scalar component 0. For those not familiar with angular velocity, it is a triad vector that contains angular rates for each axis. The $\{\otimes\}$ operator describes quaternion multiplication, which can be expanded to

$$\dot{\overline{\boldsymbol{q}}} = 1/2 \begin{bmatrix} \mathrm{q}\boldsymbol{\omega} - \boldsymbol{\omega} \times \mathbf{q} \\ -\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{q} \end{bmatrix}. \tag{3}$$

On the right side of the equation, the superscript T represents the transpose operation.

The Euler theorem states that any attitude can be represented by a single angle of rotation and a single axis of rotation. This information can be translated into a rotation matrix described as [43]

$$\mathbf{T} = \mathbf{I}_{3 \times 3} - \sin\theta\, [\mathbf{e}\times] + (1 - \cos\theta)\, [\mathbf{e}\times]^2, \tag{4}$$

where

T – Is the rotational matrix

I – Is the identity matrix

θ – Is the angle of rotation

e – Is the axis of rotation.

The $[\cdot\times]$ represents the cross-product matrix, and when expanded with **e** as an operand, we have

$$[\mathbf{e}\times] = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix}.$$

The rotational matrix shown in Eq. (4) can be mapped to the quaternion coordinate system using

$$\mathbf{T} = (q^2 + \mathbf{q}^T\mathbf{q})\, \mathbf{I}_{3\times3} - 2q\,[\mathbf{q}\times] + 2\,[\mathbf{q}\times]^2 \tag{5}$$

where

$$\overline{\mathbf{q}} = \begin{bmatrix}\mathbf{q}\\ q\end{bmatrix} = \begin{bmatrix}\sin\frac{\theta}{2}\mathbf{e}\\ \cos\frac{\theta}{2}\end{bmatrix}; \tag{6}$$

thus, we have a quaternion of rotation containing the Euler angle and axis, which in turn will be used to describe an object's attitude. Realizing, if you take the derivative with respect to time of Eq. (6) it will yield Eq. (3).

3.1.1 STS Sensor Model

The generation of quaternions is done so by mapping a rotational matrix from different references frames into the quaternion coordinate system. In Table 3.1 the reference frame representations are shown.

Table 3.1: Reference frames for star tracker system.

Reference Symbol	Reference Frame
i	Inertial Frame
b	Spacecraft Body
st	Star Tracker Camera
sm	Star Map
b, η	Random Bias, Noise

The mathematical model, including biasing and noise, for the star tracker system is

$$\overline{\boldsymbol{q}}_{sm}^{st} = \overline{\mathbf{q}}_{b,\eta} \otimes \overline{\boldsymbol{q}}_{b}^{st} \otimes \overline{\boldsymbol{q}}_{i}^{b} \otimes \overline{\boldsymbol{q}}_{sm}^{i}\,. \tag{7}$$

The notation of the subscripts and superscripts indicates the translation from one reference frame to another, e.g. $\overline{\boldsymbol{q}}_{sm}^{st}$ is a quaternion represented from the star tracker map reference frame to the

star tracker reference frame. It is desired to obtain $\overline{\boldsymbol{q}}_i^b$ in order to relate the attitude quaternion to a shared reference frame amongst other satellites. Quaternions $\overline{\boldsymbol{q}}_b^{st}$ and $\overline{\boldsymbol{q}}_{sm}^i$ are constant and known; where $\overline{\boldsymbol{q}}_{sm}^i$ is the transition from the star map to the inertial frame, and $\overline{\boldsymbol{q}}_b^{st}$ is generated if the placement of the star tracker, when fixed onto the satellite, is skewed in relation to the reference attitude of the satellite body. The noise and biases are introduced into the system when generating measurements. A bias could be possible if the sensor was not calibrated properly. It is important to realize that we cannot simply add the noise and biases quaternion $\overline{\mathbf{q}}_{b,\eta}$ to the measurement. This is because the quaternion that is generated from the device must meet the unity norm constraint

$$\|\overline{\mathbf{q}}\| = \sqrt{\mathrm{q}^2 + \mathbf{q}^{\mathrm{T}}\mathbf{q}} = 1 \ ,$$

and this must be satisfied so the coordinate system translates to an orthogonal Euler coordinate system, by definition. We use quaternion multiplication described in Eq. (7) to account for noise and biases while maintaining the unity norm constraint.

There will clearly be some deviation from the true attitude quaternion and the measured, due to the addictive white noise and random biases. To correct for these errors, we will use the EKF to estimate the noise and biases. In addition to estimating the noise and bias quaternion, the EKF will estimate the quaternion shown in Eq. (7), i.e. $\widehat{\overline{\boldsymbol{q}}}_{sm}^{st}$**.** Using the EKF estimates, we will minimize the errors contained in the star tracker measurements, namely by optimizing the filter to minimize the difference of the estimate and the true attitude with

$$\overline{\mathbf{q}}_{residual} = \overline{\boldsymbol{q}}_{sm}^{st} \otimes \left[\widehat{\overline{\boldsymbol{q}}}_{sm}^{st}\right]^{-1} , \tag{8}$$

where $\overline{\mathbf{q}}_{residual}$ represents the difference from the true attitude quaternion and the estimated attitude quaternion. Ideally the residual will be minimized, thus revealing our estimate is

accurate, and confirming our error modeling was true to what the star tracker measured. The estimation process will be shown in detail later on in the chapter.

3.1.2 Hardware

A typical hardware set-up for the star tracker is described here will follow [27].

3.1.2.1 Features

- Condensed system (45x50x95) millimeter.
- Highly exact: 2 arc sec (1σ) cross-boresight, 10 arc sec (1σ) round boresight.
- Update the rate up to 10 Hertz.
- Lost-in-space accessibility > 99 percent of the night sky.
- On-board standardization algorithm.
- Independent outlier star removal and detection.
- 40-degree baffle sun rejection angle.
- Optics and Electronics radiation tested.

3.1.2.2 Product Properties

- 5 Volt needed
- 250 grams
- 45x50x95 millimeter (with depression channel to ease harnessing)
- SPI
- Tested thoroughly: Vibration, electronics radiation, Optics radiation, etc.

3.1.2.3 Performance

- Absolute error of Knowledge (cross-boresight) 2 arc second (1σ)
- Absolute error of Knowledge (around boresight) 10 arc second (1σ)
- Theoretical power consumption is less than one Watt

3.2 Extended Kalman Filter (EKF)

In this analysis, the EKF is used as an estimation algorithm. The primary purpose of the filter is to compute a state estimate in the presence of sensor noise and random bias. The EKF is an unbiased estimator. There are two estimates that the EKF will generate: the state estimate and the state estimation error covariance. The state estimation error covariance is used to measure the uncertainty contained within the state estimate. As time passes, the uncertainties should be decreasing because, due to measurement updates, we know more information about the system. [42]

The filter functions recursively in two cycles: propagating and updating. During the propagation phase it is important to properly model the dynamics of the system. If the model is not accurate then the uncertainties will be higher, and this makes the propagation cycle estimates less useful. The update phase is where the sensor models will come into play. If the models are accurate to the true system, then the estimates generated will have less uncertainties. Both the propagation and update cycle are important to determining the gain of the filter, i.e. the Kalman gain value. The gain is optimized by assigning weights to the state estimates of both cycles. The filter will assign a higher weight to the state estimate produced with a smaller state estimation error covariance, e.g. if the last propagated state estimate has a smaller covariance than the current updated measurement, the filter will assign a larger weight to the state estimate from the propagation phase and a lower weight to the state estimate from the update phase.

The nonlinear system is modeled in the EKF as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{M}(t)\mathbf{w}(\mathrm{t}) , \tag{9}$$

where

$\mathbf{x}(t) \in \mathbb{R}^n$ is the state of the system

$\mathbf{f}(\mathbf{x}(t), t) \in \mathbb{R}^n$ is the nonlinear system model

$\mathbf{w}(t) \in \mathbb{R}^p$ is the process noise

$\mathbf{M}(t) \in \mathbb{R}^{n\times p}$ is the process noise mapping matrix.

The process noise is assumed to be a zero-mean, white noise process with a constant-power spectral density.

The nonlinear measurement is modeled as

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \,, \tag{10}$$

where

$\mathbf{h}_k(\mathbf{x}_k) \in \mathbb{R}^m$ is the nonlinear measurement model evaluated at the state $\mathbf{x}_k = \mathbf{x}(t_k)$

$\mathbf{v}_k \in \mathbb{R}^m$ is the measurement noise.

The measurement noise is assumed to be a zero-mean white-noise sequence with a noise covariance given by $\mathbf{R}_k \in \mathbb{R}^{m\times m}$. We assume covariance is constructed with $\mathrm{E}\{\mathbf{v}_k\mathbf{v}_{k'}^{\mathrm{T}}\} = \mathbf{R}_k\delta_{kk'}$ where 'E{·}' is the expected value operation and '$\delta_{kk'}$' is the Kronecker delta function. The delta function will return true if k=k' and false otherwise. Noting the subscript 'k' means we have a discrete time measurement at time $t = (t_k)$. Another assumption is that the measurement noise is not correlated with the process noise; in other words, they are completely independent of each other in time.

3.2.1 Propagation Cycle

The first step in the estimation process is to define the state vector [28]

$$\mathbf{X} = [\boldsymbol{\omega}_B^I \;\; \mathbf{q}]^T = [\omega_x \;\; \omega_y \;\; \omega_z \;\; q_1 \;\; q_2 \;\; q_3 \;\; q_4]^T \,. \tag{11}$$

The state is propagated numerically via

$$\overline{\boldsymbol{q}}_{k+1} = \widehat{\boldsymbol{q}}_k + \int_{t_k}^{t_{k+1}} (\boldsymbol{\Omega}\mathbf{q})\mathrm{dt} \,. \tag{12}$$

Here $\bar{\mathbf{q}}$ and $\hat{\mathbf{q}}$ represent a quaternion and an estimated quaternion, respectively. $\mathbf{\Omega}$ is the cross-product matrix of $\boldsymbol{\omega}_B^I$. The limits of integration are determined by the definition of the propagation phase, i.e. the time in between sample updates. The propagation state estimation error covariance is given by [42]

$$\mathbf{P}(t_k) = \boldsymbol{\phi}(t_k, t_{k-1})\, \mathbf{P}(t_{k-1})\, \boldsymbol{\phi}^{\mathrm{T}}(t_k, t_{k-1}) + \mathbf{Q}(t_k)\ , \tag{13}$$

where

$\boldsymbol{\phi}$ is the state transition function

$\mathbf{Q}$ is the process noise covariance matrix.

Eq. (13) was calculated by executing integration with limits $t = t_{k-1}$ and $t = t_k$ with initial conditions

$$\boldsymbol{\phi}(t_{k-1} t_{k-1}) = \mathbf{I}_{n\times n}$$

$$\mathbf{Q}(t_{k-1}) = \mathbf{0}_{n\times n}\ .$$

The state transition function $\boldsymbol{\phi}$ and process noise covariance matrix $\mathbf{Q}$ are calculated by evaluating integration with the same limits as the propagation state estimation error covariance, i.e. the time interval considered is $t \in \{t_k, t_{k-1}\}$. The state transition function will satisfy the differential equation

$$\dot{\boldsymbol{\phi}}(t, t_{k-1}) = \mathbf{F}(\hat{\mathbf{x}}(t), t)\, \boldsymbol{\phi}(t, t_{k-1})\ ,$$

where

$$\mathbf{F}(\hat{\mathbf{x}}(t), t) = \left.\frac{\partial \mathbf{f}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)}\right|_{\mathbf{x}(t)=\hat{\mathbf{x}}(t)}\ .$$

The process noise covariance matrix $\mathbf{Q}$, by definition, will have a time derivative yielding

$$\dot{\mathbf{Q}} = \mathbf{F}(\hat{\mathbf{x}}(t), t)\mathbf{Q}(t) + \mathbf{Q}(t)\mathbf{F}^{\mathrm{T}}(\hat{\mathbf{x}}(t), t) + \mathbf{M}(t)\mathbf{Q}_{spec}\mathbf{M}(t)\ ,$$

where $\mathbf{M}$ is the same process noise mapping matrix used in Eq. (9), and $\mathbf{Q}_{spec}$ represents a constant-power spectral density denoted by $\mathbf{Q}_{spec} = \mathbb{R}^{p\times p}$.

3.2.2 Update Cycle

At the time of a measurement update, we have the observation matrix computation given by

$$\mathbf{Hs} = \begin{bmatrix} \frac{\partial \mathbf{q}_{meas}}{\partial \boldsymbol{\omega}_{BI}} & \frac{\partial \mathbf{q}_{meas}}{\partial \mathbf{q}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{14}$$

The Kalman gain matrix is

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{Hs}_k^{\mathrm{T}} (\mathbf{Hs}_k \mathbf{P}_k^- \mathbf{Hs}_k^{\mathrm{T}} + \mathbf{Rs}_k)^{-1} \ , \tag{15}$$

where

Hs is the linear term of the Taylor series expansion of the measurement function

P is the state error covariance matrix

Rs is the measurement noise covariance matrix.

The covariance update is

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{Hs}_k)\, \mathbf{P}_k^- \ . \tag{16}$$

Recall the state error covariance matrix is unbiased, therefore $\mathbf{P}(t) = \mathrm{E}\{\mathbf{e}(t)\mathbf{e}^{\mathbf{T}}(t)\}$.

The update state estimate with measurement $\mathbf{y}_k$ is

$$\hat{\boldsymbol{q}}_k = \hat{\mathbf{q}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{h}_k(\hat{\mathbf{q}}_k^-)) \ , \tag{17}$$

where

h is the measurement function

$\hat{\mathbf{q}}_k^-$ is the state estimate propagated from the last measurement.

Eq. (13) - (17) tells us if covariance matrix **Rs** is larger than **Q**, the Kalman gain will optimize the update state estimate such that it relies more on the $\hat{\mathbf{q}}_k^-$ state estimate than the new measurement provided by the sensor. In summary, minimizing the uncertainties by proper modeling will cause the filter to converge at a faster rate and optimize efficiently.

3.3 Satellite System Model Plant

3.3.1 Overview

The plant system in terms of the satellite itself has three main parts: a magnetometer for sensing the magnetic field, a torque rod that uses magnetism for attitude adjustments, and the CubeSat structure itself. First, the hardware will be discussed, and then the mathematical models will be shown. In [29] there is a diagram that displays how each subsystem plays its role in the overall plant.

3.3.2 Cube-Sat ACS Hardware

The following descriptions show the subsystems that are to be included inside the frame of the CubeSat. Please see references and navigate to the CubeSat shop website for images of the described systems [30], [31], [32].

3.3.2.1 Attitude Sensors

The attitude sensor desired and expected operational specifications are as follows:

- Features
    - The design is highly standard
    - Side panels are detachable for higher accessibility
    - Support Several PCB sizes
    - Mechanism of Double Kill-switch
    - Extensible design to bigger CubeSat form factors
    - Well-matched with many of CubeSat Shop products
    - Flight Legacy since 2012
- Product properties
    - The product properties are provided in Table 3.2.

Table 3.2: Properties of Cube-Sat ACS hardware. [30]

Property	**Value**	**Unit**
Primary Structure Mass*	87.2	gram
Secondary + Primary Structure Mass*	107.7	gram
Outside Envelope (l x w x h)	100 x 100 x 113.5	mm
Envelope Inside (l x w x h)	~ 98.4 x 98.4 x 98.4	mm
Thermal Range (max – min)	-40 to +80	°C

*Primary mass comprises of Side Frames, Ribs & two Kill Switch devices
Secondary mass contains the Stack Rods.

- Product options
  - Distinct PCB form factors
    - PC/104 CubeSat Kit well-matched (Default)
    - 94 x 94 mm PCBs
    - Conventional designs
  - Different orientations of PCB stack
    - Vertical alignment by default
    - Horizontal alignment (only introduced for custom 94 x 94)
  - Modified structural elements
    - Special mount points
    - Cut-outs
    - Special treatments for surface
- Testing
  - Table 3.3 provided the testing results of the Cube-Sat hardware in which the QT (qualification test) is performed on the qualification/design model, whereas AT (acceptance test) is made on the unit to be transported Flight Heritage.

Table 3.3: Testing results of the Cube-Sat hardware [30]

Test	QT	AT
Vibration	✔	✔
Functional	✔	✘
Mechanical Shock	✔	✘
Thermal Vacuum	✔	✘
Thermal Cycling	✔	✘
Total Ionizing Dose	✘	✘

- Packaging
    - Contents
        - Primary structure portions (ribs, side frames)
        - Secondary structure portions (PCB mounting elements, shear panels) on demand
        - Kill-Switch mechanism (2x)
        - Dummy PCBs
        - Fasteners
        - Installation guide and user manual
    - Packaging
        - Packed in transparent storage case

3.3.2.2 Magnetorquer Rod Nctr-Moo2

This magnetorquer [31] uses an induction coil and, based on how much control is needed, will increase or decrease the current, changing the magnetic field strength, thus pushing against Earth's magnetic field and adjusting the attitude. This actuator was chosen for control because of its low power consumption and having an operation voltage at 5V.

- Features
    - Applications

        - Active hindering for spin steadied, momentum prejudiced, and gravity incline controlled CubeSats
        - Momentum dumping of reaction wheels in 3-axis stabilized spacecraft
    - Less cost standard product
    - High moment for low power
    - Low mass and small size
    - Interface is simple
    - No remaining magnetic moment
- Performance
    - Magnetic moment is greater than 0.2 Am^2
    - Linearity: improved than +/- 5% across the range of operating design
    - Residual moment is less than 0.001 Am^2
    - The range of operating is -35°C to +75°C
    - Power is 200mW from supply of 5V
    - Arbitrary vibration is 14g rms
- Product properties
    - Lifetime more than 10 years
    - Dimensions are 70- millimeter (length) x 9-millimeter (diameter)
    - Total Diameter is less than 9 mm
    - Mounting: knotted and confined direct to PCB
    - Mass is less than 30 g
    - Interfaces: coil wires solder direct to PCB pads
- Qualifications

- The greater NSS magnetorquer rods were 1st coasted in 2014 on the DX-1 and SaudiSat-4 missions. After that, this product line has been distributed to a many of international missions, all with opposing performance necessities. Till now, more than 150 CubeSat rods have also been delivered.

3.3.2.3 MSS Magnetometer

The magnetometer in [32] was chosen because of its small size and once again because the power consumption was relatively small compared to the other sensors.

- Features
  - Low mass and small size
  - Interface options are flexible
  - Radiation tolerant COTS.
  - Supplied with standardization matrix
- Performance
  - Orthogonality is +/- 1°
  - Measurement range is between -60,000nT to +60,000nT
  - Update rate is less than 18Hz
  - Resolution is less than 8 nT
- Noise density is less than 16 nT rms/Hz @ 1 Hz
- Dimensions are 96x43x17mm
- Mass is less than 85g
- Power is less than 750mW
- Thermal (operational) ranges between -25°C to +70

3.3.3 Electronic Configuration

In order to control the magnetorquer hardware and house the actuator drivers and attitude sensors, it is proposed to have a shield designed to mount onto a Raspberry Pi and interface the simulated controller. This would entail translating the current controller in MATLAB to a python script. In order to limit scope, this process has been omitted in modeling and simulation. As a reminder the modeling is assuming an ideal electronic configuration; thus, the circuit is lossless, and all digital communication are hard-linked with no delays being introduced.

3.4 System Modeling

For completeness, the system model is shown combining attitude and pointing control. As mentioned in Chapter 1 the pointing will not be simulated, but it is worth noting how the full model looks. For more information on the following equations see [29].

3.4.1 Attitude Kinematics

The satellite kinematics are represented using Euler angles

$$\begin{bmatrix} \dot{\boldsymbol{\psi}} \\ \dot{\boldsymbol{\alpha}} \\ \dot{\boldsymbol{Y}} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\psi)\tan(\alpha) & \cos(\psi)\tan(\alpha) \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi/\cos(\alpha)) & \cos(\psi)/\cos(\alpha) \end{bmatrix} \boldsymbol{\omega}_b \ , \tag{18}$$

where

$\boldsymbol{\psi}$ represents the roll angle about x-axis

$\boldsymbol{\alpha}$ denotes the pitch angle about y-axis

$\boldsymbol{Y}$ represents the yaw about z-axis.

Eq. (2) is the quaternion representation of Eq. (18).

3.4.2 Magnetic Field and Actuator Models

Magnetic field vectors are represented in the orbit reference frame. To have attitude control in all three axes with only using magnetorquer rods, there needs to be three rods in each satellite. Each rod will correspond to a unique axis where control is desired [29]. The magnetic field vector is given by

$$\mathrm{b}_1 = \frac{M_e}{r_0^3}[\cos(\omega_0 t)\,(\cos(\epsilon)\sin(i) - \sin(\epsilon)\cos(i)\cos(\omega_e t)) - \sin(\omega_0 t)\sin(\epsilon)\sin(\omega_e t)] \quad (19)$$

$$\mathrm{b}_2 = -\frac{M_e}{r_0^3}[(\cos(i)\cos(\epsilon) + \sin(i)\sin(\epsilon)\cos(\omega_e t))] \quad (20)$$

$$\mathrm{b}_3 = \frac{3M_e}{r_0^3}[(\sin(\omega_0 t)(\cos(\epsilon)\sin(i) - \sin(\epsilon)\cos(i)\cos(\omega_e t) - 2\sin(\omega_0 t)\sin(\epsilon)\sin(\omega_e t)] \quad (21)$$

where,

ω_0 - is the orbits' angular velocity relating the inertial frame

r_0 - represents the distance from earths' center to the satellites' center

i - is the orbital inclination

ϵ - represents the magnetic dipole tilt

ω_e - is the Earths' spin rate

M_e - is the magnetic dipole moment of the Earth.

3.4.2.1 Magnetic Dipole Moment

The dipole moment is crucial in determining the max torque the magnetorquer rod can provide. The dipole moment is created by current across the torque rods interacting with Earth's magnetic field. The moment can be described as

$$\mathbf{m}_{control} = \frac{1}{\|\mathbf{b}(t)\|^2}\mathbf{S}^{\mathrm{T}}\big(\mathbf{b}(t)\big)\mathbf{u}\,, \quad (22)$$

where

$\mathbf{m}_{control}$ - is the magnetic dipole moment

$\mathbf{b}(t)$ - is the magnetic field in the spacecraft body reference frame

u - is the control input provided by the controller

$\mathbf{S}^{\mathrm{T}}(\mathbf{b}(t))$ - is the magnetic field cross-product matrix transposed, and can be expanded as

$$\mathbf{S}^{\mathrm{T}}(\mathbf{b}(t)) = \begin{bmatrix} 0 & -\mathrm{b}_3 & +\mathrm{b}_2 \\ +\mathrm{b}_3 & 0 & -\mathrm{b}_1 \\ -\mathrm{b}_2 & +\mathrm{b}_1 & 0 \end{bmatrix}.$$

3.4.2.2 Magnetic Field

The magnetic field '$\mathbf{b}(t)$' is what the torque rods push against to correct attitude. For attitude correction, it is beneficial to use body frame coordinates, therefore we will have an attitude matrix '$\mathbf{A}(\mathrm{q})$' that will be constructed from an initial quaternion, and transformed into a Direction Cosine Matrix (DCM) [33]. The magnetic field can be expressed as

$$\mathbf{b}(t) = \mathbf{A}(\mathrm{q})\ \mathbf{b_0}(t)\ , \tag{23}$$

where

$\mathbf{A}(\mathrm{q})$ – is the attitude matrix and is constructed from the estimated attitude quaternion into a DCM

$\mathbf{b_0}(t)$ – is the magnetic field vector in reference to the Earth centered initial frame.

Given an initial quaternion and estimated attitude quaternions, the $\mathbf{A}(\mathrm{q})$ matrix can transform the quaternion into a DCM by

$$[\mathbf{A}(\mathrm{q})] = \begin{bmatrix} \mathrm{q}_0^2 + \mathrm{q}_1^2 - \mathrm{q}_2^2 - \mathrm{q}_3^2 & 2(\mathrm{q}_1\mathrm{q}_2 + \mathrm{q}_0\mathrm{q}_3) & 2(\mathrm{q}_1\mathrm{q}_3 - \mathrm{q}_0\mathrm{q}_2) \\ 2(\mathrm{q}_1\mathrm{q}_2 - \mathrm{q}_0\mathrm{q}_3) & \mathrm{q}_0^2 - \mathrm{q}_1^2 + \mathrm{q}_2^2 - \mathrm{q}_3^2 & 2(\mathrm{q}_2\mathrm{q}_3 + \mathrm{q}_0\mathrm{q}_1) \\ 2(\mathrm{q}_1\mathrm{q}_3 + \mathrm{q}_0\mathrm{q}_2) & 2(\mathrm{q}_2\mathrm{q}_3 - \mathrm{q}_0\mathrm{q}_1) & \mathrm{q}_0^2 - \mathrm{q}_1^2 - \mathrm{q}_2^2 + \mathrm{q}_3^2 \end{bmatrix},$$

where the quaternion being evaluated is the most recent estimated quaternion provided by the EKF. $\mathbf{A}(\mathrm{q})$ is needed so that the reference frame is body to inertial such that the magnetic field is properly referenced for torque control.

3.4.2.3 Control Torque

With the magnetic moment and magnetic field described, we can now calculate the control torque generated by the magnetorquer rods. The torque generated in each rod is depicted as

$$\mathbf{T} = \mathbf{m}_{control} \times \mathbf{b}(t) = \mathbf{S}\big(\mathbf{b}(t)\big)\mathbf{m}_{control} \,. \tag{24}$$

In Eq. (24) the dipole moment $\mathbf{m}_{control}$ is what our controller adjusts given a magnetic field vector. The magnetic field vector depends on altitude, latitude, and longitude. The synthesis of the magnetic field vector will be explained in more detail in Chapter 4. The vector can be transformed into matrix using the cross-product matrix operation; and $\mathbf{S}\big(\mathbf{b}(t)\big)$ is the result of that operation and can be expanded as

$$\mathbf{S}\big(\mathbf{b}(t)\big) = \begin{bmatrix} 0 & +\mathrm{b}_3 & -\mathrm{b}_2 \\ -\mathrm{b}_3 & 0 & +\mathrm{b}_1 \\ +\mathrm{b}_2 & -\mathrm{b}_1 & 0 \end{bmatrix}.$$

CHAPTER 4:

SIMULATION

4.1 Overview

The goals of this simulation are to validate the proof of concept that a CubeSat constellation is controllable and can be placed in a desired geometrical formation. While designing there are certain control goals:

- Obtain stable design
- Compensate for uncertainties
- Reject disturbance
- Attenuate noise

In order to meet these goals a robust controller must be used. In the previous research mentioned in Chapter 2, a fuzzy controller appears to be an adequate candidate to meet these goals. Please see Appendix B for a brief overview for how the fuzzy controller operates in MATLAB.

The signal flow for this simulation is as follows: attitude quaternions will be generated with the Star Tracker, from here an extended Kalman filter (EKF) will clean up the signal, and now the attitude estimates can be sent into the fuzzy controller to be compared against the reference, in this case a linear constellation; finally, the controller will generate actuator gains and send this information to the appropriate satellite continuously. With this signal flow in mind, the organization of the simulation will be starting with the EKF and then creating the plant, i.e. satellite kinematics, and finally with the fuzzy controller. For the design to be conceptualized graphically, Simulink is used, and MATLAB is used for the framework of functions.

4.2 Actuators

The actuators used in this simulation are three magnetorquer rods. They are arranged such that each 3D axis has an actuator for control. Typically, a reaction wheel is also used for pointing, but as mentioned previously, the simulation focus is on attitude. In order to maintain stability, the angular velocity needs to be constant at steady state; and when precise pointing is desired the angular velocity will be driven to zero [34]. Using the modeling parameters of the magnetorquer rod mentioned in Chapter 3, there are constraints to limit the voltage into the actuator to 5V, also the maximum output cannot exceed 0.2 Am^2.

4.2.1 Actuator Torque Limit

The first steps to determine the total magnetic field intensity are using Eqn. (25), set the torque limit for all three actuators. This was accomplished by using the toolbox in [35] and setting the altitude to 2,000 km. It was determined by using a magnetic field map of Earth (with data from 2015) that a maximum torque of ± 0.001 Nm can be achieved. The max torque value was determined by assuming the magnetic field will remain constant when the altitude is fixed at 2,000 km. A more accurate torque value can be achieved using the current latitude and longitude coordinates of the satellite, thus determining the available torque based on the position of the CubeSat relative to Earth.

4.2.1.1 Torque Limit for Each Axis

To obtain the latitude and longitude time history along its orbit, a two-line element process is employed. Two-line elements contain information about the satellite trajectory. The data needs to be extracted and parsed in order to obtain the pertinent information. The two-line element for an example satellite was obtain with [40]. The data was then parsed using [41]. The data is now properly formatted for the ‘sgp4’ function to be used in MATLAB. This function

will take in the parsed data and time, and return a position vector in the earth-centered earth-fixed (ECEF) frame. This ECEF position vector needs to be converted to a latitude, longitude, and altitude. This transformation is accomplished using the 'ecef2lla' function in MATLAB.

To calculate the magnetic field vector, the 'wrldmagm' function is used; with inputs: latitude, longitude, altitude, year, month, and day. A limitation of the function is that is uses data from the year 2015 as its latest source. The output of the function is then be multiplied by matrix **A**(q) such that **b**(t) is obtained, as seen in Eqn. (24); **b**(t) is the magnetic field sensed in the satellite body frame. The magnetic field vector is then transformed to $\mathbf{S}(\mathbf{b}(t))$, and will be used in Eqn. (23) to determine the magnetic dipole moment. As mentioned previously, the actuators have a limit of 0.2 Am^2, therefore **u** is tuned such that dipole moment doesn't exceed this value. The final step is to multiple the magnetic dipole moment by $\mathbf{S}(\mathbf{b}(t))$; and this resolves to our torque limit for each axis. The torque limit, for different points in time, is then saved in an array in a MATLAB script and then called into the Simulink model. The MATLAB script can be seen in Appendix D and how the limits are called into the Simulink model can be seen in Figure 4.1.

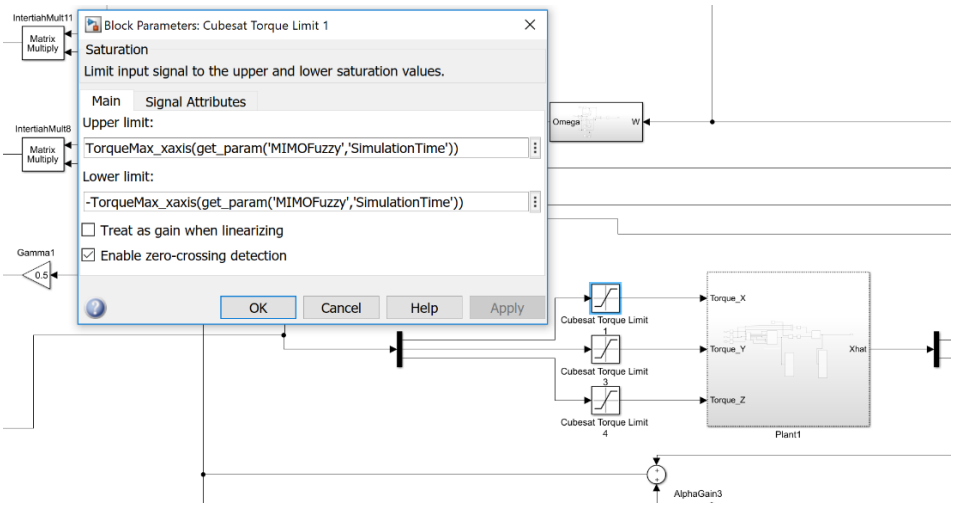

Figure 4.1: Torque limit in Simulink. The torque max array is accessed from the workspace and is indexed using the simulation time. This corresponds to the torque value limits being relative to time. Here, the 'get_param' function is called with the name of the model and the desired info to be accessed, in our case 'SimulationTime'.

4.3 Extended Kalman Filter (EKF)

The code shown in Appendix A was used to describe the EKF. The EKF, for simulation purposes, was implemented with a single line function (built-in MATLAB code) rather than implementing the code in Appendix A. When simulating in Simulink an EKF block was used and a function 'KalmanState' was declared in the workspace. The far-right block in Figure 4.2 is the EKF block [36]; it considers the current actuator torque values and measurements of the system. It incorporates both measurement noise from the Star Tracker and process noise, due to the uncertainties, within the propagation cycle of the EKF. The process noise affects the entire state; and is multiplied into the true state as a diagonal matrix with a defined covariance and zeroes elsewhere. The measurement noise only affects the quaternion portion of the state, and is multiplied into the true measurement. The units for noise covariance matrices will be a combination of the quaternion value and angular velocity (rad/s^2). These noise values are represented as white gaussian noise with covariances 1E-9 and 5E-5 respectively. These covariances were chosen as they are similar to the simulation in [28].

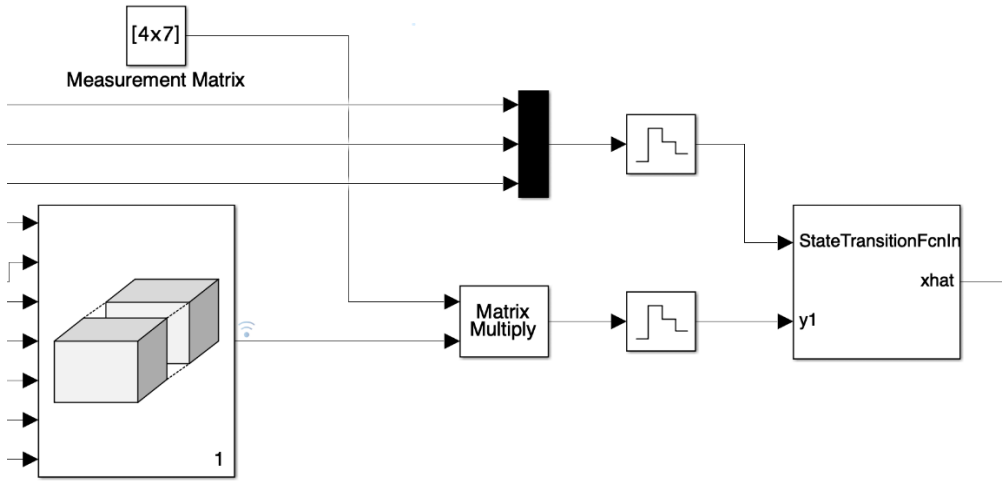

Figure 4.2: EKF in Simulink. This was simulated with a single unit delay between the true states and the estimates. The sampling rate of the filter matches the zero-order hold of 0.001, also known as 1kHz.

4.4 Plant

Using the information of the satellite's kinematics from Chapter 3 Eqn. (2), the kinematic equation is coded into Simulink as seen in Figure 4.3. The angular velocities are modeled as Omega and they can be seen in Figure 4.3. Lastly, the time derivative of the rigid body dynamic equation of the CubeSat is simulated using 'rigbody' from the tool box in [35]. The time derivative is fed into the 'KalmanState' function as

```
dxdt = rigbody(0,x,0,u,inertia,inv(inertia));
```

where,

x – attitude state vector; includes quaternions (4) and angular velocity (3)

u – control signal

inertia – inertia matrix (kg*m*m).

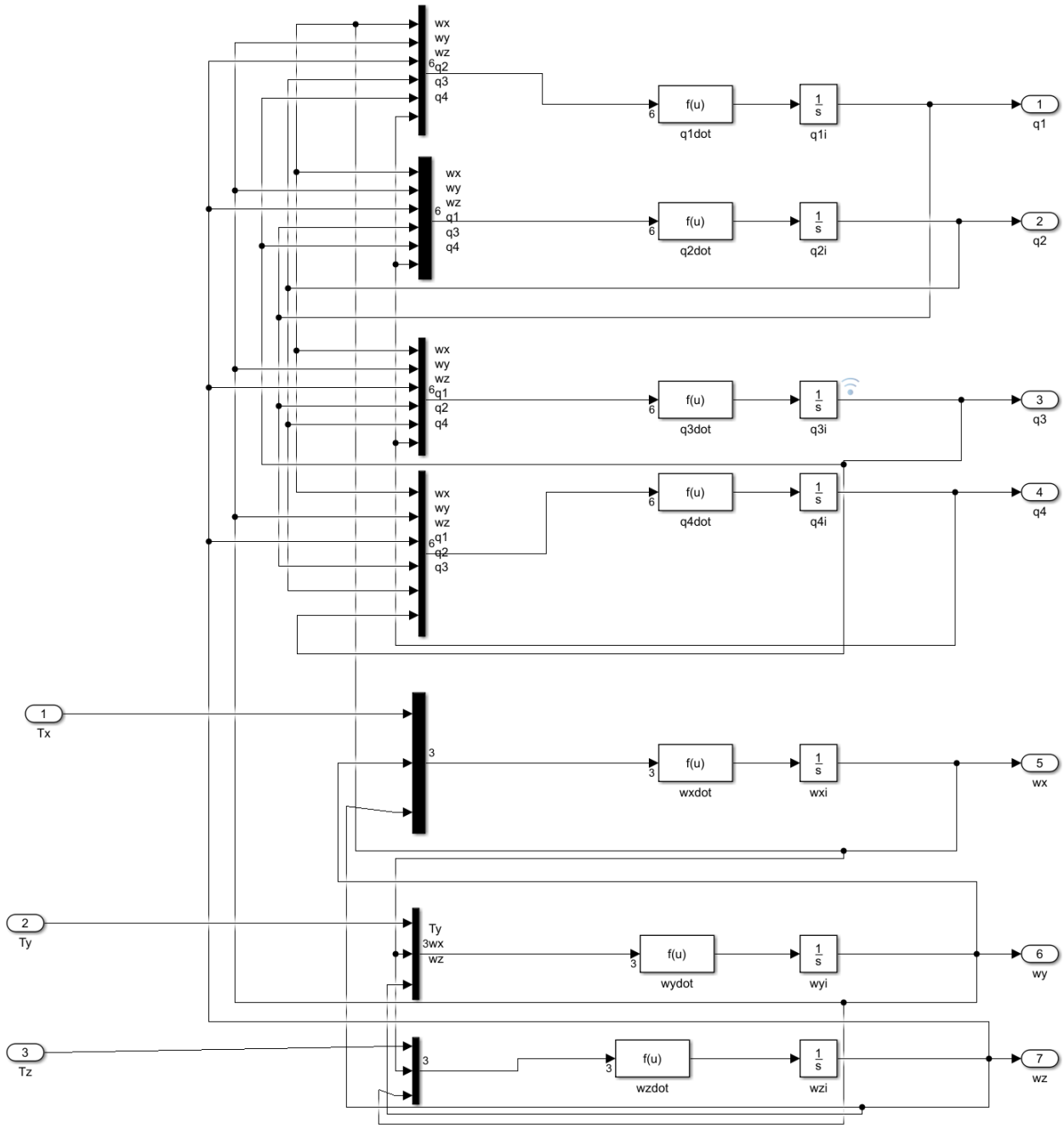

Figure 4.3: Simulink attitude kinematics. These scalar values {q1, q2, q3, q4, wx, wy,wz} are concatenated into state vector ‘x’. The vector is then multiplied with a measurement matrix and fed into the EKF to be evaluated.

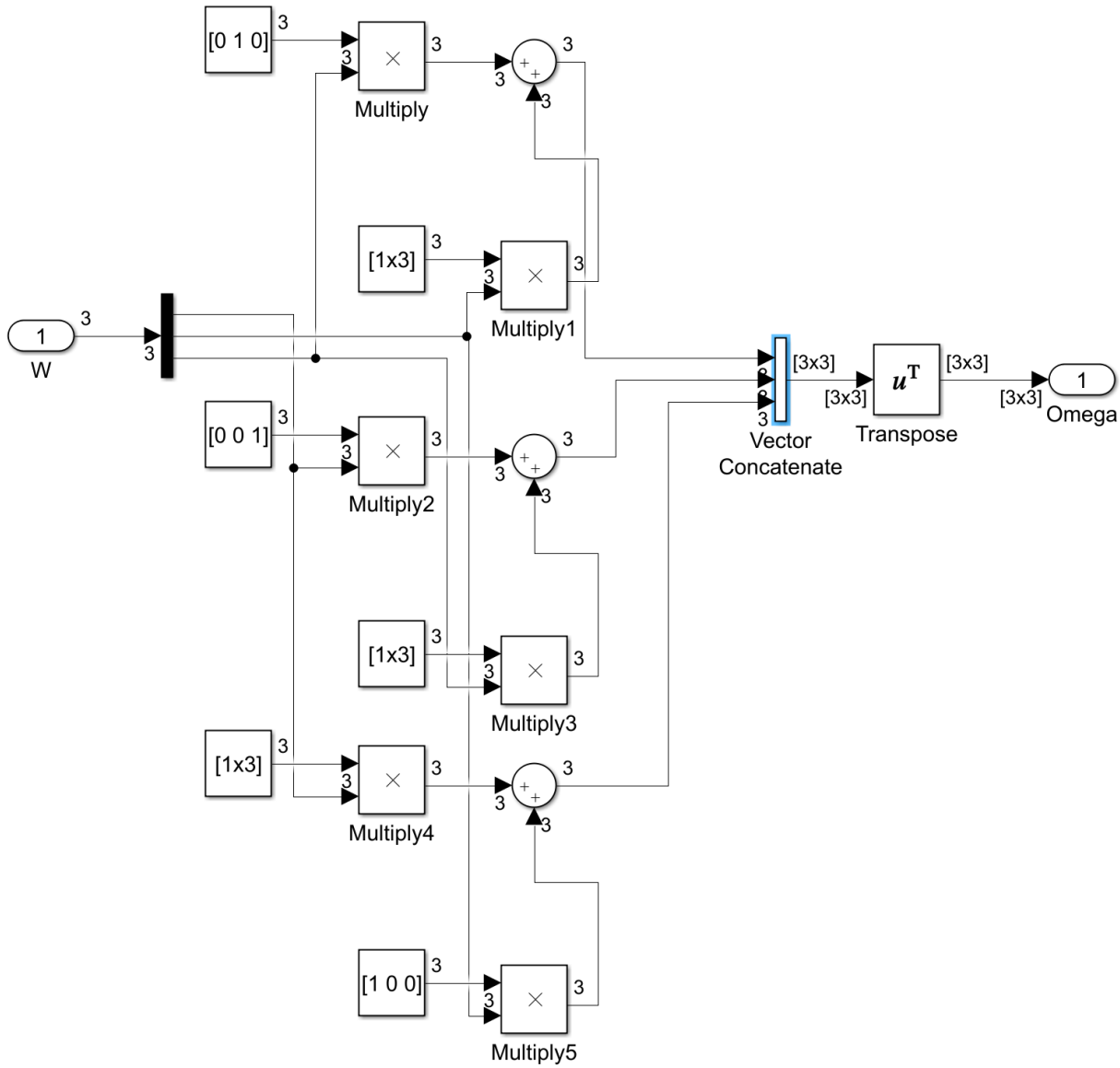

Figure 4.4: Simulink Omega model. This is constructed with the cross-product matrix operation discussed in Chapter 3. Omega is then used as an input to the sliding mode portion of the controller.

4.5 Controller

This is where the novelty of this paper lies. The first approach was to construct a PID controller to serve as a benchmark against the sliding mode fuzzy controller. See Figure 4.5 for the Simulink block diagram including the three PID's, one for each CubeSat. This idea was eventually aborted because the built-in PID tuner could not linearize a MIMO system, therefore a sliding mode controller (SMC) was the next logical controller to construct. Figure 4.6 shows the

first iteration of the SMC and its realization was influenced from the work shown in [37]. There are only two parameters that need to be tuned for the SMC: an ‘alpha’ gain and a ‘gamma’ gain. The two gains were already tuned for an inertia matrix from the paper in [4], therefore these gains were left alone, and a fuzzy controller was then incorporated for this inertia matrix in [4]. This inertia matrix corresponds to a satellite called ‘sampex’. Figure 4.7 shows the fuzzy controller constructed for the ‘sampex’ satellite, and Figure 4.8 shows the quaternion output for this controller. It is clear by looking at the oscillations in Figure 4.8 that this controller could use more tuning, i.e. there is not absolute convergence but merely boundness within ± 0.03 of a unit quaternion.

The paper in [4] did not specifically state how the control rules were defined so a control surface was designed by visualizing and matching the figure supplied. In Chapter 5 the sliding surface will be tuned more appropriately, and the fuzzy limits will be defined by calculating the proportional error and the differentiating error.

In this Chapter the control theory was shown to inform the reader how the modern control methods of SMC and Fuzzification can be implemented. This was shown by way of verification of the work in [4] with a Simulink diagram. This control law seems to be valid based on the results of Figure 4.8. In the next Chapter the inertia matrix will be constructed for a CubeSat; and both the SMC and Fuzzification will be retuned for this system.

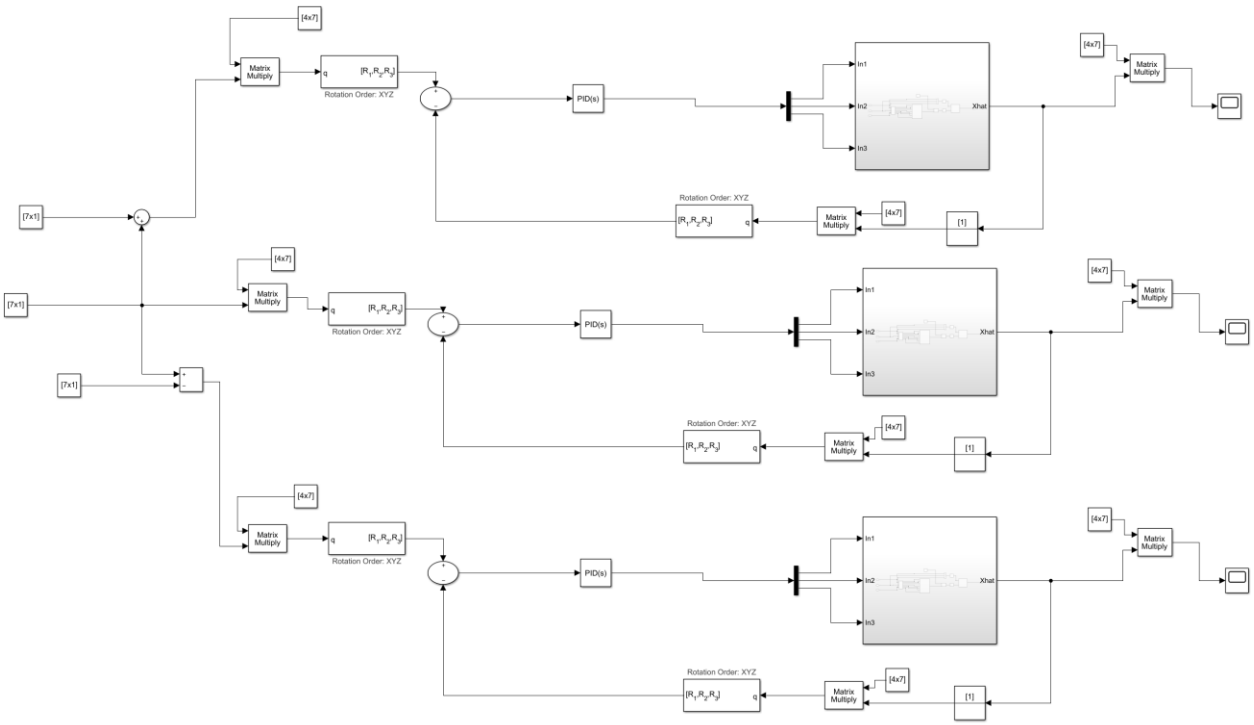

Figure 4.5: Basic PID. This was the first control approach used on the system. The idea was to have the known parameters of the plant in the middle be used as inputs to the other two plants and tune three PID's to align them in a linear fashion. This became feeble due to rigorous manual tuning. The recursive manual tuning was eventually aborted for the quest of a controller that could linearize the system.

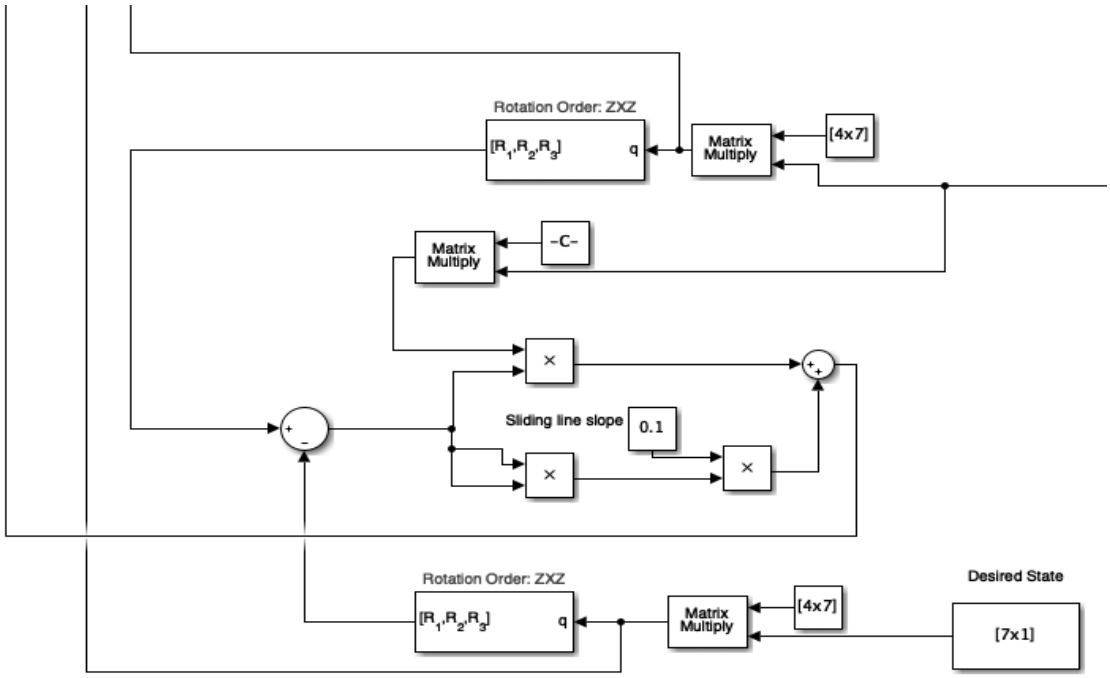

Figure 4.6: SMC architecture. The gamma gain shown as ‘c’ is used to have inputs reach the sliding surface and the alpha depicted as ‘sliding line slope’ is used to maintain the inputs on the sliding surface.

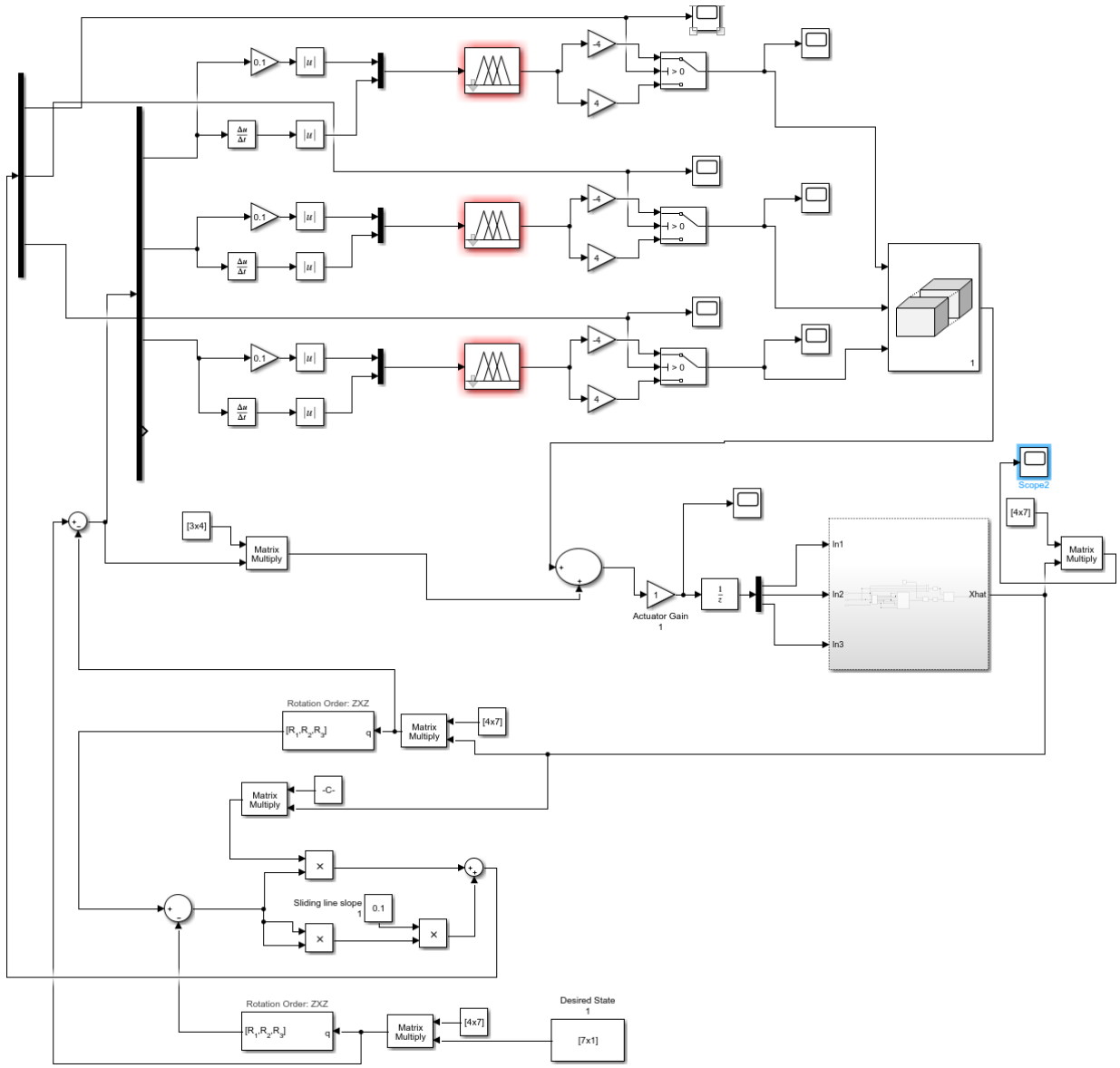

Figure 4.7: SMC with fuzzy logic. This SMC has alpha equal to 0.1 and gamma equal to 0.5. The fuzzy logic is constructed for each axis of control. This controller only controls a single plant with an inertia matrix equal to diag(8,8,12).

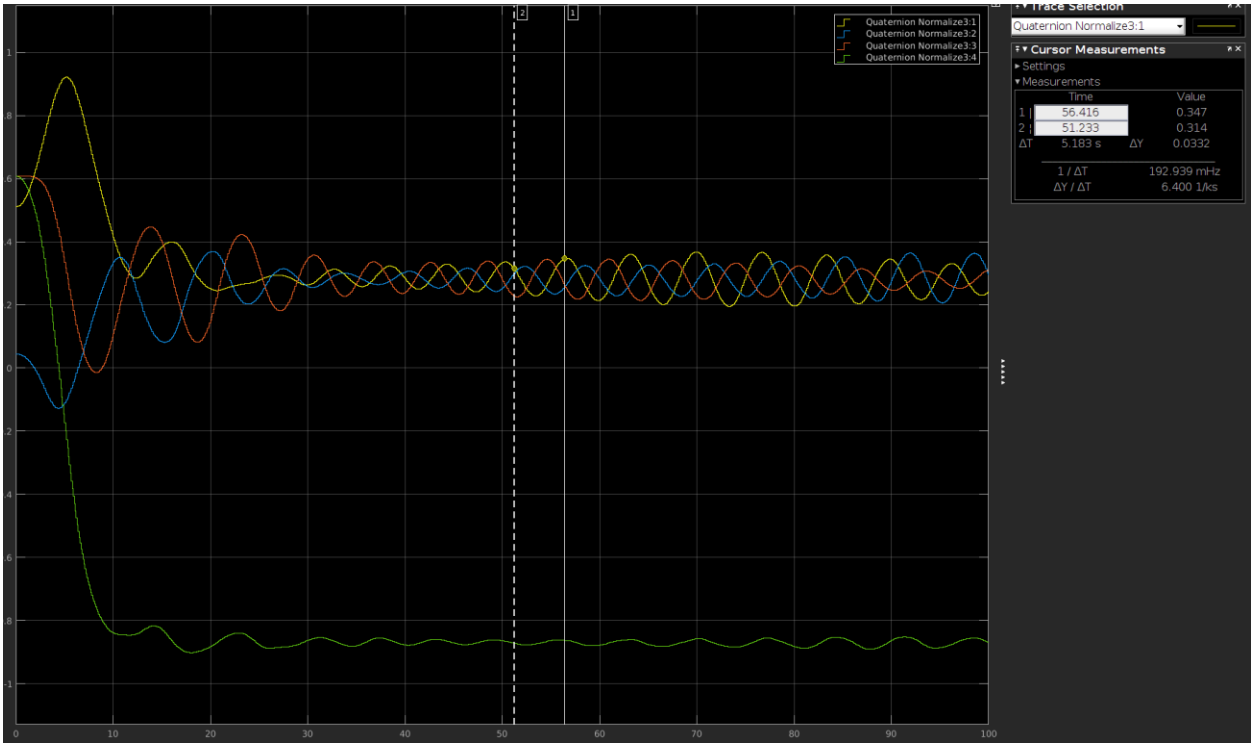

Figure 4.8: Normalized quaternion plot. The scalar quantity is not of real significance as far as desired characteristics are concerned. The vector component was selected to have a desired characteristic of all three quaternion parts converging at 0.28. This wasn't quite achieved as there are residual oscillations in steady state.

CHAPTER 5:

RESULTS AND CONCLUSION

5.1 Results

This section contains the results of the entire system simulation; which is seen in Figure 5.1. As a recap, there are three plants; they are CubeSats defined with an inertia matrix equal to diag(0.001, 0.001, 0.001). The plant subsystem takes in three inputs that are torque gains {Tx, Ty, Tz}, and the outputs are the 7 state variables that the EKF estimates. The controller fuzzy logic is tuned by taking the fuzzy sliding mode control surface that was created in Chapter 4 and rescaling the inputs for a CubeSat plant. This tuning process was achieved by checking the initial values of the proportional error (e_p) and the velocity error (e_d) values, and then setting these as the limits. See Figure A.3 for a view of the fuzzy surface. The SMC portion was tuned by simply adjusting the alpha gain to reduce chatter in steady state. The gamma portion needed no adjustment because the nonlinearities died out relatively quick, with the gamma-gain set to 0.5. Setting alpha to 0.6 appeared to remove some, but not all the chattering; this can be seen in Figures 5.2, 5.3, 5.4. The angular velocities were not part of the measurement function because a gyro was not simulated as a sensor, nevertheless the EKF did well at estimating all three axes. To show robustness of the fuzzy control the attitude error and velocity error are shown in Figures 5.6 and 5.7 respectively.

By analyzing Figures 5.2, 5.3, 5.4 the desired linear constellation can be said to have been achieved and with all CubeSats having the same orientation, i.e. oriented along the same attitude quaternion coordinate. By having the CubeSats all facing in the same direction, the

intersatellite link between them and another system is ready to be initialized. Currently, the three plants achieved steady state in about 5 hours. The next step in controller development is to incorporating pointing at a target location and this will be discussed in the future work section of this chapter.

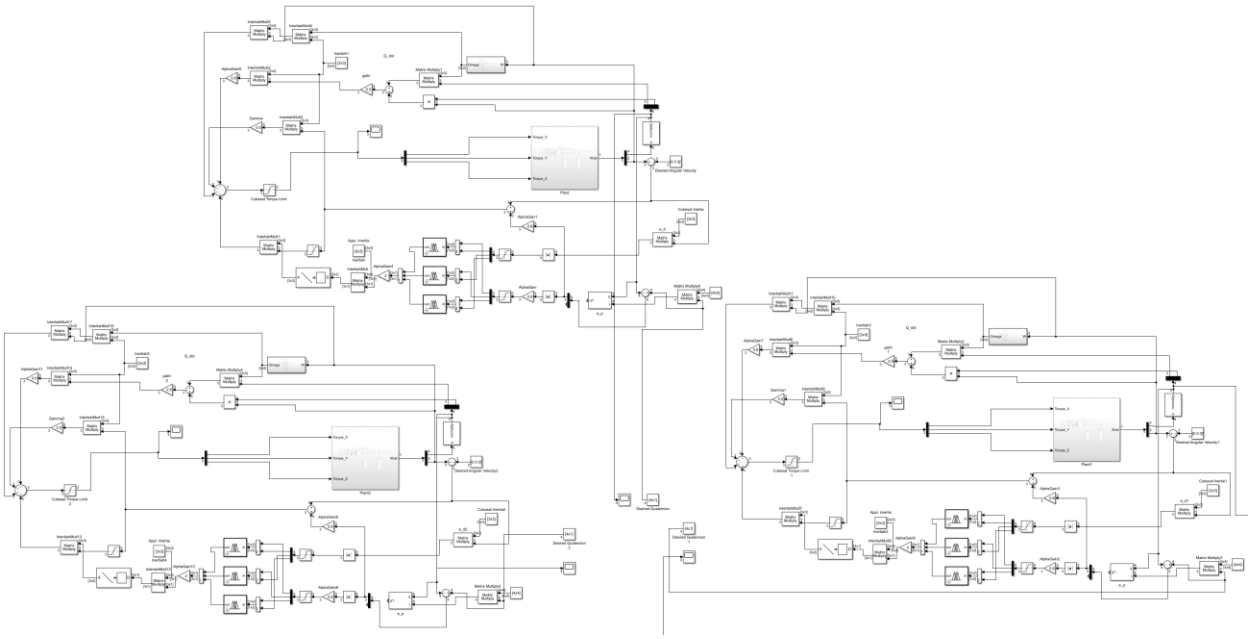

Figure 5.1: MIMO Simulink diagram. This is the same control scheme from Chapter 4 but with two more plants introduced. This approach defines the overall MIMO system by concatenating the three plants quaternion data, however all calculations resolved are isolated within the three plants.

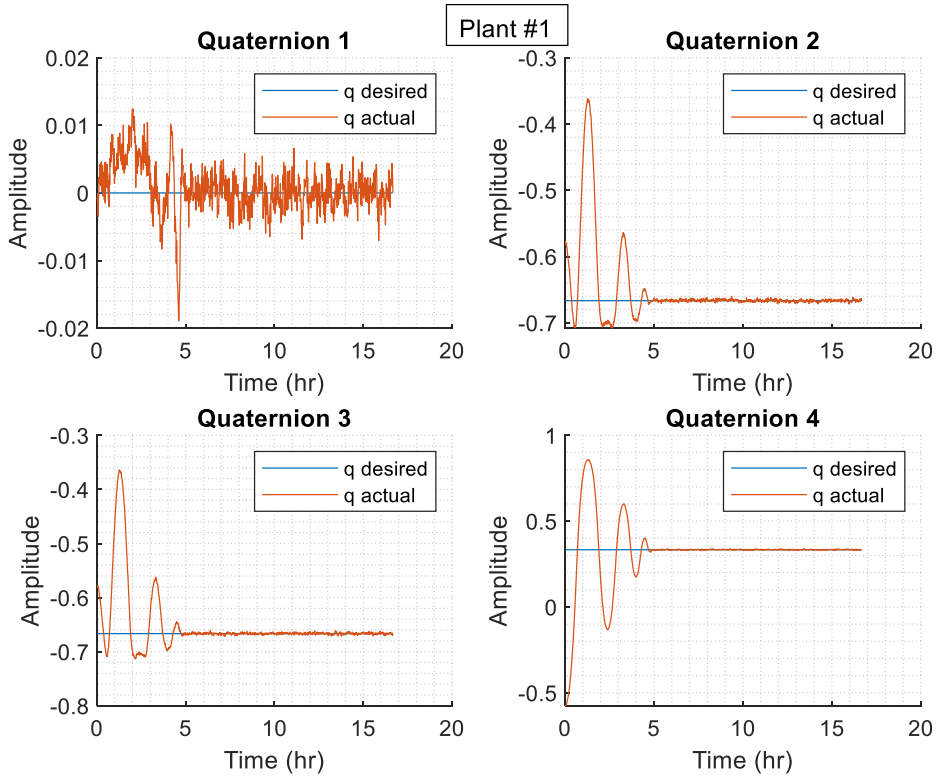

Figure 5.2: Plant #1 quaternion plot. Quaternions 1-3 represents the vector component and quaternion 4 is the scalar component.

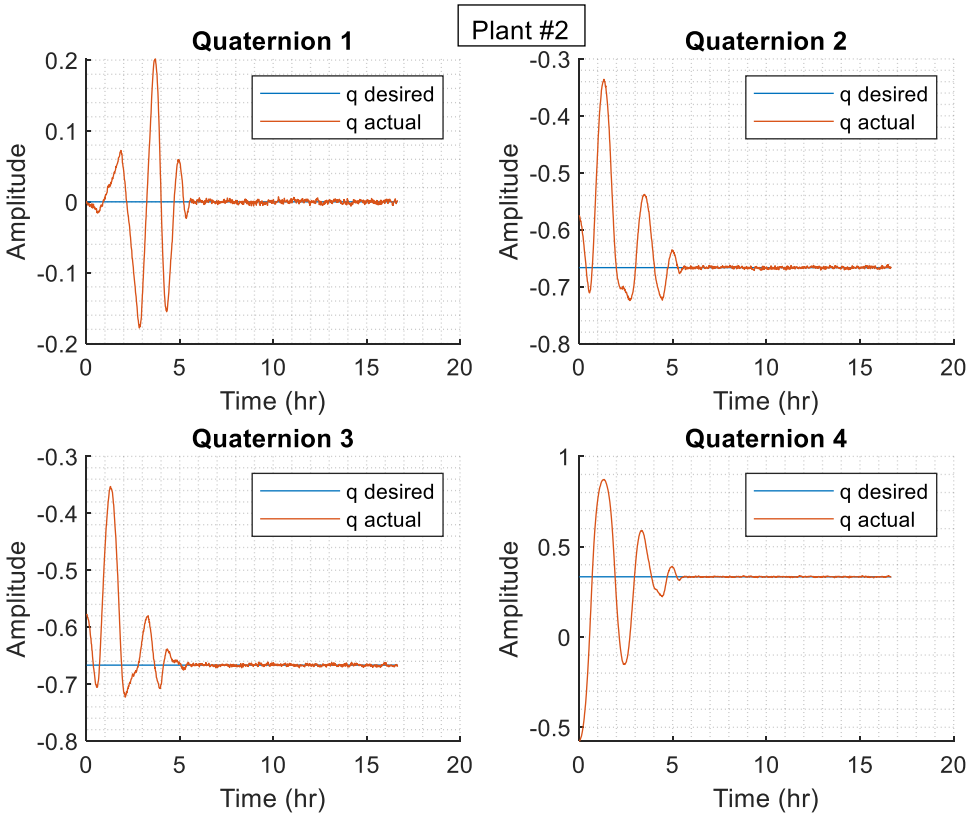

Figure 5.3: Plant #2 quaternion plot. Quaternions 1-3 represents the vector component and quaternion 4 is the scalar component.

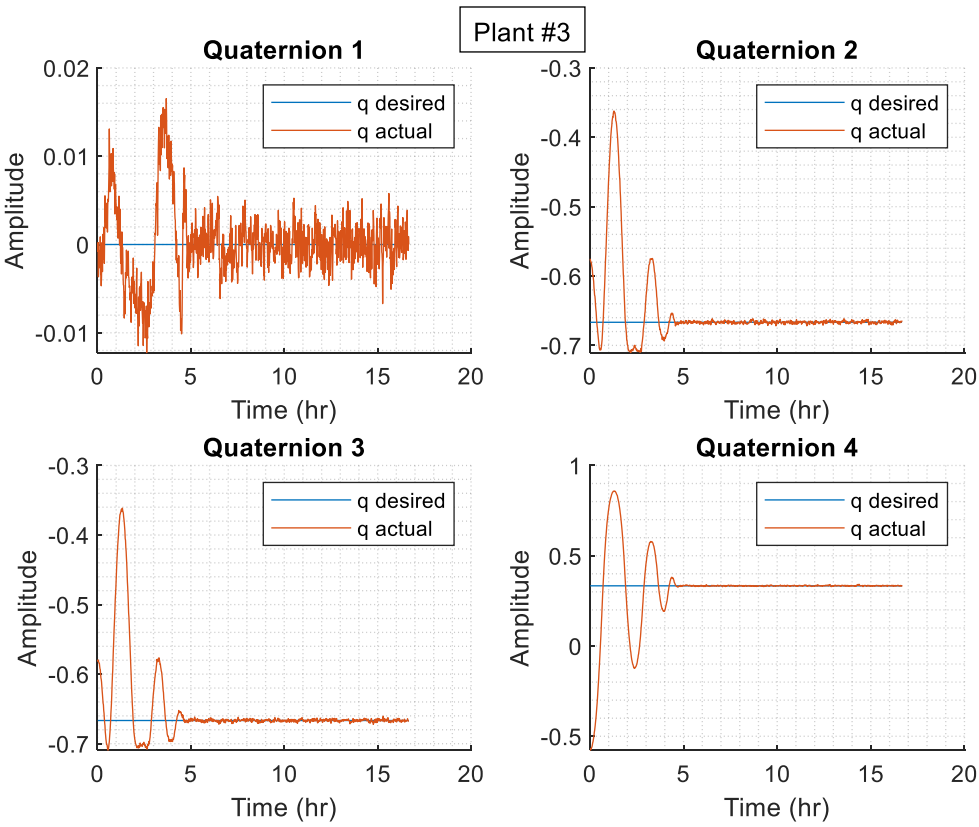

Figure 5.4: Plant #3 quaternion plot. Quaternions 1-3 represents the vector component and quaternion 4 is the scalar component.

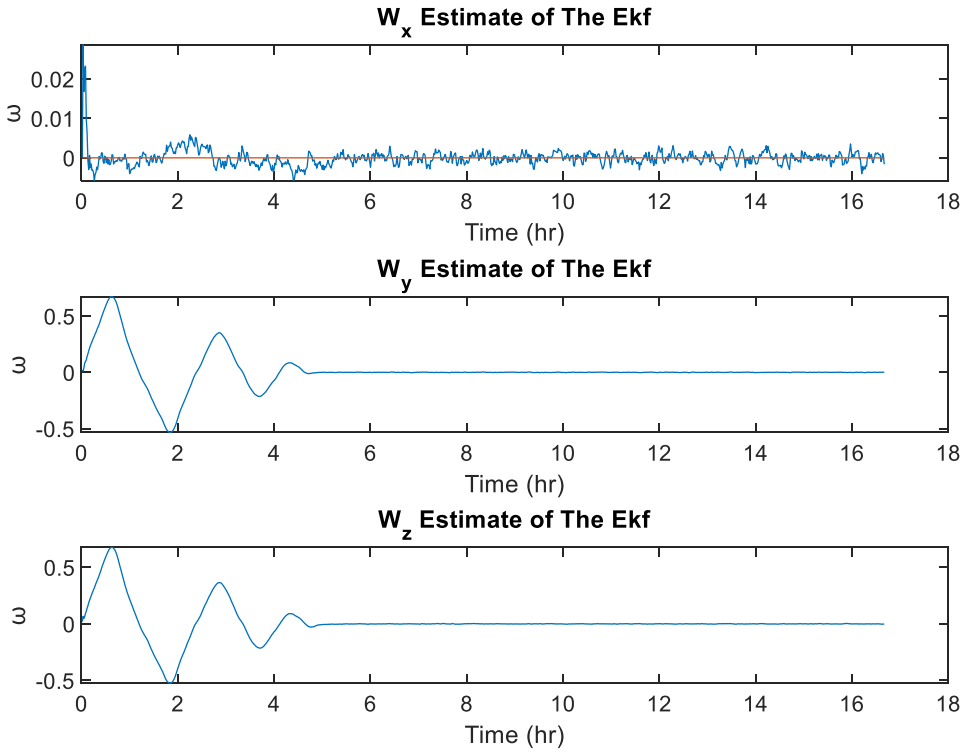

Figure 5.5: Angular velocity estimates. This shows the estimates provided by the EKF. The estimates have relatively low noise due to proper system modeling.

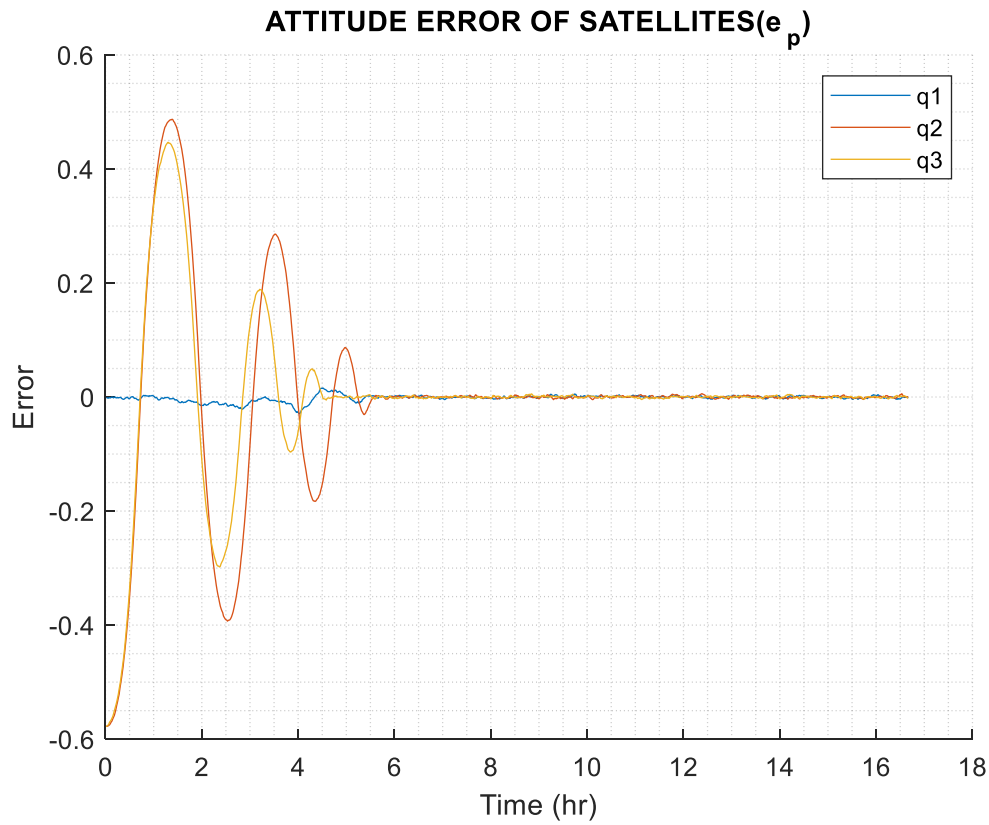

Figure 5.6: Attitude error. This plot shows the proportional error of the controller represented here as attitude.

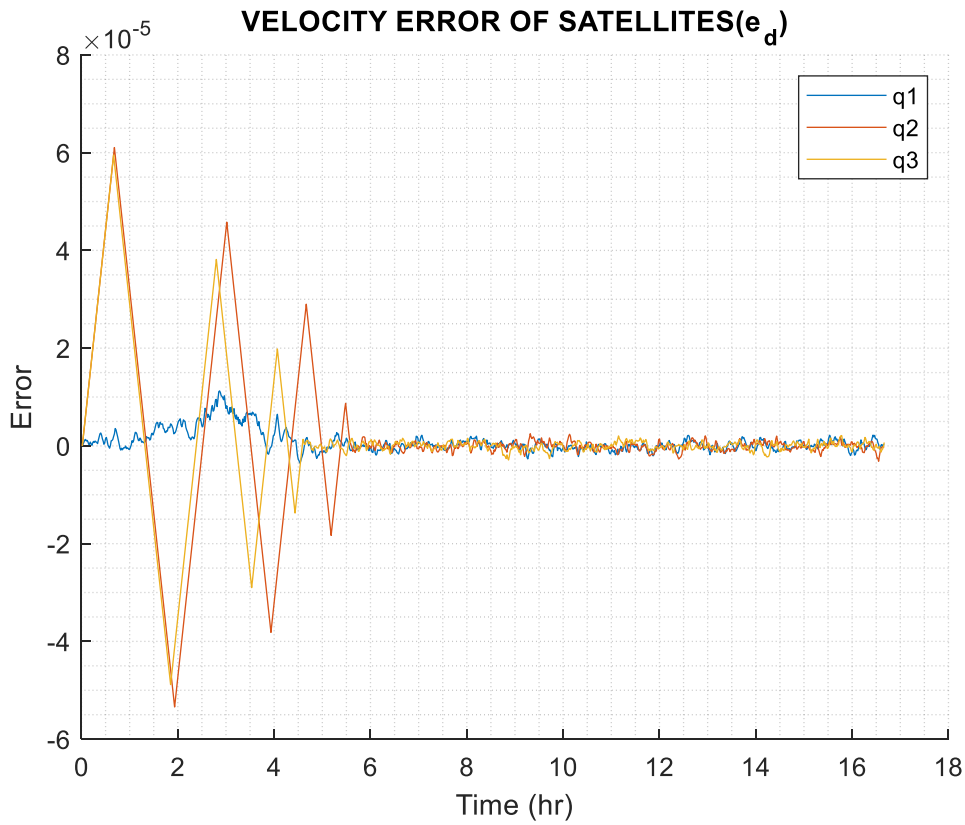

Figure 5.7: Velocity error. This plot shows the differentiating error of the controller represented here as velocity.

5.2 Future Work

The main addition for future work is to consider parameters that incorporating pointing. I will now propose a heuristic solution with a simple proportional controller taking on the reflection coefficient as a parameter (s11). Basically, a simple if-then rule to be used in the fuzzy logic. This would tune a proportional controller that minimizes reflection-coefficient, i.e., gamma, with measurements coming from a directional coupler. In the case where the constellation is stacked linearly in the y-axis, the yaw would be tuned by moving in the direction where gamma is minimized. If complex constellations formations are desired, then machine learning could be incorporated to tune the fuzzy logic based upon a desired cost function to be minimized. Also, constructing a MIMO system using N and D methods that were mentioned in Chapter 2 would be desired so that block diagram is more economically constructed.

5.3 Conclusion

Overall there were a good deal of assumptions that needed to be in place for this simulation to work out the way it did. It would be beneficial to start to analyze a more lossy and delayed system for this controller to be applicable to any real-world scenario. The method of filling out source code into the workspace and calling that information in the Simulink diagram appeared to be the best way to expedite testing and tune the controller parameters. Overall the system worked as a great case study to become familiar with modern control techniques and experience some of the challenges that come with controlling CubeSats.

The intersatellite link is crucial for communications that exist outside of the network contained within a constellation. This controller dealt with the receive portion of communication; meaning the three satellites were sent to a desired quaternion where they were aligned linearly, and then would hold position for a downloading protocol to be established. The assumptions may cause the controller design to be susceptible to real-world scenarios, however the proof of concept was proven, and a more optimized solution should be considered and take into account far-field communication.

REFERENCES

[1] A. Sharma, V. Hastir and D. S. Saini, "Transmit power optimization in Cognitive Radio Networks using game theoretic approach," *2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014)*, Ajmer, 2014, pp. 312-316.

[2] Park, Hyun Jae, Gyu-min Lee, Seung-Hun Shin, Byeong-hee Roh and Ji Myeong Oh. "Implementation of Multi-Hop Cognitive Radio Testbed using Raspberry Pi and USRP." *IJITN* 9.4 (2017): 37-48.

[3] S. Dimitrijevic and D. Antic, "Satellite antenna positioning using two inputs single output robust fuzzy controller," *4th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services. TELSIKS'99 (Cat. No.99EX365)*, Nis, Yugoslavia, 1999, pp. 427-429 vol.2.

[4] D. Mitic, M. Milojkovic and D. Antic, "Tracking System Design Based on Digital Minimum Variance Control with Fuzzy Sliding Mode," *2007 8th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services*, Nis, 2007, pp. 494-497.

[5] R. Rusli, R. Nagarajan and M. A. Rahim, "Three-axis fuzzy variable structure control with an application to micro satellite," *2009 International Conference on Space Science and Communication*, Negeri Sembilan, 2009, pp. 24-29.

[6] "Phased Array Antennas." Microwaves101, www.microwaves101.com/encyclopedias/phased-array-antennas#tip.

[7] Quinn, Matt. "Why and How to Use a Star Tracker for Photos of the Night Sky." *PetaPixel*, 24 July 2018, petapixel.com/2018/07/24/why-and-how-to-use-a-star-tracker-for-photos-of-the-night-sky/.

[8] Dunbar, Brian. "Star Trackers Lights the Way." *NASA*, NASA, www.nasa.gov/multimedia/podcasting/StarTrackers.html.

[9] Labane Chrif, Zemalache Meguenni Kadda, "Aircraft Control System Using LQG and LQR Controller with Optimal Estimation-Kalman Filter Design." *Procedia Engineering*, Volume 80,2014, Pages 245-257

[10] "News: Breaking Stories & Updates." *DIRNews*, 2017, dir.md/wiki/Kalman_Filter?host=wikipedia.org.

[11] Mithi. “Sensor Fusion and Object Tracking Using an Extended Kalman Filter Algorithm - Part 1.” *Medium.com*, Medium, 10 May 2017, medium.com/@mithi/object-tracking-and-fusing-sensor-measurements-using-the-extended-kalman-filter-algorithm-part-1-f2158ef1e4f0.

[12] Mithi. “Sensor Fusion and Object Tracking Using an Extended Kalman Filter Algorithm - Part 2.” *Medium.com*, Medium, 10 May 2017, medium.com/@mithi/sensor-fusion-and-object-tracking-using-an-extended-kalman-filter-algorithm-part-2-cd20801fbeff.

[13] Beech, Theresa W, et al. “a Study of Three Satellite Constellation Design Algorithms”. 1999, issfd.org/ISSFD_1999/pdf/CO1_3.pdf.

[14] Tasevski, ByStefan. “Drone Swarm Communication and Control Architectures.” Drone Below, 7 Dec. 2018, dronebelow.com/2018/12/07/drone-swarm-communication-and-control-architectures/.

[15] Princeton Satellite Systems, Inc. “Formation Flying Module.” Psatellite, 2011, support.psatellite.com/sct/pdfs/ff_manual.pdf.

[16] Muri, Paul; McNair, Janise (1 April 2012). "A Survey of Communication Sub-systems for Intersatellite Linked Systems and CubeSat Missions". Journal of Communications. 7 (4).

[17] Rabideau, D.J. “MIMO Radar Aperture Optimization.” Lincoln Laboratory, 2011, apps.dtic.mil/dtic/tr/fulltext/u2/a536191.pdf.

[18] R. R. Nair, L. Behera, V. Kumar and M. Jamshidi, "Multisatellite Formation Control for Remote Sensing Applications Using Artificial Potential Field and Adaptive Fuzzy Sliding Mode Control," in IEEE Systems Journal, vol. 9, no. 2, pp. 508-518, June 2015.

[19] Allgeier, Shawn E. “Determination of Motion Extrema in Multi-Satellite Systems. University of Florida”, 2011, ufdcimages.uflib.ufl.edu/UF/E0/04/32/86/00001/Allgeier_S.pdf.

[20] T. Y. Somova, "Guidance and economical digital control of a satellite orientation in initial modes," 2018 25th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS), St. Petersburg, 2018, pp. 1-5.

[21] “MIMO Transfer Functions.” Mathworks, 2018, www.mathworks.com/help/control/ug/mimo-transfer-function-models.html.

[22] “MIMO Control System.” Mathworks, 2018, www.mathworks.com/help/control/ug/build-a-model-of-a-multi-input-multi-output-mimo-control-system.html.

[23] “Foundations of Fuzzy Logic.” Mathworks, 2018, www.mathworks.com/help/fuzzy/foundations-of-fuzzy-logic.html?searchHighlight=fuzzy control&s_tid=doc_srchtitle.

[24] Utkin, Vadim. “Sliding Mode Control.” Control Systems, Robotics and Automation Vol. XIII, Encyclopedia of Life Support Systems (EOLSS).

[25] Sekhri, Even. “Design of a Robust Controller Using Sliding Mode Technique for a Linear Belt-Driven System.” *Mechatronics Program*, Tallinn University of Technology, 2017.

[26] Adnane, Akram & Bellar, A & SI MOHAMMED, M.A. & Foitih, Zoubir. (2015). Spacecraft Attitude Estimation Based on Star Tracker and Gyroscope Sensors.

[27] https://www.cubesatshop.com/product/kul-star-tracker/

[28] Jian Li, Xinguo Wei and Guangjun Zhang, “An Extended Kalman Filter-Based Attitude Tracking Algorithm for Star Sensors” 21 August 2017.

[29] Li, Junquan & Post, Mark & Wright, Thomas & Lee, Regina. (2013). Design of Attitude Control Systems for CubeSat-Class Nanosatellite. Journal of Control Science and Engineering. 2013. 10.1155/2013/657182.

[30] https://www.cubesatshop.com/product/1-unit-cubesat-structure/

[31] https://www.cubesatshop.com/product/nctr-m002-magnetorquer-rod/

[32] https://www.cubesatshop.com/product/nss-magnetometer/

[33] K. M. Srinivasa, “Spacecraft Attitude Control Using Magnetic Actuators”, Michigan Technological University, 2015.

[34] Srinivasa, Karthik M. “Spacecraft Attitude Control Using Magnetic Actuators.” Dissertations, Master's Theses and Master's Reports, Michigan Technological University, 2015,digitalcommons.mtu.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1895&context=etds.

[35] Carrara, V. “An Open Source Satellite Attitude and Orbit Simulator Toolbox for Matlab.” DINAME 2015 – Proceedings of the XVII International Symposium on Dynamic Problems of Mechanics. Natal, RN, Brazil, Feb 22-27, 2015, http://www.dem.inpe.br/~val/projetos/propat/default.htm

[36] “Extended Kalman Filter.” Mathworks, MATLAB/Simulink, 2018, www.mathworks.com/help/control/ref/ekf_block.html.

[37] Walchko, Kevin J. “Development of a Fuzzy Sliding Mode Controller for Satellite Attitude Control.” Mechanical Engineering Graduate Student Research Program, University of Florida , 2000, walchko.github.io/Publications/walchko-GSRP.pdf.

[38] Dugas, Olivier. “Void's Vault.” Extended Kalman Filter With Quaternions for Attitude Estimation - Void's Vault, blogdugas.net/blog/2015/05/10/extended-kalman-filter-with-quaternions-for-attitude-estimation/.

[39] Phadnis, Mandar. “Feedback Control of CubeSat for Attitude and Trajectory Corrections.” Department of Electrical and Computer Engineering, University of Houston, Dec. 2013, uh-ir.tdl.org/handle/10657/1553.

[40] CelesTrak, www.celestrak.com/NORAD/elements/cubesat.txt.

[41] “Satellite Orbit Computation - File Exchange - MATLAB Central.” *Satellite Orbit Computation - File Exchange - MATLAB Central*, www.mathworks.com/matlabcentral/fileexchange/28888-satellite-orbit-computation?focused=5170123&tab=function.

[42] Bishop, Robert H. “Precision Lunar Landing Navigation Mathematical Specifications.” 2014. Prepared for Astrobotics, Inc.

[43] Malcolm D. Shuster, “A Survey of Attitude Representations,” *The Journal of the Astronautical Sciences,* Vol. 41, No. 4, 1993, pp. 439-517.

APPENDICES

Appendix A: EKF MATLAB Code

The following code is used to better understand how EKF can be implemented in MATLAB. This script used code from [38] and was tested on inertia values to understand the estimated response.

```
%Net Torque is due to gravity and angular momentum. Motor Torque is due to
%magnetorques on the spacecraft

%TODO May be Integrate height measurements as well to the state equations.
%From Omega's equations, we can receive estimates for height as well.
%Currently I am not sure but may be this may help for better omega
%predictions from the current one

function [state_estimate,covariance_estimate] =
EKF(state,measurement,state_covariance,NetTorque,MotorTorque)
q_last = [state(4) ; state(5); state(6); state(7)];
w_last = [state(1) ; state(2); state(3)];

Ts = 0.01;
t = 0;
tspan = [t, t+Ts/2, t+Ts];
%Time Increment For Numerical Integration

Ix = 40.45;
Iy = 42.09;
Iz = 40.36;
Inertia = diag([Ix Iy Iz]);
options = odeset('abstol', 1e-4, 'reltol', 1e-4);

[T, Y] = ode45('rigbody', tspan, vertcat(q_last,w_last), options, [0 0 0]', Inertia, inv(Inertia));
 att_vec = Y(3, :)';        % propagated attitude vector
 quat = att_vec(1:4);       % propagated quaternion
 w_ang = att_vec(5:7);      % propagated angular velocity

%Test Inertia Values

%Covariance Set
% CovarianceOfStarTracker = 10^-9;
%
% %Altitude Vector Altitude is used In Omega Prediction But currently not
% %implemented since there is no altitude mechanics equation available to us.
% h = [15 15 15];
%
```

```
% %State Equation of the Quaternion
% q_prop = q_last + ((Omega(w_last) * q_last) * Ts / 2);
% %w_prop = (inv(Inertia) * ((NetTorque - MotorTorque) - w_last * Inertia * w_last)) * Ts *
w_last;
%
% %State Equation Of the Omega
% w_prop = ((inv(Inertia) * (-transpose(w_last) * Inertia * w_last)) * Ts) * w_last;
%
% %% State Equation Jacobian Tested And Approved by Mathematica as well
% WdotPartialW = [0 (-h(3)/Ix) - ((-Iy + Iz) * w_last(3)/Ix) (h(2) / Ix)-((-Iy + Ix) * w_last(2)
/Ix);...
%           (-h(3)/Iy) - ((Ix - Iz) * w_last(3)/Iy) 0 (-h(1)/Iy) - ((Ix - Iz) * w_last(1)/Iy);...
%           (-h(2)/Iz) - ((-Ix + Iy) * w_last(2)/Iz) (h(1)/Iz) - ((-Ix + Iy) * w_last(1)/Iz) 0];
%
% QDotPartialW = [q_last(4) -q_last(3) q_last(2);q_last(3) q_last(4) -q_last(1);-q_last(2)
q_last(1) q_last(4); -q_last(1) -q_last(3) -q_last(4)];
% QDotPartialQ = [0 w_last(3) -w_last(2) w_last(1); -w_last(3) 0 w_last(1) w_last(2); w_last(2)
-w_last(1) 0 w_last(3);-w_last(1) 0 -w_last(2) -w_last(3)];
%
% UpperSide = horzcat(WdotPartialW,zeros([3 4]));
% LowerSide = horzcat(QDotPartialW,QDotPartialQ);
%
% Phi = vertcat(UpperSide,LowerSide)*Ts + eye(7);
% %Process Noise Matrix right now it is constant
% Q = diag([4 * 10^-3 1 * 10^-3 2 * 10^-3 10^-3 10^-3 3 * 10^-3 2 * 10^-3]);
%
% %% Axuillary Matrices for calculations Kalman Gain and Observation Matrix
% CovarianceEstimate = Phi * state_covariance * transpose(Phi) + Q;
% Hs = [ 0 0 0 1 0 0 0;
%       0 0 0 0 1 0 0;
%       0 0 0 0 0 1 0;
%       0 0 0 0 0 0 1];
% KalmanGain = CovarianceEstimate * transpose(Hs) * inv(Hs * CovarianceEstimate *
transpose(Hs) + diag([10^-4 10^-4 10^-4 10^-4]));
% %% Finally set the estimated Values
% covariance_estimate = (eye(7) - KalmanGain * Hs) * CovarianceEstimate;
% state_estimate = state + KalmanGain * (measurement - Hs * (vertcat(w_prop,q_prop)));

end

% function [ mu1, sigma1, w ] = EKF( mu0, sigma0, u1, z, errZ, sigr, sigw, dt)
% % EKF with quaternions as seen in
% % 'Indirect Kalman Filter for 3D Attitude Estimation'
% % by Trawny and Roumeliotis
% % Author of test code: Olivier Dugas, ing. Jr
```

The code above can be streamlined using a MATLAB built in script 'extendedKalmanFilter'. In the following script it can be seen that state function is the modeling of the kinematics of the satellite. The process noise models the uncertainties contained in the propagation cycle. The measurement function is the modeling of what is being measured, in this case it is a star tracker camera generating quaternions with noise being introduced in the form of white gaussian noise being multiplied with the true state.

```
state_func = @(x,u)(KalmanState(x,u));
%
%% Kalman State
function state = KalmanState(x,u)
%   Detailed explanation goes here
Ix = .001;
Iy = .001;
Iz = .001;
inertia = diag([Ix Iy Iz]);
dxdt = rigbody(0,x,0,u,inertia,inv(inertia)); %x is quaternion and angular velocity, u is torque
Ts = 0.1;
state = x + dxdt * Ts;
end

ekf = extendedKalmanFilter(state_func,@MeasurementFunc,[1 1 1 1 0 0 0]');
ekf.ProcessNoise = diag([5*10^-5 5*10^-5 5*10^-5 5*10^-5 5*10^-5 5*10^-5 5*10^-5]);
ekf.MeasurementNoise = diag([10^-9 10^-9 10^-9 10^-9]); %Covariance noise of star tracker
noise=.001*randn([4,m+1]); %random addictive white noise
noisen=[noise/norm(noise)]; %normalized
Xn(1:4)=qmult(Xn(1:4),noisen(:,ii)); %mixing noise into true state of attitude quaternions

%Simulates only the star tacker quaternions being measured from the state%
function meas = MeasurementFunc(state)
meas = [1 0 0 0 0 0 0;0 1 0 0 0 0 0;0 0 1 0 0 0 0;0 0 0 1 0 0 0] * state;
end
```

Appendix B: Fuzzy Control Concepts

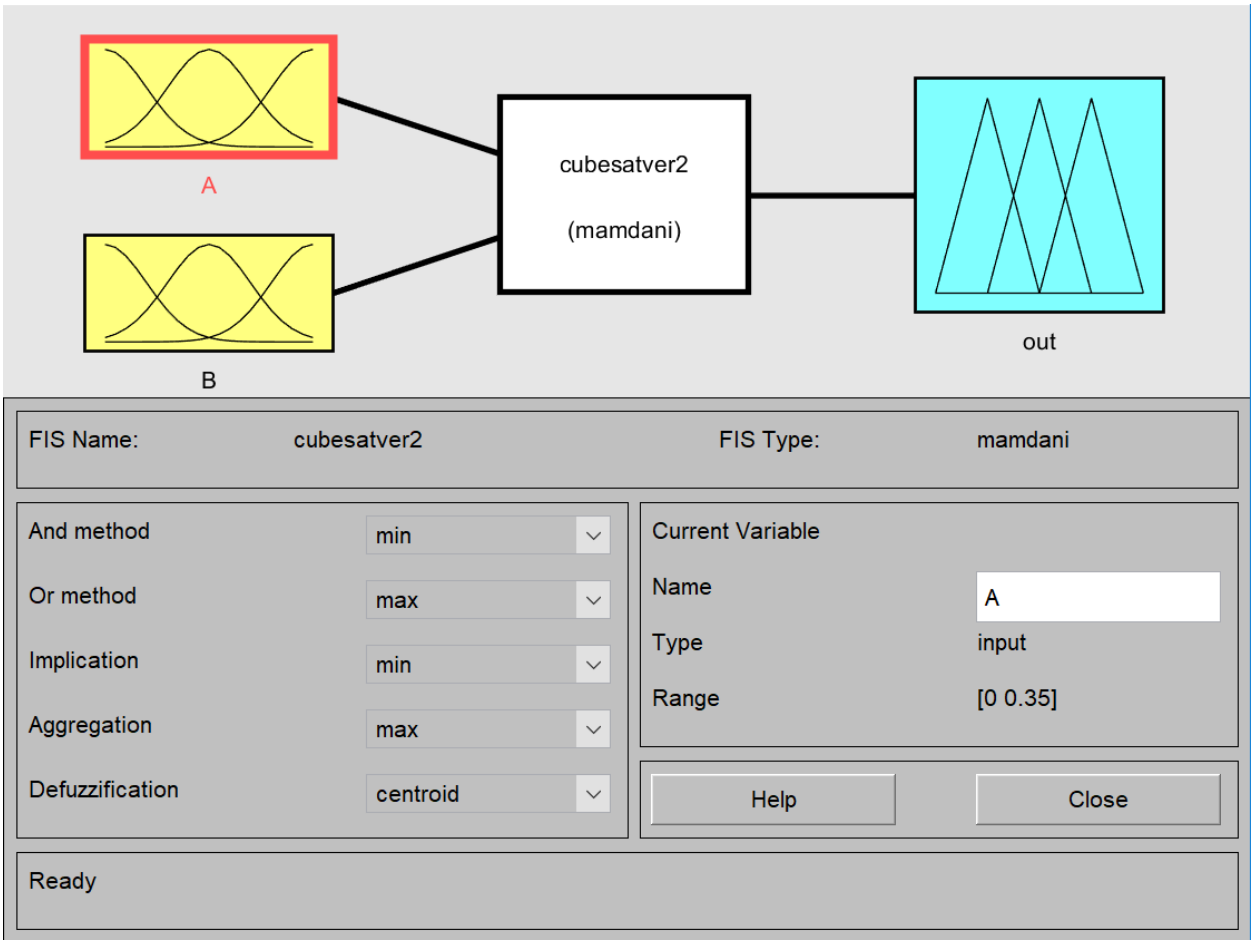

Figure A.1: MATLAB Fuzzy Toolbox interface. This shows how the fuzzification and defuzzification are resolved. For simplicity the controller was selected to use min, max, and centroid options. A and B correspond to the inputs of the proportional error and the derivative error respectively. The output corresponds to the control signal that will be delivered to the actuator.

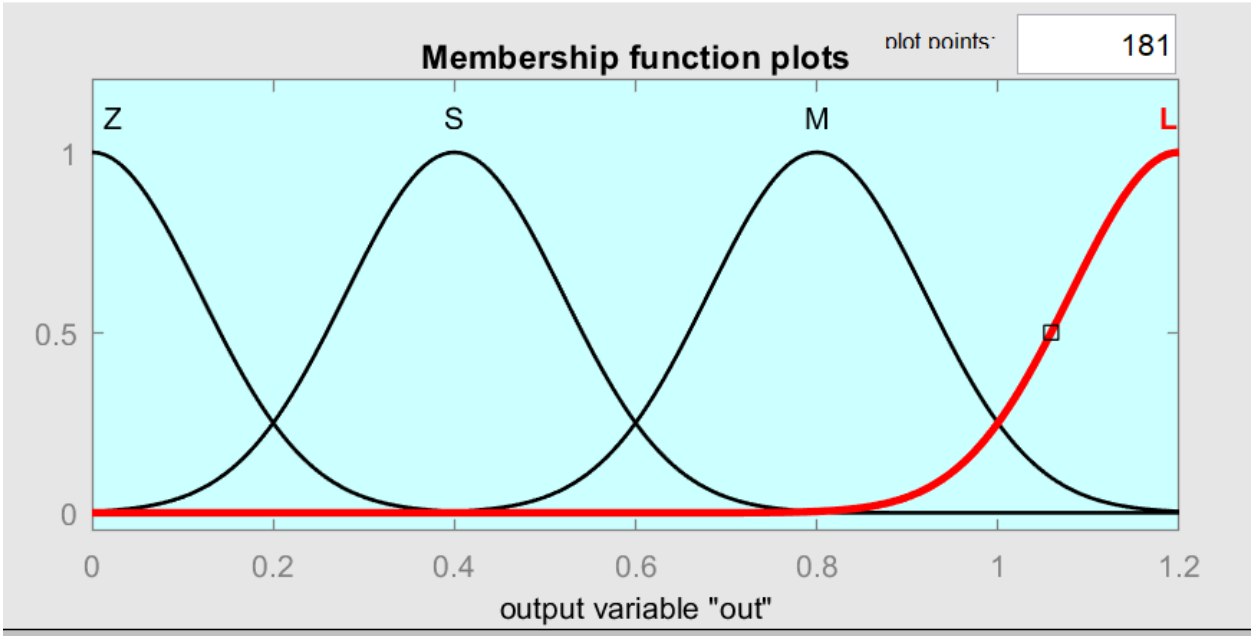

Figure A.2: Membership function plot. This shows four membership functions that correspond to zero, small, medium, large gains. The infimum is 0; this is a necessary condition for stability. The max gain is selected to 1.2 due to control limits on the actuator. The rule base that depicts which membership function the input gets mapped to is not trivial and these rules are defined in [37].

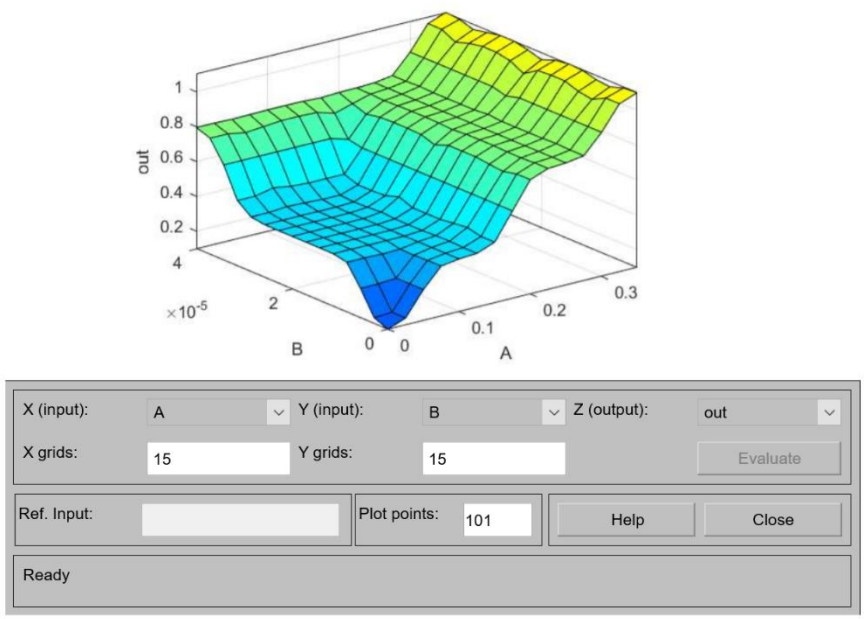

Figure A.3: Surface viewer. This helps visualize how the fuzzy controller assigns an output depending on how large or small the system errors are. If there are no errors then output is 0.

Appendix C: Quaternion Concepts

Euler's Theorem says any rotation can be denoted by a single angle called the Euler Angle. This Euler Angle is in reference to a Euler Axis. Much like going from polar coordinates to cartesian coordinates, a Euler Angle can be represented as a quaternion. A quaternion has four components [39]:

$$q = q_0 + q_1 i + q_2 j + q_3 k \,, \tag{1}$$

which can also be represented as

$$q = [q_0, q_1, q_2, q_3] \,, \tag{2}$$

here

q_0 – is the scalar part of the quaternion

$[q_1, q_2, q_3]$ – is the vector part of the quaternion.

For attitude control the vector is crucial for feedback control and the scalar part is typically normalized to a unit quaternion. A unit quaternion is defined by dividing the quaternion by its norm; and can be seen as

$$\frac{q}{\|q\|} \,. \tag{3}$$

Knowing how to take an inertia vector and transform it into a quaternion is very useful. The first step is to convert the vector into a diagonal matrix. This is done by taking an identity matrix, with rank equal to the length of the vector, and plugging in vector [1] into matrix [1,1] and vector [2] into matrix [2,2] and vector [3] into matrix [3,3]. This creates a diagonal matrix with q_1 in the upper left corner, q_2 in the middle, and q_3 in the bottom right corner, and zeroes everywhere else. Consider a rotational matrix 'R', where

$$\mathrm{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} . \tag{4}$$

The quaternion can be determined by

$$q_0 = \pm \frac{1}{2}\sqrt{1 + trace(R)}\,, \tag{5}$$

here

$$\text{trace(R)} = r_{11} + r_{22} + r_{33}\,,$$

resolving

$$q_1 = \frac{r_{23} + r_{32}}{4q_0} \tag{6}$$

$$q_2 = \frac{r_{31} + r_{13}}{4q_0} \tag{7}$$

$$q_3 = \frac{r_{12} + r_{21}}{4q_0}\,. \tag{8}$$

There are many more operations that could be considered; however, these are some of the fundamentals for using quaternions as a means of reference for attitude. Like matrix multiplication, it is important to remember quaternion multiplication is a non-commutative process.

Appendix D: Max Torque for Actuators Code

The following code was used to determine the limits of torque for the magnetorquer rods.

```
%%%Calculating MAX Torque of Actuators

Tf = 1000;% Simulation Time seconds
Ts = 1;% Step Time in seconds
direction=[-1;-1;-1;1];
qn=[direction/norm(direction)]; %normalized initial quaternion
for i = 1:Tf
 T(i) = i-1; %Creating discrete instances of time
%%TLE Provided by https://www.celestrak.com/NORAD/elements/cubesat.txt %%%
longstr1='1 28895U 05043F   19152.89631413  .00000082  00000-0  24487-4 0  9996';
longstr2='2 28895  97.9567 296.2056 0017789  48.4291 311.8433 14.63664148724792';

satrec=twoline2rv(84,longstr1,longstr2,'m','e'); % Initializing Orbit parameters
%whichconst = 84; % set grav constants to wgs 84 conventions
[ECEF_Init,r,v] = sgp4(satrec,T(i)) % Position Vector for each Time Instant in ECEF Frame
w.r.t ECI Frame
LLA_Init = ecef2lla(r); % Transform to Geodetic Frame w.r.t ECI Frame
Latitude = LLA_Init(1); % Compute Real-Time Latitude
Longitude = LLA_Init(2); % Compute Real-Time Longitude
Altitude = LLA_Init(3)/-10000; % Compute Real-Time Altitude in Km

%%% Compute Magnetic Field vector using World Magnetic Model
[b_0_t] = wrldmagm(Altitude, Latitude, Longitude, decyear(2015,12,31))*1E-9; % Convert nT
to T;
A_q = quat2dcm(qn'); % Attitude Matrix
b_t = (A_q)*b_0_t; % Magnetic Field in Satellite Body Frame
S_b_t = [0, b_t(3), -b_t(2); -b_t(3), 0, b_t(1); b_t(2), -b_t(1), 0]; % cross-product Matrix
Transpose = S_b_t';
Abs = (norm(b_0_t)^2)^(-1);
m = Abs*Transpose*.0000059; %Testing different constants till dipole moment is .2
% m_coils(:, i) = m; % Residual Dipole Moment in A-m^2
Torque=S_b_t*m; %Calculating Torque Matrix
TorqueMax_zaxis=abs(Torque(1,2)); %declaring max torque in N-m
TorqueMax_yaxis=abs(Torque(3,1)); %declaring max torque in N-m
TorqueMax_xaxis=abs(Torque(2,3)); %declaring max torque in N-m
%Finally, creating an array of Torque values for the duration of the simulation
Torque_zaxis(:,i)=TorqueMax_zaxis; %This will limit un(4)
Torque_yaxis(:,i)=TorqueMax_yaxis; %This will limit un(3)
Torque_xaxis(:,i)=TorqueMax_xaxis; %This will limit un(2)
end
```