DEVELOPMENT OF A FUZZY SLIDING MODE CONTROLLER FOR SATELLITE ATTITUDE CONTROL

Kevin J. Walchko Dr. Paul C. Mason

GSRP Final Report

FALL 2000

Mechanical Engineering
University of Florida
Gainesville, FL 32611

Acknowledgments

I would like to thank Tom Stengle for being my NASA mentor. He has provided me with guidance and support. In addition, I would like to thank Dr. Landis Markley and Dr. Russell Carpenter for serving as my NASA technical advisors. They provided me with a significant amount of their technical expertise and practical knowledge of the field.

Also I would like to thank Dr. Gerlad A. Soffen who heads the GSRP for Goddard. Through his diligent efforts, I was able to be exposed to some of the challenges that face NASA and Goddard engineers and provide some possible solutions to these problems in the area of attitude controls.

Finally I would like to thank my advisor Dr. Paul Mason who introduced me to satellite control research. He has provided great insight for me into the area of controls and space technology. He has also provided his time to look over my work and provide an objective opinion.

Executive Summary

Introduction

Small spacecraft, which are very cost effective, are an essential tool for achieving the objectives in Goddard's mission statement. In general, attitude control and determination plays a crucial role in the success of any space mission. Without robust attitude control, a spacecraft can not be pointed with any reasonable accuracy nor reliability. Due to this importance, the Flight Dynamics Analysis Branch was formed to provide the various support functions to the flight projects including navigation, mission design, attitude control and determination.

Due to the inherent non-linearities, uncertainties, saturation constraints, and physical limitations, traditional attitude control schemes can not always provide the required pointing precision. Therefore, the objective of this Graduate Student Research Program (GSRP) work is to developed new more robust attitude control schemes provide higher levels of accuracy in the presence of uncertainty and non-linearities. In addition, these schemes should be modular and can be ported to similar spacecraft with minimal effort.

One of the major contributions of this work is the development and simulation of a generic fuzzy sliding mode controller for NASA satellites. Due to its generic nature, this control scheme can be adapted to a variety of existing and future satellites with minimal effort. The hybrid structure of the controller takes advantage of classical sliding mode control logic while maintaining a significant degree of robustness, performance and portability. The fuzzy sliding mode controller was compared to conventional sliding mode, fuzzy logic MIMO control, and a PD controller. The fuzzy sliding mode (all of which were formulated as part of the first year of the GSRP) provided much better performance in pointing and tracking than the other controllers. Furthermore, the fuzzy sliding mode controller out performed the other control schemes in the presence of noise and disturbance rejection. Finally, the fuzzy sliding mode controller was more portable than the other control schemes. The results of this investigation are documented in this report

Conclusions

A simulation study, which examined the performance of the four attitude control schemes mention above was performed using three different spacecraft models. The conventional sliding mode control, which was formulated in this work, was able to provide the appropriate control in the presence of noise and disturbance. However, the fuzzy logic / sliding mode hybrid was able to provide better control response than the conventional sliding mode, but at the price of a higher fuel consumption. The fuzzy logic controller and the tuned PD controller performed on an even footing, with the fuzzy logic controller achieving a slight advantage over the PD controller. However, without additional tuning with respect to other satellites neither one of these two controllers was able to properly control anything other than the SAMPEX satellite.

A drawback of the fuzzy sliding mode is it's more computationally expensive than the classical sliding mode controller. However, this can easily be overcome by partitioning the input and output

space of the fuzzy controller into a look up table (i.e. similar to gain scheduling). This method is common in reducing the computational expense of fuzzy logic.

Overall, fuzzy logic did not reach the level of performance expected in the outset of this project. It was believed that fuzzy logic, which is model independent, would provide a much better performance with much less design time. However, this was not the case. Significant design time still went into designing the fuzzy logic controller and the fuzzy sliding mode controller, with only slightly better performance. In fact, fuzzy sliding mode had to be utilized due to the fact that fuzzy logic alone did not accomplish the desired results. In conclusion, sliding mode with or without fuzzy logic, was capable of performing generic control across very different spacecraft.

Finally, some level of adaptation would need to be incorporated into the controller to estimate the inertia matrix in order to truly provide generic control. Originally this was going to be accomplished by reinforcement learning, but current adaptive control theory or the use of an extended kalman filter could also provide it.

Abstract

The initial objective of this Graduate Student Research Program (GSRP) work is to develop an intelligent fuzzy attitude control strategy for a spacecraft. This new approach ensures the required performance in the presence of disturbance, uncertainty and various non-linearities. Furthermore, the proposed attitude control strategy is developed based on a modular formulation allowing for portability such that it can be applied to other spacecraft with minimal effort. Simulation studies are used to illustrate the merits of the control scheme. The SAMPEX (solar, anomalous, magnetospheric particle explorer) was the first virtual test bed for the proposed work. The proposed algorithms were also applied to the MAPS and SMART mission without redesign to illustrate portability. By utilizing the developed scheme NASA can reduce the cost and time for designing control systems while increasing the performance and reliability of current and future missions.

General Management Plan

The major objectives of the first year of this NASA GSRP project were accomplished and surpassed. This project followed the time line shown below in Table 1. Table 2 shows current activities associated with the project. The overall objective of the first year was the development and comparison of several robust attitude controllers for a small spacecraft. These controllers must take into account practical constraints and limitations while being modular.

Because of the technical and practical guidance that was provided by the GSRP program, Kevin Walchko has completed his masters and is now working on his Ph.D. His current Ph.D. work is an extension of the first year GSRP results. Some of the areas being investigated are shown in Table 2. Both Dr. Mason and Kevin Walchko are investigating the possibility of extending some of the first year results to formation control. This work can be the foundation of advanced autonomous formation control methodologies for space and land based missions.

Table 0-1: Year 1 Activities and Milestones

	Aug	Sept	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	June	July
Modelling Sampex	■	■										
Controller Development												
QFB - Standard		■	■	■								
Fuzzy Hybrid - AI			■	■	■							
Stability Analysis		■	■	■	■							
Disturbance Rejection				■	■	■						
Saturation Characteristics					■	■	■					
Reinforcement Adaptation							■	■	■			
Nonlinear Controllers												
Sliding Mode								■	■	■		
Fuzzy Sliding Mode									■	■	■	
Case Study / Comparison										■	■	■

Table 0-2: Ongoing Project Activities

	Aug	Sept	Oct	Nov	Dec
Visual Simulation Tool					
Fuzzy Slide Refinement					
Adaptation Mechanism					
Formation Attitude Control					

Copyright

I. Why a Generic Controller

Introduction

Small spacecraft, which are very cost effective, are an essential tool for achieving some of the objectives in Goddard's mission statement. In general, attitude control and determination plays a crucial role in the success of any space mission. Without robust attitude control, a spacecraft can not be pointed with any reasonable accuracy (or reliability). Due to the importance of attitude and other mission operations (mission design, orbit determination, navigation, etc), the Flight Dynamics Analysis Branch was formed to provide the various support functions to the flight projects. To support the next generation spacecraft missions, the Flight Dynamics branch has put a significant amount of resources in to robust, autonomous attitude control and determination.

Due to the inherent non-linearities, uncertainties, saturation constraints, and physical limitations, traditional attitude control schemes can not always provide the required pointing precision for the next generation missions. Therefore, the objective of this Graduate Student Research Program (GSRP) work is to developed new robust schemes that can provide higher levels of accuracy in the presence of uncertainty and non-linearities. In addition, these schemes are formulated based on a modular structure, and can be ported to similar spacecraft with minimal effort.

This report summarizes the development and simulation of several robust attitude controllers including a generic fuzzy sliding mode attitude controller. Due to its generic nature, the developed fuzzy sliding mode scheme outperformed the other robust controllers. It can also be adapted to a variety of existing and future satellites with minimal effort. The fuzzy sliding mode scheme takes advantage of classical sliding mode control and fuzzy logic, which provides robustness, performance and maintains a significant degree of portability due to its hybrid structure. To illustrate the merits of the fuzzy sliding mode controller is compared to conventional sliding mode, fuzzy logic MIMO control, and a PD controller in the presence of noise and disturbance.

In order to provide the reader with a better grasp of the merits of this work, a brief review of some of the controller development is furnished. This background/overview contains the relevant works and is not supposed to be encompassing.

Background

Conventional Attitude Control Schemes

Having the ability to correct large slewing errors is a motivating factor for many works in attitude control. This capability is essential for spacecraft that are spin stabilized during deployment, which

could produce large initial attitude errors. For example, if deployment occurs from the space shuttle, then the reaction jets must be turned off until the spacecraft reaches a minimum safe distance of 200 ft. Since the deployment rate is approximately one ft/sec, then no corrections can occur for 200 sec. An imperfect deployment (as little as .5 deg/sec) could cause the spacecraft to drift away from the desired attitude where a large angle correction (up to 180 degrees) would be needed to reorient to the proper attitude. To address this problem, Wie and Barba [1] developed several computationally efficient control schemes for large angle maneuvers. Many of these stabilizing schemes utilize quaternion and angular velocity feedback. The use of quaternion representation allows for more realistic, large angle maneuver control schemes. These schemes are formulated based on Liapunov analysis, which produces a range of positive stable gains for that control law. Thus in order to meet desired performance, engineers must iterate through a significant number of gain combinations to obtain the desired response in a clean simulation. However, even when a satisfactory response is finally obtained, there are no guarantees how the satellite will behave in the presence of disturbances, noise, or uncertainties.

Crassidis and Markley [2] developed a model based nonlinear predictive control method for spacecraft, which allowed for large-angle maneuvers. This method utilized a predictive filter to determination the trajectory and control effort one-time step ahead. This predictive control scheme determined the torque input required to make the predicted trajectories match the desired trajectories by minimizing the norm-squared error between the two. This method was robust against model error, and was tested in simulation on the Microwave Anisotropy Probe (MAP). Although this control scheme outperformed than other traditional controllers presented in the paper, the design method was very complicated and time consuming. Thus, the design method would require an expert with in-depth knowledge to redesign a control system for each new satellite.

Fuzzy Logic

Intelligent controllers, such as fuzzy logic, can account for uncertainty, reduce complexity, and have been shown to out perform conventional schemes. It can even be used in place binary logic in software and hardware. “A lower cost microcontroller can be used since fewer lines of logic are used, complex mathematical models do not have to be developed, and less computationally expensive programs are generated.” [3] Even though this statement has been proved, few American companies employ the power of fuzzy logic. Zadeh [4], who was the first to develop the basic concepts associated with fuzzy theory, sees the lack of acceptance of fuzzy logic in America as a “lack of understanding of what fuzzy logic is and what it has to offer.” While Sibbigtroth [5] sees the problem as “US engineers typically take the position that any control methodology without precise mathematical models is unworthy of serious consideration.” Even with this lack of commitment to fuzzy logic exist, many chip manufacturers (ie. Motorola 68HC12, Adaptive Logic AL220, SGS-Thomson, etc.) are producing various microcontrollers specifically designed to utilize fuzzy logic as the primary control scheme. These microcontrollers greatly reduce the computation of fuzzy logic by using a hardware solution rather than a software one. This means that even very complicated fuzzy controllers can run at extremely fast hardware speed, thus enabling real-time control of non-linear systems with significant model error [6].

Unlike classical controls, whose design is model dependent and determines exact control actions, fuzzy logic controllers can be developed without the use of analytical models. In addition, it can

accommodate a large degree of uncertainty. Walchko et al [7] performed numerous experiments with a hybrid fuzzy PID controller on linear systems and showed that in case where the inputs and outputs had the same ranges, the hybrid fuzzy controller outperformed its' classical counter part. These bench mark experiments (DC servomotor, the hydraulic piston, the pneumatic linear actuator, and a ball screw) illustrate the flexibility (generic trait) of a single controller to successfully control different systems with few to no modifications, thus showing fuzzy logic to be model independent. *Although other authors [8,9,10,11,12,13,14] have developed hybrid fuzzy PD and PID controllers, none have investigated their generic capabilities. In addition, few researchers have applied fuzzy logic to satellite control problem.*

One of the initial works which applied fuzzy to satellites was by Woodard. Woodard [15] developed a fuzzy controller for the Fast Auroral Snapshot Explorer (FAST) and compared it to the traditional controller (AGSS). FAST was launched in August 1996 utilizing the Attitude Ground Support System (AGSS). AGSS is a group of algorithms developed in the 1970's for the Dynamics Explorer mission. Woodard's fuzzy controller proved to be inferior to the traditional AGSS. During pointing maneuvers the fuzzy controller took 15% longer to point in the desired direction and had a larger range of errors during pointing. However Woodard did point out that the fuzzy logic controller was mathematically simpler and more flexible, but not as accurate as the traditional method. "The performance of the fuzzy logic controller was slightly less desirable than that of the AGSS. This reinforces the general notion that performance with fuzzy logic controllers is sacrificed somewhat." [10] This is an unfair notion of fuzzy logic's performance capabilities. Fuzzy logic does have the capability to achieve or surpass the performance level of other traditional control schemes, as shown in this work.

In another work, Steyn [16] performed a comparison study between an adaptive MIMO LQR and a fuzzy logic controller for small satellites in a low Earth orbit. The fuzzy logic controller was capable of producing slightly better performance than the LQR in controlling the non-linear satellite dynamics. Steyn's simulation included sensor noise and a cyclic disturbance. Steyn [17] further describes the design of the fuzzy logic controller, which utilizes a 3-axis magneto-torquing to maintain its attitude. Fuzzy logic was capable of "maximizing the desired influence on the controlled axis and minimize the undesired disturbances to the other axes. It was found fuzzy logic can achieve these goals in a computationally efficient way."

Fuzzy controllers, which have already been applied to satellites, have performed well against other well known controllers. Fuzzy logic's strength is its' model independence and intuitive nature in designing the rule base. It is hoped these factors can help in the design of the generic controller presented here.

Intelligent Adaptive Mechanisms

Controllers that achieve the desired response in the presence of variations in the system dynamics or the environment by altering control parameters based on some adaptation mechanism are referred to as adaptive schemes. In most cases, these schemes do not alter the basic structure of the controller. There are many different conventional and intelligent adaptation mechanisms that can alter a controller's parameters and structure to adapt to its new environment or dynamics [18,19]. Some of these techniques are described below.

Guo and Huang [20] used genetic algorithms as a tool to alter membership functions of a fuzzy controller for the motion control of an autonomous underwater vehicle. Their method of adaptation was capable of tuning a fuzzy controller, with very poor initial performance, to produce acceptable results. Genetic algorithms are efficient at the adaptation process, due to their ability to get out of local minimums through the use of mutations in the gene pool. However, due to the complexity of genetic algorithms (many generations must be explored in order to produce the desirable offspring) a powerful processor dedicated to solving the adaptive process is required. Thus for satellites, in the era of faster, cheaper, smarter, this is not a useful method.

Recently, Buijtenen et al [21] developed an adaptive fuzzy logic PD controller. This method utilizes a critic that predicts the future system performance and a stochastic exploration module to explore the space of possible actions. The actual adaptation process is powered by reinforcement learning scheme. Reinforcement Learning (RL) is a family of biologically inspired algorithms that indirectly evaluates a controllers action and rewards desired outcomes and punishes undesirable outcomes. However, this method was only applied to one attitude axis (roll, pitch, or yaw) while the methods developed in this work provide full authority for all axes. Since the attitude dynamics of a satellite are coupled and non-linear, changing one element of the attitude could have a dramatic effect on the other values. In addition, their method relies on a critic accurately predicting several steps into the future. This is a critical component of the RL, since the critic is used in the reinforcement process. Thus if the critic is inaccurate in predicting the future, this process has no hope of properly adapting.

Due to the potential of fuzzy logic some researchers have utilized it as an adaptation mechanism. For example, the PID controllers can be altered such that their gains changed adaptively by fuzzy logic. Visioli [22], utilized a fuzzy logic system based on the error and derivative of error to tune the PID controller. The values of the proportional, integral, and derivative gain are determined from the well-known Ziegler-Nichols formula. This method is robust against parameter variation and only requires a small computational effort. The author however only implements the controller on simple linear systems.

Many of the current fuzzy techniques are very application dependent. For example Erbatur et al [23] designed a fuzzy adaptive sliding mode controller, where the gain matrix K was adjusted according to fuzzy adaptive rules. Their desire was to reduce chatter in the positioning of a serial linked robot arm. The method proved to work nicely for the simple arm chosen for the study. However, for the adaptive mechanism to work, the initial matrix gain K and adaptive constants must be accurate to provide reasonable results. Thus these initial values were determined through a trial and error process.

With respect to intelligent adaptation techniques, the two most promising adaptive methods appear to be genetic algorithms and RL, but genetic algorithms are too computational to implement. RL seems more promising, since the satellite would itself learn through trial and error. The couple constants present in RL are intuitive, thus easy to pick depending on the desired learning rate. Other methods seem to heavily depend on the starting point of the optimization, and can get caught at local minimums.

Sliding Mode

Unlike linear techniques, nonlinear controllers can accommodate nonlinear dynamics and can provide a higher level of robustness and performance. Sliding mode control, a nonlinear controller, has excellent stability, robustness, and disturbance rejection characteristics. It is categorized as a variable structure control system, which has been studied in the Soviet Union for many years. The use of sliding mode control is not new to satellite attitude control. Lo and Chen [24] designed a sliding mode controller scheme which avoids the inverse of the inertia matrix, and smooths the control effort of the controller. Such a strategy provides for a more efficient use of the fuel.

In another work, Coleman and Godbole [25] conducted performance trade study between fuzzy logic, PID, and sliding mode control. The controllers were tuned for and tested on three similar linear plants. In all three cases, the sliding mode controller out-performed the fuzzy logic controller. This is a typical result, where sliding mode tends to provide a superior model based performance compared to fuzzy logic, assuming the model is very accurate.

Emphasis / Direction of this Research

Recently, Mason and Walchko [27,28] and others have successfully applied fuzzy logic to the satellite attitude control problem. Given the capability to reduce the design and tuning time of current control schemes, NASA could reduce the cost of developing and maintaining a satellites by implementing intelligent sliding mode controllers which are capable of adapting to different satellites. Sliding mode has been shown to provide superior performance to that of fuzzy logic, but fuzzy logic is model independent and robust in the presents of noise and disturbances. The combining of these two technologies could greatly benefit NASA’s ability to control satellites.

Originally the hope of this work was to design an adaptive fuzzy logic controller that would learn how to adapt to different satellites. However, after talking to NASA engineers at Goddard, there seemed to be great skepticism in anything more complicated than a PD controller. Thus we moved away from the more radical idea, and embraced sliding mode control. Sliding mode is capable of handling systems with modeling inaccuracies, and best of all, stability of the controller can be shown. It is hoped that a hybrid controller composed of the well understood sliding mode architecture and fuzzy logic will be better received.

The controllers developed in this work will be implemented on several systems. Specifically, this work uses the SAMPEX, MAP, and SMART spacecraft to illustrate the robustness of the controller.

Outline of Report

This report is laid out in the following manner. Section 2 presence the theory pertaining to fuzzy logic, reinforcement learning, and sliding mode. Section 3 describes the satellites that the controller has been applied too and a brief overview of attitude dynamics. Section 4 is the design of a proportional derivative (PD) controller, a sliding mode controller, fuzzy sliding mode controller and the multi-input/multi-output (MIMO) fuzzy controller. Section 5 is the results of various simula-

tions performed on the SAMPEX, MAP and SMART satellites. Section 6 contains conclusions, and recommendations.

II. Theory

This section lays the basic foundation for the formulation of the controller presented in this work. First a brief overview of fuzzy logic is provided with a more detailed discussion of fuzzy logic given in Appendix D. Next, the hybrid structure, which is a composite of classical and fuzzy logic, is described. Finally, in an effort to automate the control parameter optimization process in an on-line manner for fuzzy logic, reinforcement learning as an adaptation mechanism is introduced and developed.

Fuzzy Logic

Fuzzy logic was conceived by Lotfi Zadeh and brought to the attention of the world in his paper "Fuzzy Set" in 1965. At that time, fuzzy logic was a radical deviation from classical logic, and initially received little interest and attention. However, fuzzy logic is slowly gaining more and more credit in the United States as a legitimate method for controls. The motivation behind the adoption of fuzzy logic to controls applications are: model independence, use of expert knowledge to control systems, robustness to noise/disturbances, capable of controlling nonlinear systems, etc.

A graphical depiction of the entire fuzzy logic control process is shown in Figure 2-1. The n inputs on the left are fuzzified by MFs. Then each of the fuzzified values is evaluated by the rules that produce m outputs.

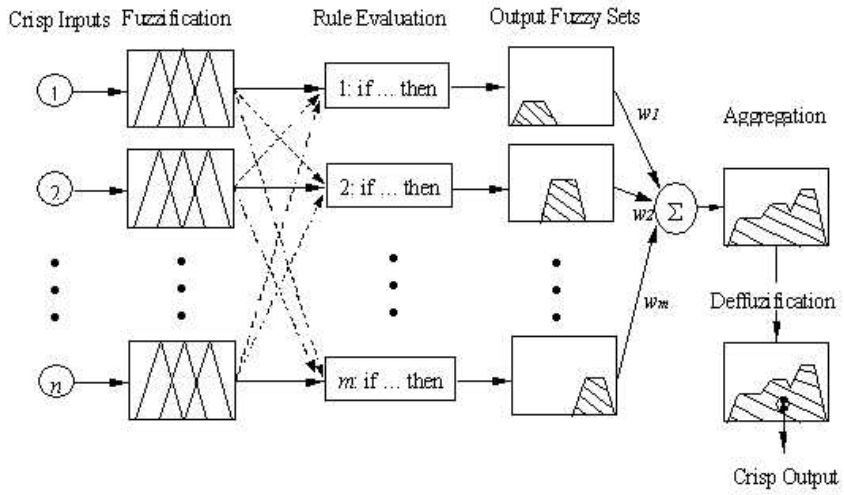


Figure 2-1. Fuzzy logic from input to output. [29]

Each of these m outputs are combined (there are various methods) and the defuzzification of the resulting area is found. Now the whole process of fuzzification, rule application, and defuzzification can be put together.

Fuzzy Stability

Stability analysis is still one of the major obstacles opposing the wide spread adoption of fuzzy logic. Due to recent efforts in the field, there are many generally accepted methods of determining whether a fuzzy controller is stable or unstable. One such method described by Petroff et al [30] utilized a linearized representation of the fuzzy control surface over small areas and then determine whether the controller is stable based on energy methods. However, fuzzy logic control surfaces tend to be very complex and analyzing dense grids over large areas is not always practical.

Another method presented by Sio and Lee [31] utilized the passivity theorem, which allow the determination of a stability region of the effective parameters to be determined. Once this region is defined, along with some sufficient conditions for a stable fuzzy controller, the rule base can easily be defined to produce a stable controller. This method only applies to systems that are passive, which is only a small subset of non-linear systems.

A Liapunov stability analysis is used to demonstrate stability for the proposed work in the context of a PD structure. The down side of this method is that it is not always applicable to fuzzy schemes which are unstructured or have a drastically different structure. This hybrid structure is chosen because of it's generality and intuitiveness. The next section provides a brief description of the hybrid PD structure.

Hybrid Structure

The hybrid controller structure is based upon a classical architecture such that it can be inserted into any error-based control system. A graphical description of this hybrid algorithm is provide in Figure 2-2. There are three standard inputs into the fuzzy inference system: error, derivative and integral. These inputs can be used to compute the fuzzy gains or the control effort directly. The fuzzy gains are time varying and provide a higher level of robustness and control. The values of these gains are dependent on the input and the fuzzy rules, which are based on the users linguistic knowledge of the system. This allows for the control of nonlinear/time varying systems without exact analytical descriptions of these systems. This robust/flexible characteristic allows for the control of a variety of systems with little or no modification of the controller. In this work, this is referred to as the generic characteristic.

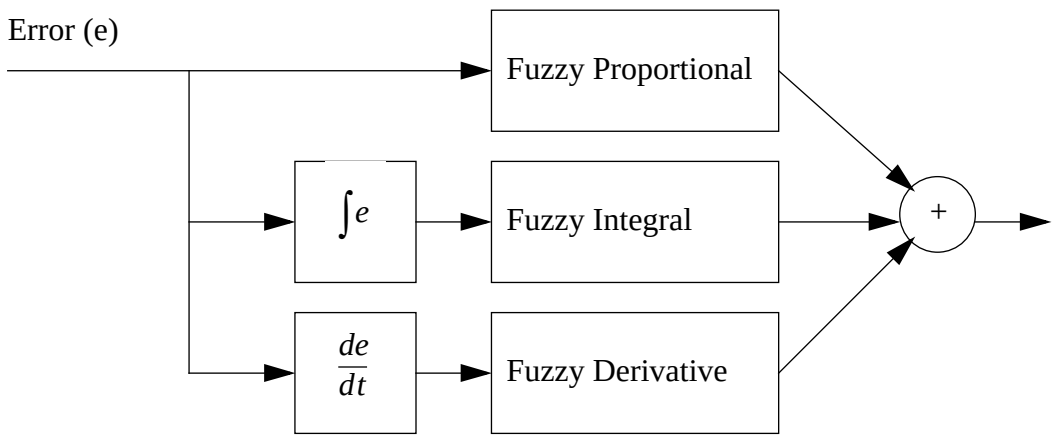


Figure 2-2. The Hybrid structure of the fuzzy controller.

Robust Characteristics

Due to the structure of the current hybrid scheme, a generic characteristic is embedded into the hybrid algorithm such that it can be integrated into a large number of systems to enhance performance without a significant amount of effort. A robust controller simplifies the design of new control application while still maintaining performance. This characteristic introduces a degree of "plug and play" to controller design process. The paragraph below illustrates this attribute for a SISO system. Later, further work will show the robust properties of the controller as they pertain to MIMO satellites.

Given a second order system with a standard PID controller, the closed-loop transfer function is

$$TF = \frac{(K_D s^2 + K_P s + K_I)\omega_n^2}{s^3 + (2\varsigma\omega_n + K_D\omega_n^2)s^2 + (1 + K_D)\omega_n^2 s + K_I\omega_n^2} = \frac{N(s)}{D(s)} \tag{2.1}$$

Replacing ω and ς with $\tilde{\omega}$ and $\tilde{\varsigma}$. Where $\tilde{\omega} = (1 + \Delta)\omega$, $\tilde{\varsigma} = (1 + \beta)\varsigma$, and Δ and β are uncertainties in the system parameter from the nominal or modelled values. The new characteristic equation is

$$D(s) = s^3 + (2\tilde{\varsigma}\tilde{\omega}_n + K_D\tilde{\omega}_n^2)s^2 + (1 + K_D)\tilde{\omega}_n^2 s + K_I\tilde{\omega}_n^2 = 0 \tag{2.2}$$

In a fuzzy PID structure, the control effort can be written as

$$u_c = fuzzy_P(e) + fuzzy_D(\dot{e}) + fuzzy_I(\int e) \approx K_P(1+\Delta_e)e + K_I(1+\Delta_I)\int e + K_D(1+\Delta_D)\dot{e} \quad (2.3)$$

where Δ_e, Δ_I ,and Δ_D are functions of the universe of discourse. The fuzzy universe of discourse can be adjusted to account for variations in the model. This conclusion can be extended to state that the fuzzy PID is a generalized controller (generic) like its classical controls counter part. Although there may be variations, systems having similar levels of controllability and dominant dynamics will yield similar controlled responses. Therefore, this hybrid algorithm can be classified as a generic controller.

Reinforcement Learning

Although fuzzy logic is capable of generic (i.e. System independent) control, in order to achieve better performance a method for fine tuning or optimizing the fuzzy controller's performance was needed. The technique chosen to provide this tuning is reinforcement learning. An introduction to reinforcement learning follows.

Reinforcement Learning (RL) according to Sutton and Barto [32] is learning based on interaction with the environment. Basically, an agent (satellite in this case) tries to map situations to actions in order to maximize a numerical reward signal (which is similar to minimizing a cost function). The agent is not told what actions to take, but must learn (through trial and error) which actions result in the greatest rewards. An action taken does not immediately result in the desired response, thus rewards are delayed in an effort to determine which actions resulted in what out come. Unfortunately for the agent to learn the optimal state-action pairs, it must pass through all possible states. Thus, RL is heavily dependant on the agent exploring its environment before committing to a solution for all time.

For example, a man leaves a bar after drinking heavily. He accelerates to twice the speed limit. Ten minutes later, he approaches a hair pin turn, and turns the wheel sharply in an effort to stay on the road. Unfortunately, due to the speed of the car and his poor hand eye coordination (from drinking) the car flies off the road. Now, if rewards were immediately given in this situation instead of delayed, what would be penalized? Only sharply turning the wheel action would be penalized as what not to do, rather than also penalizing the agent for drinking and speeding too. There are various methods for determining this delayed reward.

A diagram of the agent-environment is shown in Figure 2-3. RL contains four main sub-elements in addition to the agent and the environment: a policy, a reward function, a value function, and (optionally) a model of the environment. Policy determines how an agent learns. It is the method by which perceived states of the environment are mapped to actions. Policies may be functions, look up tables, stochastic, or computational search algorithms. A common policy is (ε)-greedy, where an agent in a given state picks the action with the highest possible reward. However, ε percent of the time the agent will pick a random action instead of picking the action with the highest possible reward. This policy enables the agent to explore all possible actions to see if they lead to higher rewards, thus preventing the adaptation mechanism from getting caught in local minimums. However this also requires the agent to spend lots of time exploring all possible states.

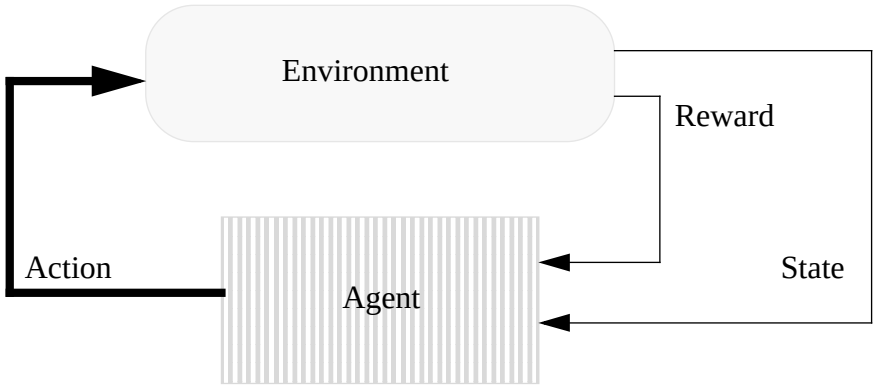


Figure 2-3. Q-Learning block diagram which shows how the agent takes the inputs of reward and states to produce an action.

Reward function maps each perceived state (s) or state-action pair (s,a) to a single reward value (r). As previously stated, the agent's main objective is to maximize the reward received. The reward function is the basis for altering policy.

The value function (Q) indicates how much reward an agent gets by taking a certain action in a certain state. This function is typically initialized randomly to small values then updated by the reward function as an agent explores its environment utilizing its policy.

Q-Learning

Q-leaning is one of many algorithms in RL. It is a method for approximating the optimal state-action pair. The algorithm is shown below in List 2-1.

List 2-1. Q-Learning

```
Initialize Q(s,a) to small random values
Repeat for each episode:
   Initialize s
   Repeat for each step in an episode:
      Choose a form s using policy derived from Q (i.e. e-greedy)
      Take action a, observe r, s'
      Q(s, a) ← Q(s, a) + α[r + γmax_a' Q(s', a') – Q(s, a)]
      s ← s'
   until s is terminal
```

III. System Models and Dynamics

In this section, a brief description of the three satellites used in this study, SAMPEX SMART and MAP, is given. Their background and mission are briefly stated for familiarity. In addition, there is a concise review of satellite dynamics.

MAP Satellite

The MAP (Microwave Anisotropy Probe) satellite's primary mission is to probe conditions in the early universe by measuring the properties of the cosmic microwave background radiation over the full sky. MAP will accomplish this by using passively cooled differential microwave radiometers with dual Gregorian 1.4 x 1.6 meter primary reflectors. Questions that MAP could answer: How old is the universe? How fast is the Universe expanding? Is the universe infinite? Is there a cosmological constant? What is the density of ordinary ("baryonic") matter? When did the first stars form? What is the origin of structure in the universe?

SAMPEX Satellite

SAMPEX (Solar, Anomalous, and Magnetospheric Particle Explorer) [33,34] is a product of the SMEX (Small Explorer Program). This program realizes the advantages of small, quick turnaround projects. The SAMPEX satellite is designed and equipped to study the energy, composition, and charge states of particles from the explosions of supernovas, solar flares, and from the depths of interstellar space. Closer to home, SAMPEX monitors the earth's middle atmosphere as magnetospheric particle populations occasionally plunge into it.

SMART Satellite

The SMART (Scientific Microsatellite for Advanced Research and Technology) [35] is a project funded by the Italian Space Agency to design and develop a multi-mission microsatellite for remote sensing applications in Sun-synchronous orbits. The microsatellite consists of main bus (40 kg) supplied with 64W (average) power from a solar array.

Satellite Summary

The satellites used in this study are very different in size, however, their model structure is the same. The table below lists their inertia matrices, which reveals three basic size classifications:

small, medium, and large. Utilizing these three different sizes, the generic capabilities of the proposed controllers can be measured.

Table 3-1: Satellite Intertia Matrices

Satellite	**Inertia Matrix (kg-m^2)**
MAP	$\begin{bmatrix} 399 & -2.81 & -1.31 \\ -2.81 & 377 & 2.54 \\ -1.31 & 2.54 & 377 \end{bmatrix}$
SAMPEX	$\begin{bmatrix} 15.52 & 0 & 0 \\ 0 & 21.62 & -0.194 \\ 0 & -0.194 & 15.23 \end{bmatrix}$
SMART	$\begin{bmatrix} 0.57 & 4.21\times10^{-4} & -4.86\times10^{-2} \\ 4.21\times10^{-4} & 0.72 & -5.8\times10^{-3} \\ -4.86\times10^{-2} & -5.8\times10^{-3} & 0.45 \end{bmatrix}$

Spacecraft Attitude Dynamics

In this section, a brief review of the kinematic and dynamic equations of motion for a three-axis stabilized spacecraft is presented [36]. The attitude is assumed to be represented by the quaternion, defined as

$$q = \begin{bmatrix} q_{13} \\ q_4 \end{bmatrix} \tag{3.1}$$

with

$$q_{13} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \hat{n} \cdot \sin(\theta) \tag{3.2}$$

$$q_4 = \cos\left(\frac{\theta}{2}\right) \tag{3.3}$$

where $\hat{n}$ is a unit vector corresponding to the axis of rotation and is the angle of rotation. In Wertz [37] the quaternion kinematic equations of motion are derived by using the spacecraft's angular velocity,

$$\dot{q} = \frac{1}{2}\Omega q \tag{3.4}$$

The dynamic equation of motion, also known as Euler's equation, for a rotating spacecraft is given by

$$\dot{H} = -\omega \times H + u \tag{3.5}$$

where is the total system angular momentum, u is the total external torque (which includes, control torques, aerodynamic drag torques, solar pressure torques, etc.). In addition, the angular velocity from of Euler's equation (given below) could be used.

$$J\dot{\omega} + \omega \times Jw = u \tag{3.6}$$

where J is the inertia matrix of the spacecraft, and u is the total torque.

IV. Controller Designs

This section provides the formulation of the attitude controllers studied in this work. The first controller is the standard quaternion feedback controller. The second is the sliding mode attitude controller. Third is fuzzy logic PD hybrid and the last is a fuzzy sliding mode control hybrid. These algorithms are chosen because of their potential and robustness. This initial work will the basis of future works in robust intelligent attitude control.

Proportional Derivative Control

In modern control theory output feedback control is one of the most practical and used schemes. However, it cannot arbitrarily place all of the poles of the system. If the system is observable and controllable, this method is quite adequate to meet the need of many existing applications. In attitude control, quaternion feedback control is well suited for onboard real-time control due to the computational efficiency of quaternions. This method feeds back both attitude and angular velocity. Therefore allowing for full control authority (pole placement). The quaternion based PD controller defined by Bong Wie [38] is

$$u = -K \cdot q_{error} - C \cdot \omega_{error} \tag{4.1}$$

where q_{error} is the quaternion error and ω_{error} is the angular velocity error in the system. Wie lists four different controllers gains (for the same control structure), where the positive scalar values for K, α, β and c are obtained via simulation (trial and error).

Controller 1: $K = kI \qquad C = diag(c_1, c_2, c_3)$

Controller 2: $K = \dfrac{k}{q_4^3} \qquad C = diag(c_1, c_2, c_3)$

Controller 3: $K = k \cdot sign(q_4)I \qquad C = diag(c_1, c_2, c_3)$

Controller 4: $K = [\alpha J + \beta I]^{-1} \qquad K^{-1}C > 0$

where k is a scalar and I is the identity matrix. In quaternions space, the error is determined by multiplying the inverse of the commanded attitude quaternion by the current attitude quaternion.

$$q_{error} = q_{commanded}^{-1} \cdot q_{current} = q_c^{-1} \cdot q \tag{4.2}$$

$$q_{error} = \begin{bmatrix} q_{4c} & q_{3c} & -q_{2c} & -q_{1c} \\ -q_{3c} & q_{4c} & q_{1c} & -q_{2c} \\ q_{2c} & -q_{1c} & q_{4c} & -q_{3c} \\ q_{1c} & q_{2c} & q_{3c} & q_{4c} \end{bmatrix} \cdot \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{4.3}$$

Once the quaternion error and the angular velocity error terms are determined, the control effort is determined by equation 4.1. In the current formulation, the control efforts are decoupled and K, C can be written as diagonal matrices. In addition, the gains of this controller are constant and do not account for uncertainties or variations in the system dynamics. The next controller investigated can account for uncertainties and is formulated for nonlinear systems.

Sliding Mode Control

Sliding Mode Control, which is a variable structure control scheme, is more than a promising technique in the field of non-linear control. It permits the realization of very robust and simple regulators, with appealing characteristics. It utilizes model and uncertainty information (bounds) to obtain high accuracy robust performance in a variety of applications. In this work, sliding mode control is applied to attitude control. This formulation, which is a full-state feedback technique, utilizes the existing dynamic model and compensates for uncertainty while formulating a control effort that tracks the desired trajectories. The equations of motion for a satellite's attitude (in quaternion space) are given by [38].

$$J\dot{\omega} = \Omega J\omega + f + u \tag{4.4}$$

$$\dot{q} = \frac{1}{2}\Omega q + \frac{1}{2}q_4\omega \tag{4.5}$$

where:

$$\Omega = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \qquad q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

f in the equation 4.4 represents the modelling error and potential disturbances in the system. q is a vector composed only of the three functional elements of a quaternion. In the original controller formulation a second order matrix representation is utilized (two first order equations, n = 2). However, in this work, we can take the state variable x to be the euler angle orientation, and equate this to q (see Appendix A for derivation).

$$x = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} = 2 \cdot \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} \tag{4.6}$$

Thus the sliding surface (s) is given by [39].

$$s = \left(\frac{d}{dt} + \lambda\right)^{n-1} \tilde{x} = \dot{\tilde{x}} + \lambda \tilde{x} = \tilde{\omega} + \lambda \tilde{q} \tag{4.7}$$

where

$$\tilde{x} = x - x_{desired} \tag{4.8}$$

In order to calculate the equivalent control effort ($\hat{u}$), we need to take the derivative of the sliding surface and set it equal to zero.

$$\dot{s} = \dot{\tilde{\omega}} + \lambda \dot{\tilde{q}} \tag{4.9}$$

Now multiplying both sides by the inertia matrix (J) will allow us to substitute in the equations of motion (4.4 and 4.5).

$$J\dot{s} = \hat{J}\dot{\tilde{\omega}} + \lambda \hat{J}\dot{\tilde{q}} = \hat{J}\dot{\omega} + \lambda \hat{J}\dot{q} - \hat{J}\omega_{desired} - \lambda \hat{J}\dot{q}_{desired} \tag{4.10}$$

$$J\dot{s} = \Omega \hat{J}\omega + \hat{f} + \hat{u} + \lambda \hat{J}\dot{q} - \hat{J}\dot{\omega}_{desired} - \lambda \hat{J}\dot{q}_{desired} \tag{4.11}$$

The hat over particular variables denotes that they are estimates. In reality, we may not accurately know the true inertia matrix. Assuming a pointing maneuver, there is no desired angular acceleration or velocity. Therefore,

$$\dot{\omega}_{desired} = \dot{q}_{desired} = 0$$

However for tracking, one or both of these terms would exist, but here the derivation is only for pointing so those terms drop out.

$$J\dot{s} = \Omega \hat{J}\omega + \hat{f} + \hat{u} + \lambda \hat{J}\dot{q} = 0 \tag{4.12}$$

Now solving for the equivalent control effort from equation 4.8 yields.

$$\hat{u} = -\Omega \hat{J}\omega - \hat{f} - \lambda \hat{J}\dot{q} \tag{4.13}$$

where

$$\hat{f} = \alpha \hat{J} s = \alpha J(\tilde{\omega} + \lambda \tilde{q}) \tag{4.14}$$

$$F = \hat{f} - f = \tilde{J} s \tag{4.15}$$

Finally, the sliding mode control effort is given by

$$u = \hat{u} - K \cdot sat(s) \tag{4.16}$$

$$K = F + \eta = \tilde{J} s \tag{4.17}$$

where η is defined as zero in our formulation. However, other applications utilize this as a secondary tuning parameter. If we examine equation 4.13, [26] refers to the first and third terms as feed forward adaptive terms. While the second term has a PD structure. Thus,

$$\alpha \approx k_D \qquad \lambda \approx TD = \frac{k_D}{k_p}$$

The K term in equation 4.17, is defined based on the magnitude of the uncertainly in the model. This switching term provides the appropriate control action to quickly drive the trajectories back onto the sliding surface. Unlike other works, the sign function is replaced by a saturation function. Since real actuators do not have an infinite bandwidth, one can expect to get chatter due the use of the sign function. The saturation function not only smoothness the control response, it also reduces the amount of fuel used by the controller.

Fuzzy Logic Control

Introduction This section furnishes the basic formulation and structure of a SISO fuzzy controller. Next, a description of the method used to extend the SISO fuzzy controller to a MIMO controller is given. Finally, the stability and robustness of the controller are examined.

The fuzzy attitude controller utilizes the same rules and structures as Walchko et al [7] which was used to control various linear systems. The rule base was developed by first designing a conventional PID controller using the root locus method, and formulating the relationship between errors (error, integral of error, and derivative of error) and the control. The first set of rules was constructed to mimic this relationship.

Like the conventional PID, the fuzzy PID hybrid consists of three inputs (error, integral of error, and derivative of error) and one output (control effort). Since this is a multi dimensional fuzzy controller, it is very difficult to display the rule tables. Thus, only some the possible rules will be shown. The rule tables for the controller are shown below in Table 4-1, where NEG, SN, SSN, Z, SSP, SP, POS are negative, small negative, small small negative, zero, small small positive, small positive, positive respectively:

Table 4-1: Fuzzy PID controller output for selected inputs

INT / ERROR	NEG	SN	SSN	Z	SSP	SP	POS
POS	Z	SSP	SP	POS	POS	POS	POS
SP	SSN	Z	SSP	SP	POS	POS	POS
SSP	SN	SSN	Z	SSP	SP	POS	POS
Z	NEG	SN	SSN	Z	SSP	SP	POS
SSN	NEG	NEG	SN	SSN	Z	SSP	SP
SN	NEG	NEG	NEG	SN	SSN	Z	SSP
POS	NEG	NEG	NEG	NEG	SN	SSN	Z

Table 4-2: Fuzzy PID controller output for selected inputs

DER / ERROR	NEG	Z	POS
POS	POS	POS	POS
SP	SP	SP	POS
SSP	SSP	SSP	POS
Z	Z	Z	Z
SSN	NEG	SSN	SSN
SN	NEG	SN	SN
NEG	NEG	NEG	NEG

Table 4-3: Fuzzy PID controller output for selected inputs

INT / DER	NEG	Z	POS
POS	POS	POS	POS

Table 4-3: Fuzzy PID controller output for selected inputs

INT / DER	NEG	Z	POS
SP	SP	POS	POS
SSP	SSP	SP	SP
Z	Z	Z	Z
SSN	SN	SN	SSN
SN	NEG	NEG	SN
NEG	NEG	NEG	NEG

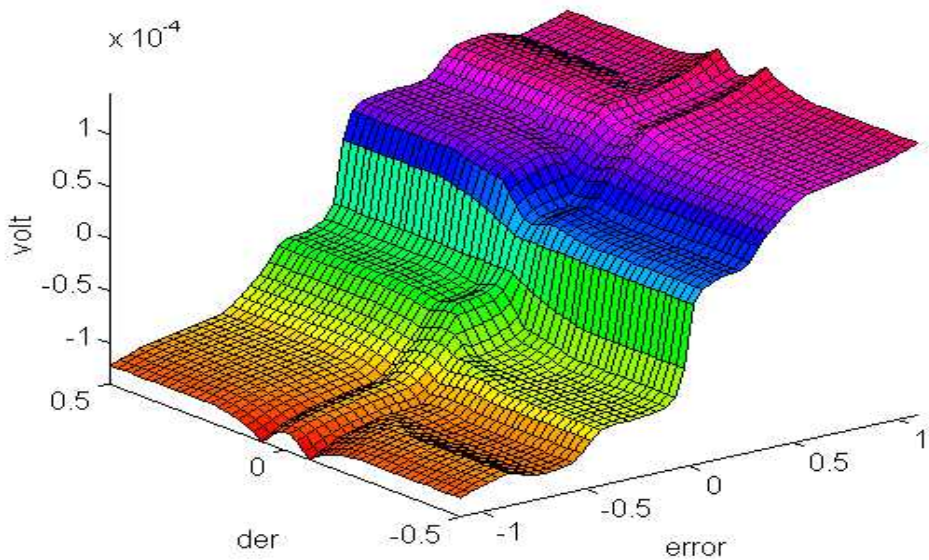


Figure 4-1. Control surface for fuzzy logic controller.

Extension of the SISO Fuzzy Hybrid Scheme to MIMO Applications

The fuzzy PID controller was originally designed for SISO systems. In order to apply it to the satellite problem, a technique to extend the SISO scheme into the multi input realm must be developed. This conversion (or extension) can be accomplished through the manipulation of the error vector, which is used to define the direction and magnitude of the control effort to reduce the error in a minimal amount of time. The motivation and the formalization of this technique are given below.

The error used by the initial fuzzy PD controller is a scalar. However, in the MIMO case, it is a vector. In order to maintain the fuzzy PID structure, the vector is transformed into a scalar repre-

sentation. This scalar representation is used to calculate the MIMO control effort. The transformation is based on the norm of the error vector.

$$|\grave{e}| \Rightarrow Fuzzy(scalar)$$

This scalar representation is used by the fuzzy inference system to determine control magnitude, which is used along with a given direction to create the MIMO control vector. The control effort level must be based on the ratios of the error magnitudes, which are defined by the direction. This direction is the normalized error vector, which account for different levels of possible errors. For example, if the first element was .001 and the third element was 4000, the input error to the fuzzy controller would be dominated by 4000 and the output control effort would be large. This would result in over controlling of the system, or result in instability.

The control effort vector is defined by magnitude and direction. This relationship is used to extend the single control effort to multi output control effort.

$$direction = \frac{e}{|e|}$$

$$u = u_{fuzzy} \cdot direction$$

Since the fuzzy input is always positive, the complexity of the fuzzy inference system is significantly reduced. The number of rules is cut in half, which alleviates the need for all of the negative MF's on the inputs and output (it is important to remember this fact when we discuss stability later). This does not mean that there will never be a negative control effort. Negatives are reintroduced back into the system by the direction vector.

Is This Stable?

The next section addresses the issue of stability and robustness of the controller. The stability will be examined via Liapunov. For SAMPEX, or the standard small spacecraft control problem, the dynamics and control is given by

$$\dot{q} = q \cdot \omega \tag{4.18}$$

$$J\dot{\omega} = \dot{H} = (\omega \cdot x)H \tag{4.19}$$

$$u = -K_p(q_{ref})^T \cdot q - K_D(\omega - 0) \tag{4.20}$$

The following Liapunov function is chosen.

$$V = \frac{1}{2}\omega^T J\omega + \frac{1}{2}(K_p(q - q_{ref})^T(q - q_{ref})) \tag{4.21}$$

Upon simplification

$$\dot{V} = -K_p \omega^T I(q_{ref}) - K_D \omega^T \omega \tag{4.22}$$

V is always positive as long as Kp is positive and V dot is always negative as long as Kp and Kd are positive which were determined earlier to be the criteria needed for stability. This is easy to verify, since it was stated in the beginning that the magnitude of the error vector is sent into the fuzzy controller (always a positive value) and thus always a positive value will emerge from the fuzzy controller. Therefore, the system is stable and robust.

Fuzzy Sliding Mode Control

This section discusses the fuzzy sliding mode controller presented in this research. Since sliding mode and fuzzy logic have already been introduced, this section will be briefly describe the formulation of this new hybrid. The blending of fuzzy logic and sliding mode occurs in Eqn. 4.16 where the K term is replaced by a fuzzy inference system rather than a derived expression for K. The input to the inference system is quaternion error and it's derivative. These inputs provide a accurate description of the state of the system dynamics. Therefore, rules associated with the accuracy and speed of convergence can be formulated. The rule base and control surface are shown below in Table 4-1 and Figure 4-2. Note that Z is zero, S is small, M is medium, and L is large.

Table 4-1: Fuzzy sliding mode rule base

DER / ERROR	Z	S	M	L
Z	Z	S	S	M
S	S	S	S	M
M	M	M	M	M
L	L	L	L	L

The fuzzy part of the fuzzy sliding mode controller is only based on the error and the derivative of the error. Also since the K term of a slinding mode controller is supposed to always be positive (for stability reasons), the fuzzy part always returns a positive result.

Robustness (Generic)

Since the intertia matrix of the satellite is used in the formulation of the controller, the controller will be able to successfully operate across a wide range of satellites. Since the control effort is essentially scaled by the intertal matrix. The fuzzy part of the controller does not depend on size and is generic in its ability to control satellites. The initial performance of the controller to a new satellite is good considering it was not initially tested on the spacecraft. However, as with all off the

rack types, it can be tailored to specific satellites. The performance can be tuned however, by the incorporation of an auto-tuning method which would be able to increase performance.

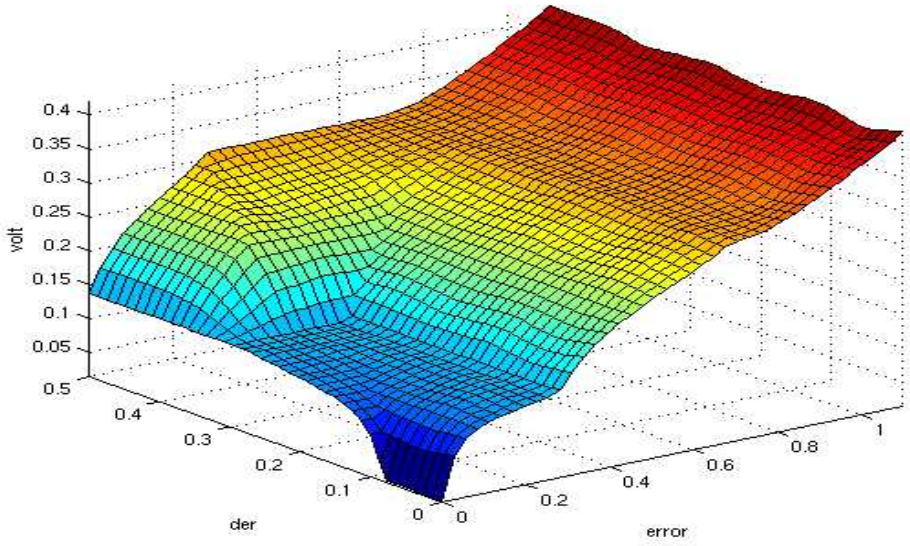


Figure 4-2. Fuzzy sliding mode control surface which replaces the constant gain K in traditional sliding mode control. Note that the out put of the fuzzy logic part is always positive, which is a requirement for K for stability reasons.

V. Simulation Results

This section discusses the results of this work. First will be a short description of things that did not work, and then the results of various simulations of the proposed hybrid controller.

Reinforcement Learning

Even though fuzzy logic has been shown to have generic control capabilities, it still needs adjustment of various parameters to result in better performance. Initially, reinforcement learning (RL) was hoped to provide this capability of auto tuning, and was based on the work done by Benjius [21]. However the equations Benjius used to update his fuzzy logic controller did not improve performance at all, and usually deteriorated performance.

Since we were unable to get RL to work properly with fuzzy logic, a step back to a simpler problem was taken in hopes of discovering what was being done wrong. Q-learning was used to try to find an optimal PD gain mapping at each state. The state was the quaternion error and the action was the PD gain. Intuitively one would expect that at large quaternion errors RL would produce large PD gains, and in states with small quaternion errors RL would produce small PD gains. However this was not the case. RL produced mappings that did not seem logical and produced poor (unstable sometimes) responses. After many weeks, RL was abandoned.

After looking through several books and papers, RL is typically used in different situations than what we attempted to use it in. There are numerous examples of RL being used in path planning, or decision making in robotics. However, all of these examples are in situations where trial and error has the worst outcome of a robot bumping into a wall, not going unstable and becoming millions of dollars worth of space junk. Upon further examination of Benjius's work, he deviates from the standard Q-learning framework. Also, he starts off with a fuzzy controller that performs horribly and is able to control one attitude (ie. either roll, yaw, or pitch) only. His RL scheme results in a fuzzy controller that performs much better than it started off as, but the performance is still poor.

Other Things Tried

Originally, it was hoped that the fuzzy MIMO controller would be able to control the different satellites since it is a model independent controller. A significant amount of time was spent on RL, already discussed above, and also embedding other information into the fuzzy controller in order to get the desired response.

One of the many methods tried was a variable TD term in the fuzzy PD controller. The TD term is basically a weighting factor that decides which error (i.e. position or velocity) is more important.

Various attempts were made to enable the fuzzy controller to generate not only a control effort but also a TD term at each time step. Basically the logic was that the positional error was more important in the initial stages of a step response, but as that error decreased to a certain level, the importance of the velocity error would start to become more important. The results of this work were not fruitful. Eventually it was abandoned.

Simulation

In order to test the generic capabilities of the fuzzy sliding mode controller, three test satellites that had different inertia matrices were chosen. All controllers (i.e. PD, fuzzy logic, sliding mode, and fuzzy sliding mode) were originally designed for the middle sized satellite SAMPEX and then applied to the smaller SMART and larger MAP satellite with no redesign of the controller. The only change was for the sliding mode controllers, the inertia matrix for the different satellites was used. The reason that the inertia matrix for the correct satellite was used, is because it is approximately known before hand. Thus for each section, the approximate inertia matrix used in that simulation is given. Note that the inertia matrix used is not the true inertia matrix of the system, just an approximate uncouple inertia matrix.

Each simulation was conducted in Matlab 5.3 and run for 1000 seconds (16.6 minutes) using a fixed step Runge-Kutta algorithm with the time step set to one second. The satellite was commanded to follow the eiganvalue path of:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \sin\left(\frac{2\pi t}{700}\right) \\ \sin\left(\frac{2\pi t}{700}\right) \\ 0 \end{bmatrix} \tag{5.1}$$

which was then converted into quaternions. The desired velocity was the time derivative of the attitude path and the magnitude of the velocity in either roll or pitch did not exceed .009 rads/sec. This magnitude, incidentally, was well below the maximum input level in the fuzzy logic controller which was .5 rads/sec. Two simulation were conducted, one with a normal gaussian noise added (.001*N[0,1]) into the state feedback of the quaternions (and then renormalized) and a second simulation with a disturbance add to the equations of motion.

Also in Figure 5-1 below, notice that the output of the controller is saturated by a different amount for each satellite. An attempt was made to determine what the maximum control efforts for each satellite where, but only SMART’s were easily available. Both MAPs and SAMPEX were given arbitrary limits.

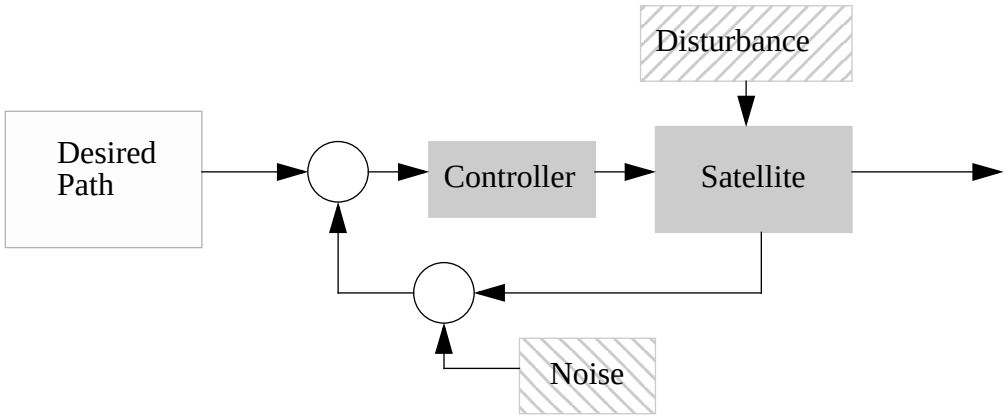


Figure 5-1. Block diagram of signal flow.

In order to quantify the results of all controllers, each simulation will present the amount of fuel consumed during the maneuver and the integral time absolute error (ITAE). This performance measure was used because initial errors are weighted less than errors that occur later on in the simulation.

$$ITAE = \int_0^t (t \cdot |error|)dt \tag{5.2}$$

SAMPEX Satellite Results

The sliding mode controllers (fuzzy and normal) used the following estimated inertia matrix to determine their control efforts.

$$\hat{J} = \begin{bmatrix} 15 & 0 & 0 \\ 0 & 21 & 0 \\ 0 & 0 & 15 \end{bmatrix} \tag{5.3}$$

Note that this matrix is different from the correct inertia matrix given for SAMPEX. Also, SAMPEX was assumed to be capable of producing .01 N-m of torque from its reaction wheels.

Normal

The following simulation was conducted under normal conditions (in the absence of noise and disturbance). Each of the controllers' responses to following the path provided is shown below. Also, the individual control efforts (u1-3) and the norm of the control effort are shown.

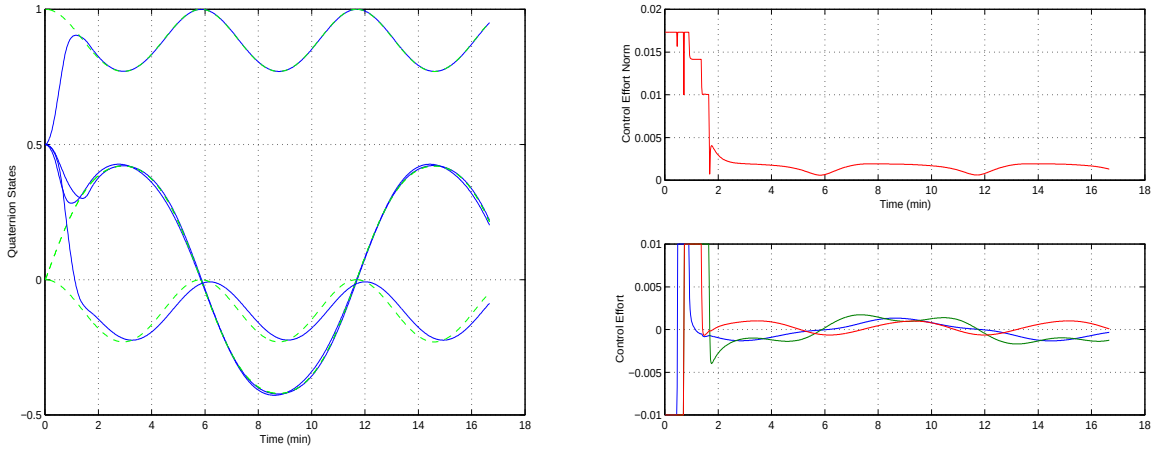


Figure 5-2. Quaternion states and control efforts for the fuzzy sliding mode.

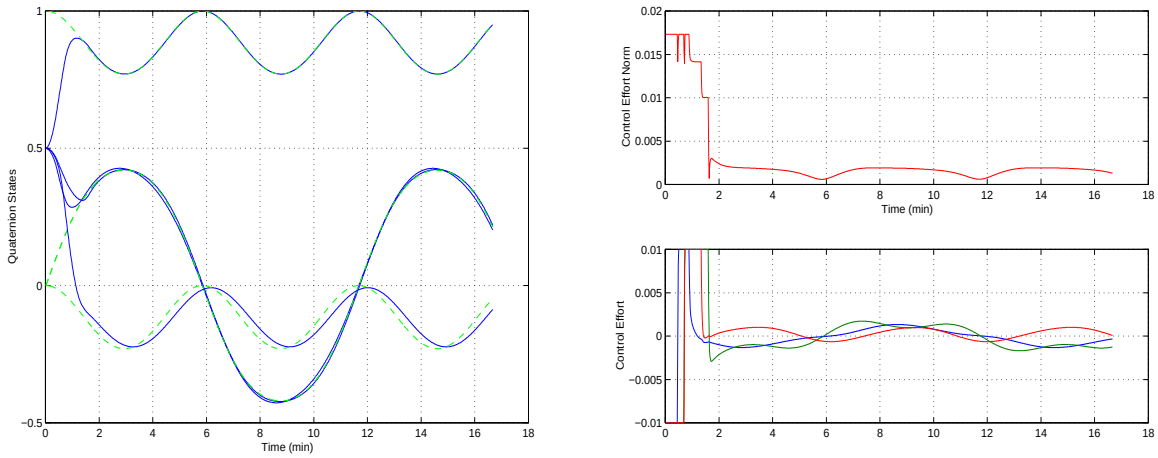


Figure 5-3. Quaternion states and control efforts for the sliding mode controller.

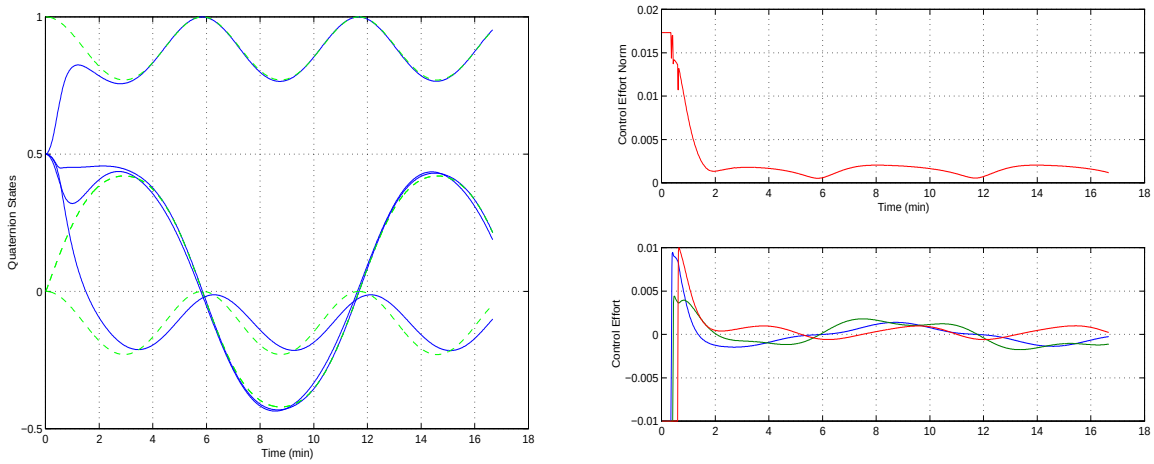


Figure 5-4. Quaternion states and control efforts for the fuzzy logic controller.

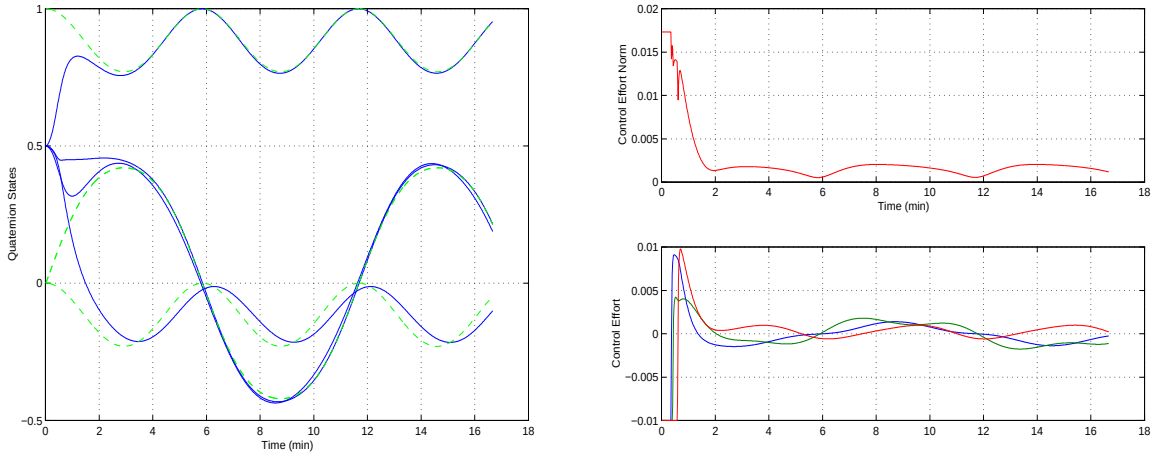


Figure 5-5. Quaternion states and control efforts for the PD controller.

Noise

Next noise was introduced in to the quaternion feedback to simulate sensor noise.

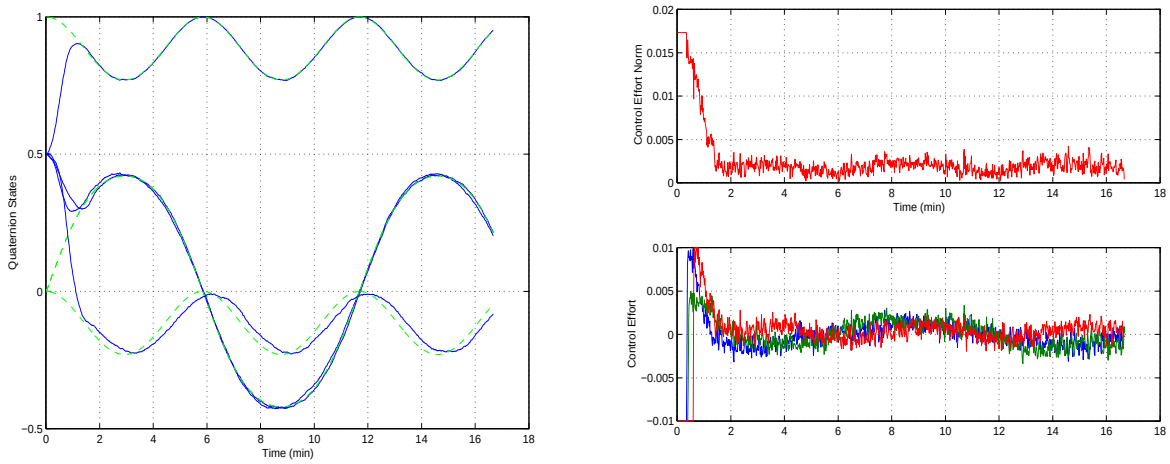


Figure 5-6. Quaternion states and control efforts for the fuzzy sliding mode controller in the presents of noise.

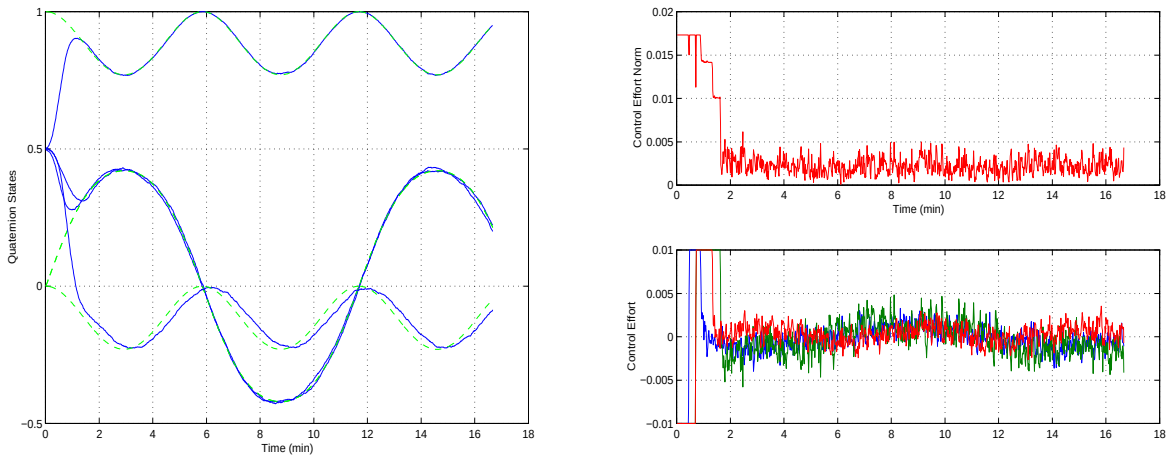


Figure 5-7. Quaternion states and control efforts of the sliding mode controller in the presents of noise.

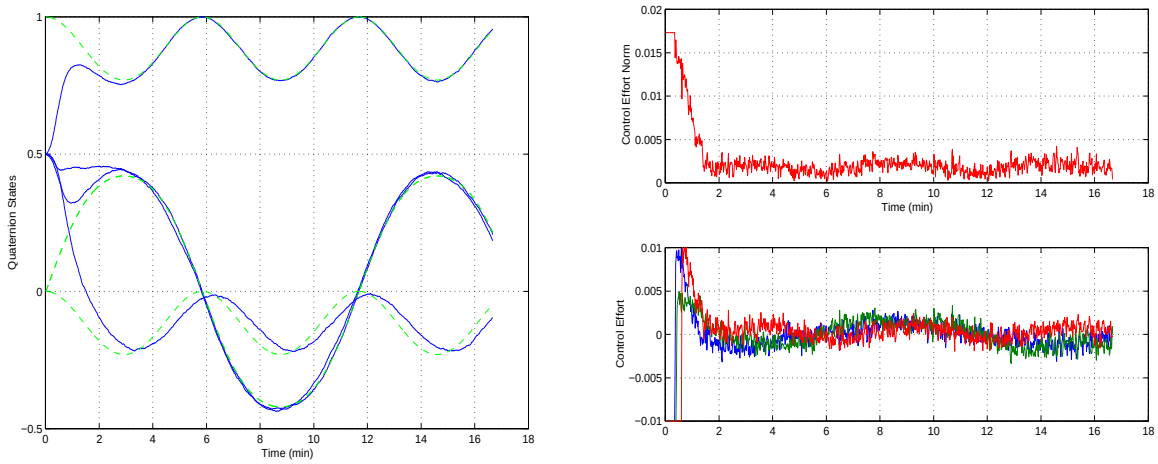


Figure 5-8. Quaternion states and control efforts of the fuzzy logic controller in the presents of noise.

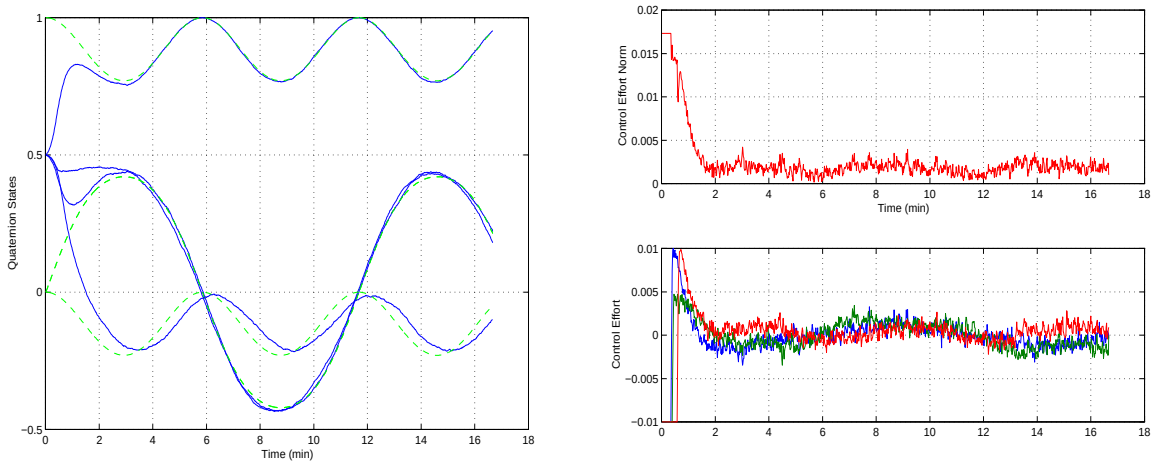


Figure 5-9. Quaternion states and control efforts of the PD controller in the presents of noise.

Disturbance

The disturbance added into the system for the SAMPEX satellite was .005 N-m which is 50% of the maximum control effort SAMPEX was capable of.

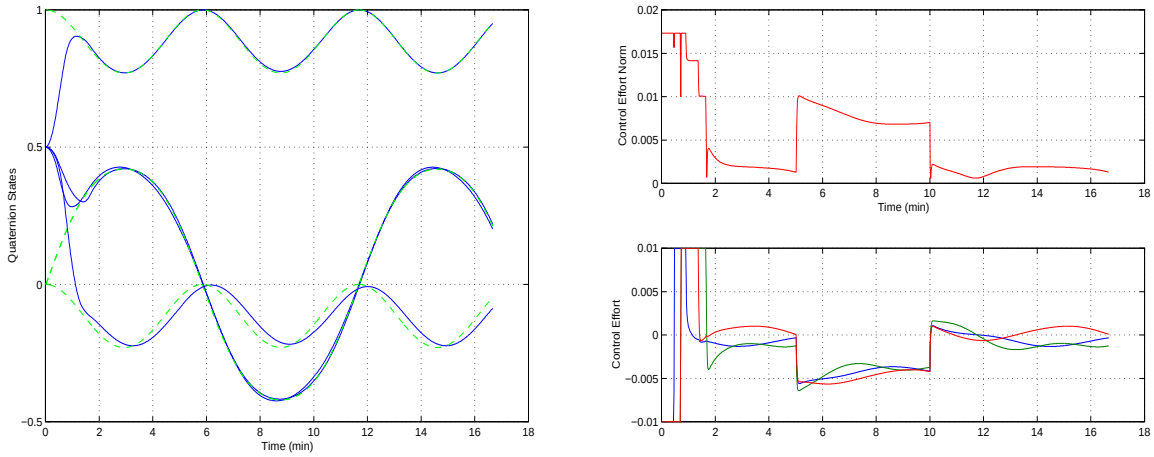


Figure 5-10. Quaternion states and control efforts of the fuzzy sliding mode controller with a disturbance.

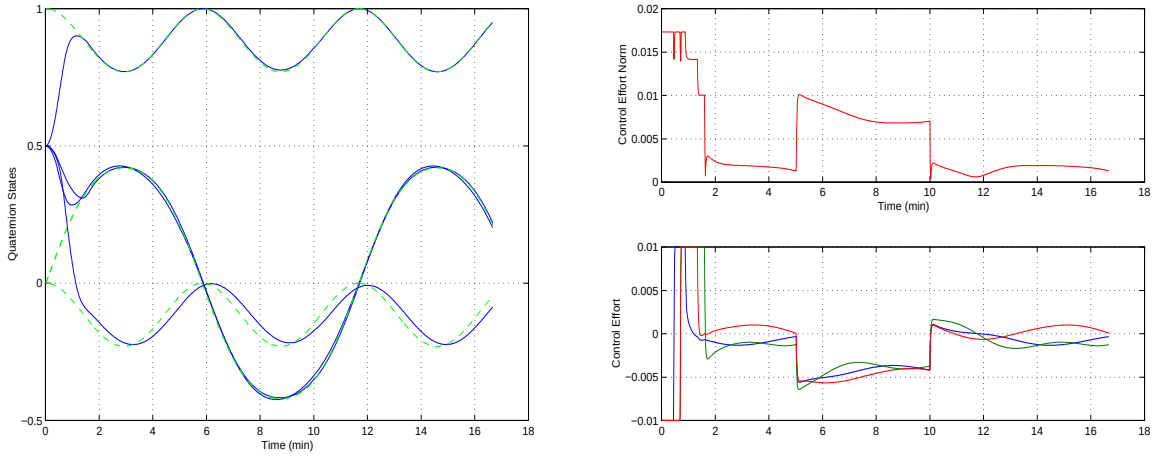


Figure 5-11. Quaternion states and control efforts of the sliding mode controller with a disturbance.

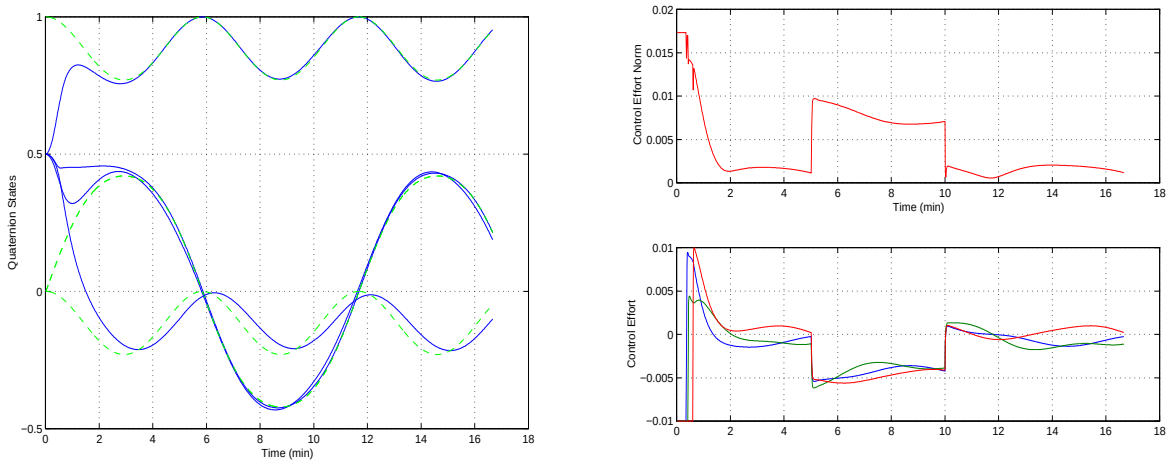


Figure 5-12. Quaternion states and control efforts of the fuzzy logic controller with a disturbance.

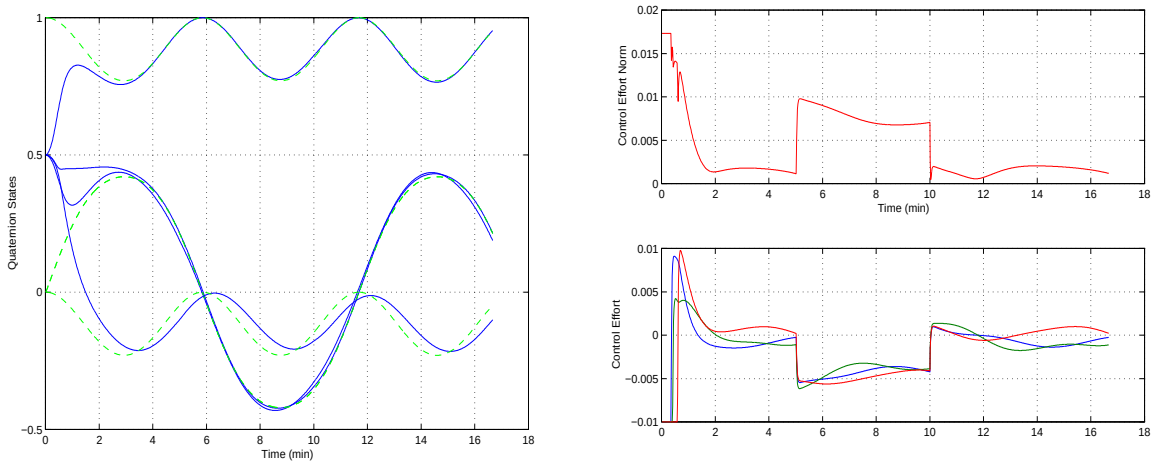


Figure 5-13. Quaternion states and control efforts of the PD controller with a disturbance.

SAMPEX Controller Summary In each of the two types of simulations, the fuzzy sliding mode controller provided better performance, but at a cost of a higher fuel consumption. The sliding mode controller was very close behind the performance of the fuzzy sliding mode, while both the fuzzy

controller and the PD controller where bringing up the rear. The fuel is represented in N-m-minutes.

Table 5-1: SAMPEX

Simulations	Fuzzy Sliding Mode	Sliding Mode	Fuzzy Logic	PD
Normal [ITAE]	15,425	15,581	22,060	22,128
[Fuel]	2.96	2.90	2.35	3.45
Noise [ITAE]	15,640	15,858	21,506	22,282
[Fuel]	3.55	3.47	2.64	2.57
Disturbance [ITAE]	15,830	16,040	22,293	22,485
[Fuel]	4.84	4,78	4.21	4.22

MAP Satellite Results

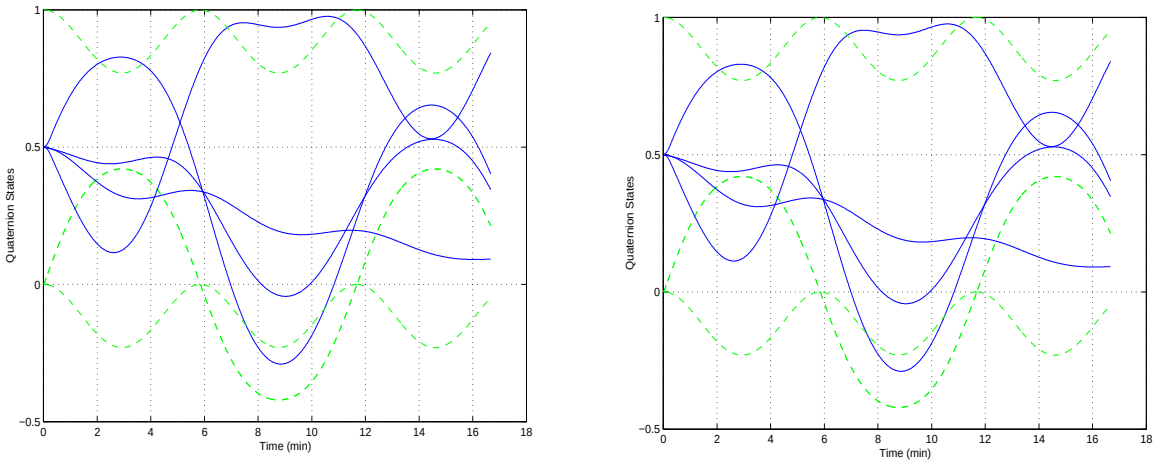
The MAP satellite was assumed to be able to produce a maximum of .2 N-m of torque from its reaction wheels. The estimated inertia matrix used in the sliding mode controllers is shown below.

$$\hat{J} = \begin{bmatrix} 399 & 0 & 0 \\ 0 & 377 & 0 \\ 0 & 0 & 377 \end{bmatrix} \tag{5.4}$$

Also, due to MAP being a larger satellite, the fuel consumption will be larger than either SAMPEX or SMART while the ITAE is approximately the same.

Normal

Both the fuzzy and PD controllers were unable to properly control the MAP satellite following a path without a disturbance or noise present. Thus only the sliding mode controllers will be discussed in this section.



(a) (b)

Figure 5-14. The (a) Fuzzy and (b) PD controllers were unable to follow the desired states in normal conditions.

Noise

Noise was introduced into the feedback quaternion states, simulating sensor noise.

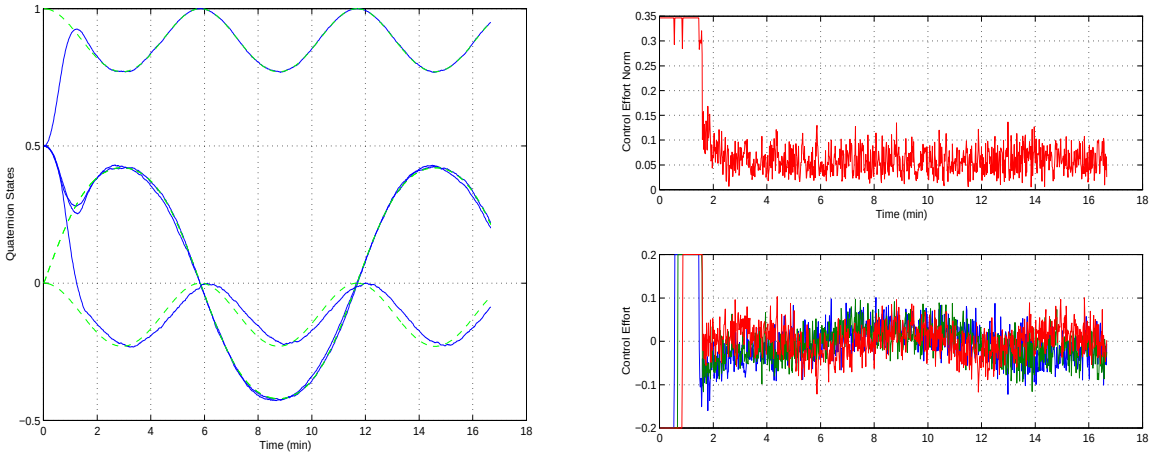


Figure 5-15. Quaternion states and control efforts for the fuzzy sliding mode controller in the presents of noise.

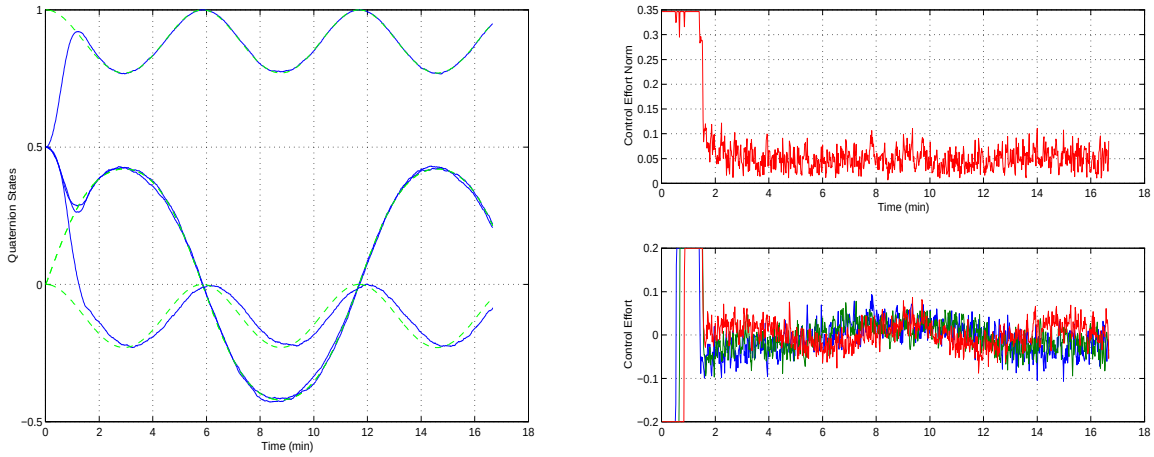


Figure 5-16. Quaternion states and control efforts for the sliding mode controller in the presents of noise.

Disturbance

The disturbance injected into the system was .1 N which is 50% of the maximum control effort.

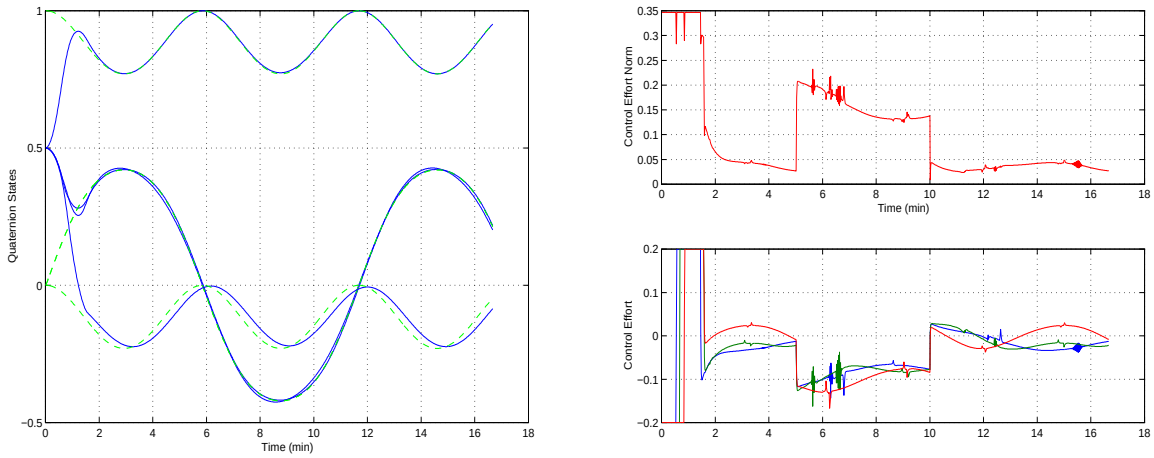


Figure 5-17. Quaternion states and control efforts for the fuzzy sliding mode controller with a disturbance.

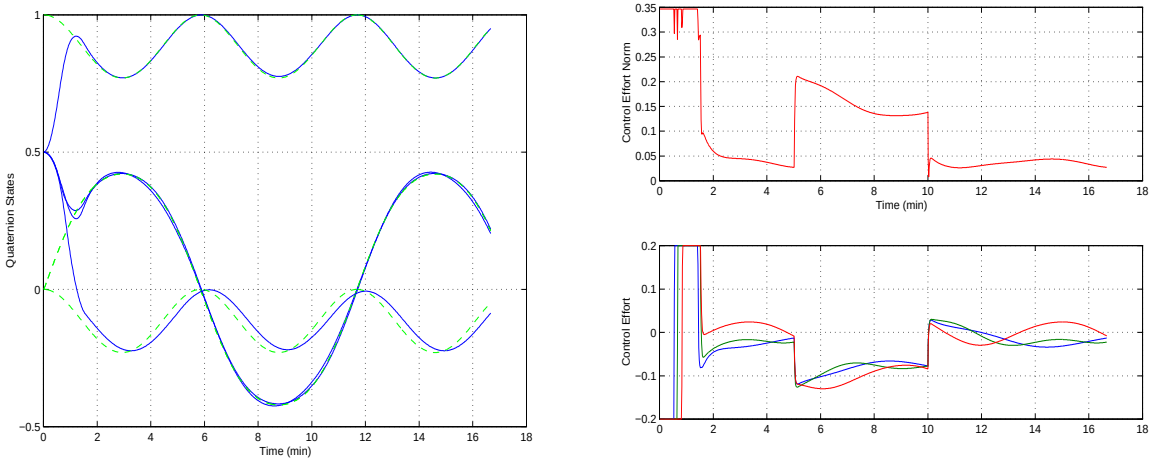


Figure 5-18. Quaternion states and control efforts for the sliding mode controller with a disturbance.

MAP Summary

Again, the sliding mode controllers were capable of providing the proper control to follow the path, while both the fuzzy logic and the PD controller were unable to. Once again, the fuzzy sliding mode provided better performance at the cost of a slightly higher fuel consumption. The fuel is represented in N-m-minutes.

Table 5-1: MAPS

Simulations	Fuzzy Sliding Mode	Sliding Mode
Noise [ITAE]	15,924	16,072
[Fuel]	86.62	77.42
Disturbance [ITAE]	15,391	15,954
[Fuel]	103.03	102.00

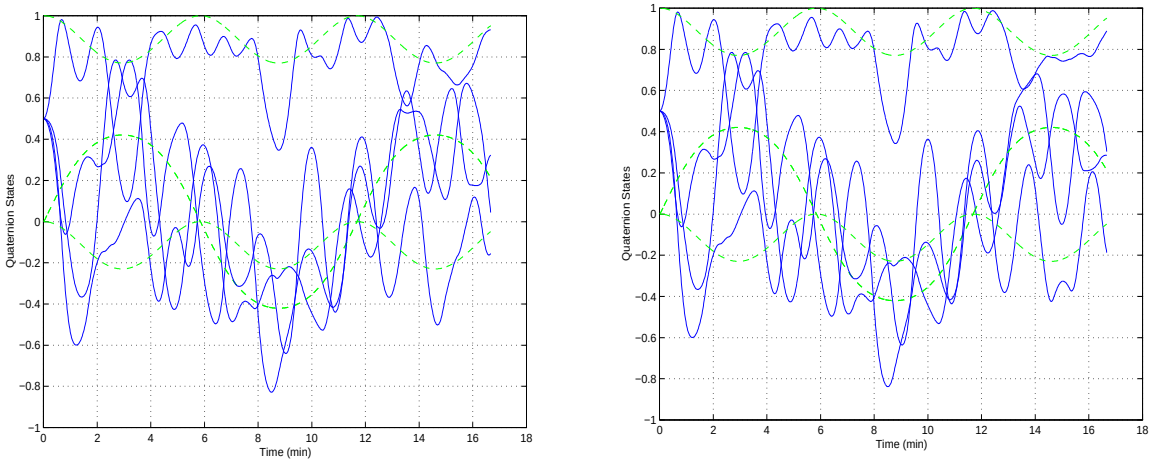
SMART Results

The maximum control effort for this satellite was 7.5E-4 N-m. The inertia matrix used for both of the sliding mode controllers is shown below.

$$\hat{J} = \begin{bmatrix} 0.565 & 0 & 0 \\ 0 & 0.719 & 0 \\ 0 & 0 & 0.454 \end{bmatrix} \tag{5.5}$$

Normal

Neither the fuzzy or the PD controller were able to properly control the satellite to follow the desired path. Thus no discussion of them will be present in this section.



(a) (b)

Figure 5-19. The (a) Fuzzy and (b) PD controllers were unable to control the MAP satellite under normal conditions.

Noise

The results for the four controllers in the presents of noise on the SMART satellite are shown below.

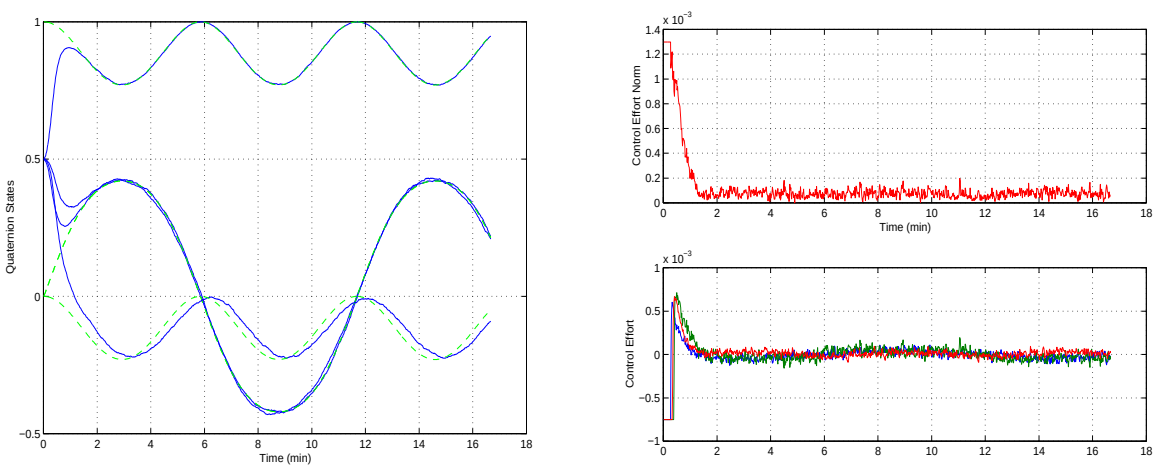


Figure 5-20. Quaternion states and control efforts for Fuzzy sliding mode controller in the presents of noise.

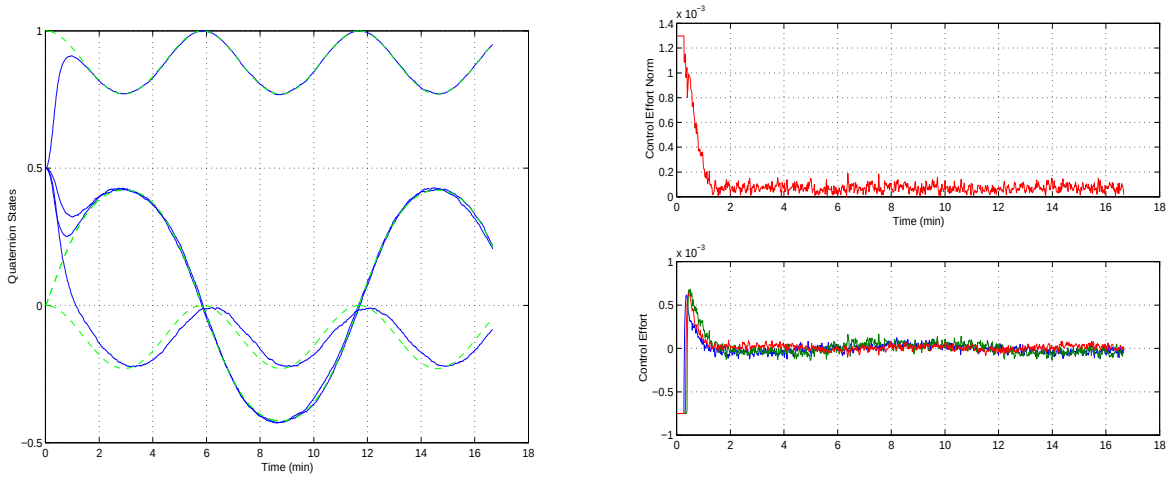


Figure 5-21. Quaternion states and control efforts for the sliding mode controller in the presents of noise.

Disturbance

The results of the four controllers in the presents of a disturbance is shown below. The disturbance was 3.5E-4 N-m which is 50% of the maximum control effort that SMART was capable of producing.

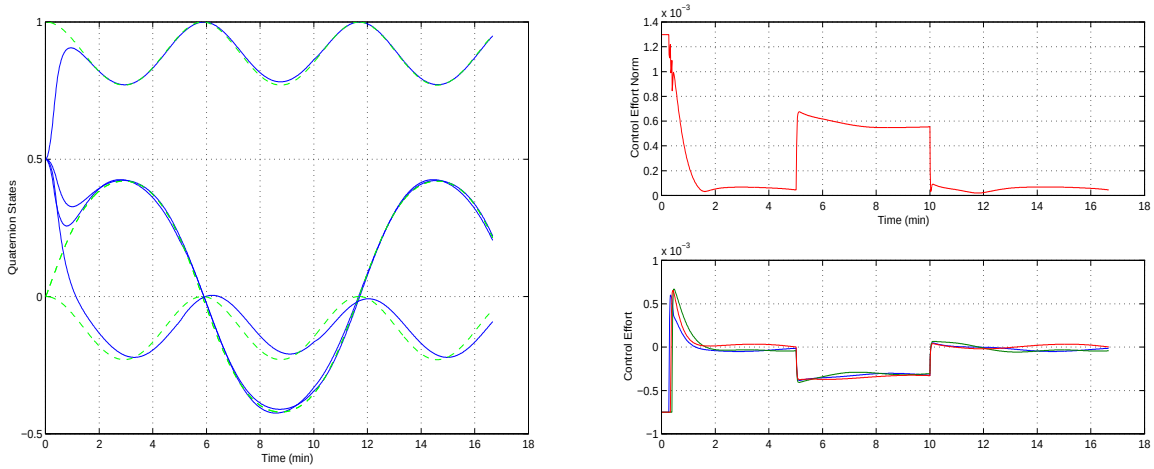


Figure 5-22. Quaternion states and control efforts for the fuzzy sliding mode controller with a disturbance.

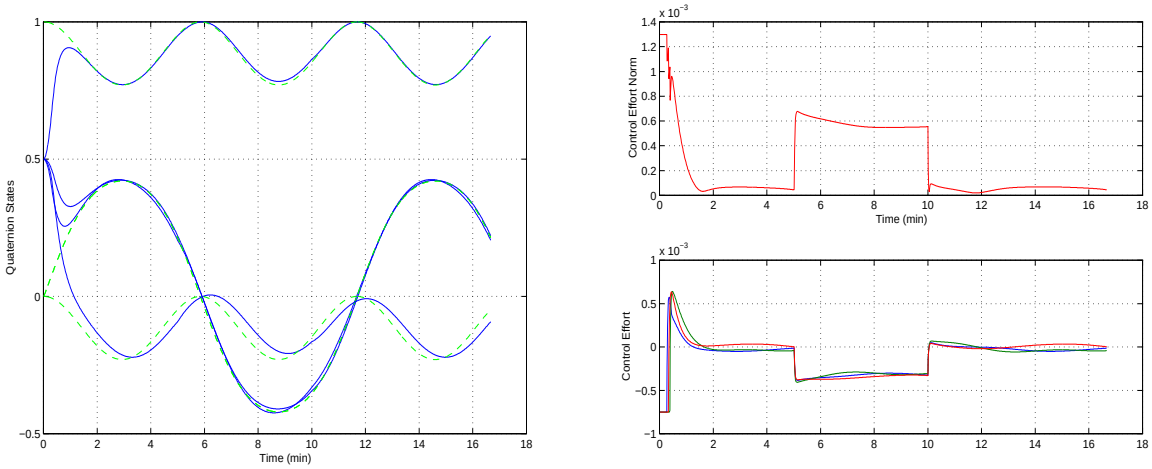


Figure 5-23. Quaternion states and control efforts for the sliding mode controller with a disturbance.

SMART Summary

The PD and fuzzy logic controllers were not able to properly control the satellite, thus eliminating them as truly generic controllers. Both of the sliding mode controllers were able to properly control the satellite with the fuzzy sliding mode controller providing better performance at almost no additional cost of fuel consumption. The fuel is represented in N-m-minutes.

Table 5-1: SMART

Simulations	Fuzzy Sliding Mode	Sliding Mode
Noise [ITAE]	16,074	16,374
[Fuel]	.12	.12
Disturbance [ITAE]	17,801	18,148
[Fuel]	.26	.26

VI. Conclusions

Throughout the test simulations of noise and disturbance for all three test spacecraft, the sliding mode controllers were able to prove proper control. The fuzzy logic sliding mode, was able to provide better control than the regular sliding mode, but at the price of a higher fuel consumption. The fuzzy logic controller and the PD controller performed on an even footing, with the fuzzy logic controller achieving a slight advantage over the PD controller. However, neither one of these two controllers were able to properly control anything other than the SAMPEX satellite.

The fuzzy sliding mode also has the draw back of being much more computationally expensive than the classical sliding mode controller. However, this can easily be overcome by partitioning the input and output space of the fuzzy controller into a look up table (i.e. similar to gain scheduling). This method is common in reducing the computational expense of fuzzy logic.

Overall, fuzzy logic did not reach the level of performance expected in the outset of this project. It was believed that fuzzy logic, which is model independent, would provide a much better performance with much less design time. This was however not the case. Significant design time still went into designing the fuzzy logic controller and the fuzzy sliding mode controller, with only slightly better performance. In fact, fuzzy sliding mode had to be utilized due to the fact that fuzzy logic alone did not accomplish the desired results. In conclusion, sliding mode with or without fuzzy logic, was capable of performing generic control across very different spacecraft.

Finally, some level of adaptation would need to be incorporated into the controller to estimate the inertia matrix in order to truly provide generic control. Originally this was going to be accomplished by RL, but it could also be provided by current adaptive control theory such as was done by Ahmed et al [40] or the use of a kalman filter.

VII. References

1. Wie, B., and Barba, P., “Quaternion Feedback for Spacecraft Large Angle Maneuvers,” Journal of Guidance, Control and Dynamics, Vol. 8, No. 3, May-June 1985, pp. 360-365.
2. Crassidis, J.L., and Markley, F.L., “Predictive Filtering for Attitude Estimation Without Rate Sensors,” Journal of Guidance, Control and Dynamics, Vol. 20, No. 3, May-June 1997, pp. 522-527.
3. Sakkas, Chris, “An Introduction to Fuzzy Logic,” Circuit Cellar INK, Issue #75, Oct. 1996, pp. 12-14.
4. Zedah, L. A., “Fuzzy Control: A Personal Perspective,” Control Engineering, July1996, pp. 51-52.
5. Sibigtroth, Jim, “Fuzzy Logic for Embedded Microcontrollers,” Circuit Cellar INK, Issue #56, March 1995, pp. 30-37.
6. Bartos, Frank J., “Fuzzy Logic Reaches Adulthood.” Control Engineering, July 1996, pp.50-55.
7. Walchko, K., Petroff, N., Mason, P.A.C., and Fitz-Coy, N., “Development of a Generic Hybrid Fuzzy Controller,” Intelligent Engineering Systems Through Artificial Neural Networks, Vol. 8, 1998, pp. 214-218, St. Louis, MS.
8. Brehm, T., and Kuldip, S. R., “Hybrid Fuzzy Logic PID Controller,” IEEE International Conference on Fuzzy Systems, Vol. 3, 1994, pp. 1682-1687.
9. Brehm, T., and Kuldip, S. R., “Proportional Fuzzy Logic Controller: Classical Proportional-Plus-Derivative Like Response,” IEEE International Conference on Systems, Man, and Cybernetics, Vol. 3, 1995, pp. 2029-2033, Piscataway, NJ.
10. Gonsalves, P. G., and Caglayan, A. K., “Fuzzy Logic PID Controller for Missile Terminal Guidance,” IEEE International Symposium on Intelligent Control - Proceedings, Vol. 1, 1995, pp. 377-382.
11. Misir, D., Malki, H. A., and Chen, G., “Design and Analysis of a Fuzzy Proportional-Integral-Derivative Controller,” Fuzzy Sets and Systems, Vol. 79, 1996, pp. 297-314.
12. Von Altrock, Constantin, “Practical Fuzzy-Logic Design,” Circuit Cellar INK, Issue #75, Oct. 1996, pp. 16-20.
13. Brehm and Rattan, “Hybrid Fuzzy Logic PID Controller,” IEEE International Conference on Fuzzy Systems, Vol.3, 1994, pp. 1682-1687.
14. Misir, D., Malki, H. A., Chen, G., “Design and Analysis of a Fuzzy Proportional-Integral-Derivative Controller,” Fuzzy Sets and Systems 79 (1996) 297-314.

15. Woodard, M., “Fuzzy Open-Loop Attitude Control for the FAST Spacecraft,” http://fdd.gsfc.nasa.gov/ mwoodard/aiaa_96/aiaa_96.html.

16. Steyn, William H., “Comparison of Low-Earth-Orbit Satellite Attitude Controllers Submitted to Controllability Constraints,” Journal of Guidance, Control and Dynamics, Vol. 17, No. 4, July-Aug 1994, pp. 795-804.

17. Steyn, William H., “Fuzzy Control for a Non-Linear MIMO Plant Subject to Control Constraints,” IEEE Transactions on Systems, Man, and Cybernetics, Vol. 24, No. 10, Oct. 1994, pp 1565-1570.

18. Ju, M. S. and Yang, D.L., “Design of Adaptive Fuzzy Controls Based on Natural Control Laws,” Fuzzy Sets and Systems, 81(1996), pp. 191-204.

19. Lo, J. C. and Yang, C. H., “A Heuristic Error-Feedback Learning Algorithm for Fuzzy Modelling,” IEEE Transactions of Systems, Man and Cybernetics, Vol. 29, No. 6, November 1999, pp. 686-691.

20. Guo, J. and Haung S. H., “Control of an Autonomous Underwater Vehicle Testbed Using Fuzzy Logic and Genetic Algorithms,” Proceedings of the 1996 Symposium on Autonomous Underwater Vehicle Technology, June 2-6, 1996, Monterey, California, pp.485-489.

21. Buijtenen, W. M., Schram, G., Babuska, R., and Verbruggen, H. B., “Adaptive Fuzzy Control of Satellite Attitude by Reinforcement Learning,” IEEE Transactions on Fuzzy Systems, Vol. 6, No. 2, May 1998, pp. 185-194.

22. Visoli, Antonio, “Fuzzy Logic Based Set-Point Weight Tuning of PID Controllers,” IEEE Transactions on Systems, Man, and Cybernetics, Vol. 29, No. 6, November 1999, pp.587-592.

23. Erbatur, K., Kaynak, O., Sabauouoc, A. and Rudas, I., “Fuzzy Parameter Adaptation for a Sliding Mode Controller as Applied to the Control of an Articulated Arm, “Proceedings of the 1997 IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, April 1997, pp. 817-822.

24. Lo, S. and Chen, Y., “Smooth Sliding Mode Control for Spacecraft Attitude Tracking Maneuvers,” Journal of Guidance, Control, and Dynamics, Vol. 18, No. 6, Nov.-Dec. 1995, pp.1345-1348.

25. Coleman, C. P. and Godbole, D., “A Comparison of Robustness: Fuzzy Logic, PID, & Sliding Mode Control,” Proc. of 3rd IEEE Intl. Conf. On Fuzzy Systems, pp. 1654-1560.

26. Cristi, Roberto, Burl, Jeffery, and Russo, Nick, “Adaptive Quaternion Feedback Regulation for Eigenaxis Rotations,” Journal of Guidance, Control and Dynamics, Vol. 17, No. 6, Nov-Dec 1994, pp. 1287-1291.

27. Mason, P., and Walchko, K., “Fuzzy Gain Scheduling for Satellite Attitude Control,” to be published in the Proceedings of the Flight Mechanics/Estimation Theory Symposium, Goddard Space Flight Center, Greenbelt, MD, 1999.

28. Walchko, K., “Development of a Fuzzy Logic MIMO Controller for Satellite Attitude Control,” Master’s Thesis, University of Florida at Gainesville, May, 1999.

29. Gulley, N., and Roger Jang, J. S. Fuzzy Logic Toolbox for use with Matlab. Nortick, MA: The MathWorks, Inc., 1995.

30. Petroff, N., Walchko, K., Mason, P.A.C., Reisinger, K.D., “Numerical Stability Analysis of a Fuzzy Controller”, Intelligent Engineering Systems Through Artificial Neural Networks, Vol. 8, 1998, pp. 219-224, St. Louis, MS.

31. Sio, K. C. and Lee C. K., “Stability of Fuzzy PID Controllers,” IEEE Transactions on Systems, Man, and Cybernetics, Vol. 28, No. 4, July 1998, pp. 490-495.

32. Sutton, Richard S. and Barto, Andrew G., Reinforcement Learning, An Introduction, The MIT Press, 1998, p. 148.

33. Baker, D. N., Mason, G. M., Figueroa, O., Colon, G., Watzin, J. G., and Aleman, R. M., “An Overview of the Solar, Anomalous, and Magnetospheric Particle Explorer (SAMPEX) Mission,” IEEE Transactions on Geoscience and Remote Sensing. Vol. 31 No. 3, May 1993.

34. Flatley, Thomas W., Forden, Josephine K., Henretty, Debra A., Lightsey, E. Glenn, Markley, F. Landis, “MME-Based Attitude Dynamics Identification and Estimation for SAMPEX,” Proceedings of the Flight Mechanics/Estimation Theory Symposium, Goddard Space Flight Center, Greenbelt, MD, 1990, pp.497-511.

35. Pastenea, M. and Grassi, M., “SMART Attitude Acquisition and Control,” The Journal of the Astronautical Sciences, Vol. 46, No. 4, Oct. - Dec. 1998, pp. 379-393.

36. Rimrott, F. P. J., Introduction to Attitude Dynamics, Springer-Verlag, 1989.

37. Wertz, James R., Spacecraft Attitude Determination and Control, Boston: Kluwer Academic Publishers, 1995.

38. Wie, Bong, “Space Vehicle Dynamics and Control”, AIAA Education Series, J. S. Przemieniecki, 1998.

39. Slotine, Jean-Jacques E., Li, Weiping, Applied Nonlinear Control, Prentice Hall, 1991.

40. Ahmed, Jasim, Coppola, Vincent T., and Bernstein, Dennis S., “Adaptive Asymptotic Tracking of Spacecraft Attitude Motion with Inertia Matrix Identification,” Journal of Guidance, Control, and Dynamics, Vol. 21, No. 5, Sept-Oct. 1998, pp. 684-692.

41. Kuipers, Jack B., Quaternions and Rotation Sequences, Princeton Univ. Press, 1999.

42. Crane, C., and Duffy, J., Kinematic Analysis of Robot Manipulators, Cambridge University Press, 1998.

43. Kosko, Bart. Fuzzy Engineering. New Jersey: Prentice Hall, 1997.

44. Reznik, Leonid, Fuzzy Controllers, Newnes, 1995.

45. Dorf, R., Bishop, R., Modern Control Systems, 8th Ed. Menlo Park, CA: Addison Wesley, 1998.

46. Ogata, K., Modern Control Engineering, 3rd Ed. Upper Saddle River, NJ: Prentice-Hall, 1997.

VIII. Biographical Sketch

Kevin J. Walchko

He was born on 1 Nov. 1972 in Sarasota, FL. He attended the University of Florida from 1991-1997 and received a BS degree in mechanical engineering. Most of the research Kevin has done was in the areas of controls and dynamics. Specifically, Kevin has focused on intelligent controls such as fuzzy logic and neural networks. Kevin received his master's degree in mechanical engineering Spring of 1999, and will stay at the University of Florida to complete a Ph.D. degree in mechanical engineering. Also during this time period, Kevin was a member of the Florida Army National Guard. Kevin served seven years as a cavalry scout and rose to the rank of sergeant (E5). Most notable was his service during both hurricane Andrew and Opal which hit the Florida coast. Kevin is also concurrently working on a masters degree in electrical engineering. He is specializing in robotics and digital signal processing.

APPENDIX A: Small Angle Approximation

Quaternions

Ref. [41] defines the euler angles to quaternion conversions as:

$$
\begin{aligned}
q_r &= \cos\left(\frac{\psi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) + \sin\left(\frac{\psi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right) \\
q_x &= \cos\left(\frac{\psi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right) - \sin\left(\frac{\psi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) \\
q_y &= \cos\left(\frac{\psi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) + \sin\left(\frac{\psi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right) \\
q_z &= \sin\left(\frac{\psi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) - \cos\left(\frac{\psi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right)
\end{aligned}
\tag{A.1}
$$

where

$$
\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} roll \\ pitch \\ yaw \end{bmatrix}
\tag{A.2}
$$

Now when the small angle approximation is taken on (A-1).

$$
\begin{aligned}
q_r &= 1 + \frac{\psi\theta\phi}{8} \approx 1 \\
q_x &= \frac{\phi}{2} - \frac{\psi\theta}{4} \approx \frac{\phi}{2} \\
q_y &= \frac{\theta}{2} + \frac{\psi\phi}{4} \approx \frac{\theta}{2} \\
q_z &= \frac{\psi}{2} - \frac{\theta\phi}{4} \approx \frac{\psi}{2}
\end{aligned}
\tag{A.3}
$$

APPENDIX B: Attitude Representations and Rotation Matrices

Euler Angles

Euler angles are typically though of in terms of roll, pitch, and yaw. These terms are shown graphically below.

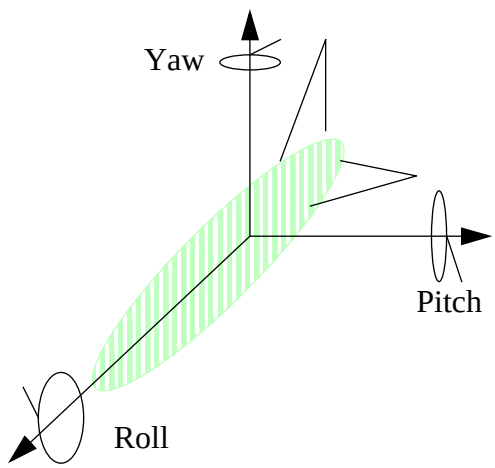


Figure B-1. Spaceship with pitch, yaw, and roll

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} roll \\ pitch \\ yaw \end{bmatrix}$$

In order to perform these operations on the spacecraft, an attitude matrix (A) is used to find the new orientation of the spacecraft given the old orientation. Given below are the attitude matrices for rotations about the x, y, and z-axes.

$$A_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{B.2}$$

$$A_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{B.3}$$

$$A_z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.4}$$

Now, subsequent rotations about these primary axes can be accomplished by multiplying the matrices together. Thus, successive rotations about the z-axis, x-axis, and y-axis are given by:

$$A_{xyz}(\psi, \theta, \phi) = A_z(\psi) \cdot A_y(\theta) \cdot A_x(\phi) \tag{B.5}$$

Quaternions

Quaternions, also known as Euler symmetric parameters, are more mathematically efficient way to compute rotations of rigid and non-rigid body systems than traditional methods involving standard rotational matrices or Euler angles. Quaternions have the advantage of few trigonometric functions needed to compute attitude. Also, there exists a product rule for successive rotations that greatly simplifies the math, thus reducing processor computation time. The major disadvantage of quaternions is the lack of intuitive physical meaning. Most people would understand where a point was if they were given [1 2 3], however, few would comprehend where a point was if given the quaternion [1 2 3 4]. This section does not attempt to provide the extensive understanding needed to employ quaternions but rather a simple introduction. Further information can be found in Wertz [37] or Crane and Duffy [42].

Quaternion Algebra.

The quaternion is composed of a scalar and a vector part. The scalar is a redundant element that prevents singularities from occurring since the four elements are all dependent upon each other.

$$q = \begin{bmatrix} q_x & q_y & q_z & q_r \end{bmatrix}^T \tag{B.6}$$

Table H-1: Quaternion Algebra Summary

Add / Subtract	$q \pm q' = \left[q_x \pm q_x' \;\; q_y \pm q_y' \;\; q_z \pm q_z' \;\; q_r \pm q_r'\right]^T$
Scalar Multiplication	$\alpha q = \left[\alpha q_x \;\; \alpha q_y \;\; \alpha q_z \;\; \alpha q_r\right]^T$
Quaternion Multiplication	$q \cdot q' = \begin{bmatrix} q_4 & q_3 & -q_2 & -q_1 \\ -q_3 & q_4 & q_1 & -q_2 \\ q_2 & -q_1 & q_4 & -q_3 \\ q_1 & q_2 & q_3 & q_4 \end{bmatrix} \cdot q'$
Conjugate	$q_{conj} = \left[-q_x \;\; -q_y \;\; -q_z \;\; q_r\right]^T$
Inverse	$q^{-1} = \dfrac{q_{conj}}{q_{norm}}$

Rotations of rigid bodies in space.

In order to perform a rotation (ϕ) of a rigid body about an arbitrary moving/fixed axis (e) in space, where e is defined by

$$e = \left[e_1 \;\; e_2 \;\; e_3\right]^T \tag{B.7}$$

The quaternion representation of this operation is

$$q_{13} = e \cdot \sin\left(\frac{\phi}{2}\right) \tag{B.8}$$

$$q_4 = \cos\left(\frac{\phi}{2}\right) \tag{B.9}$$

Notice that only one sine and one cosine function call is needed to calculate a quaternion. While euler would require three sine and three cosine function calls, one each for roll, pitch, and yaw. Since trigonometric function calls are computationally expensive, this is a great savings.
The attitude matrix (A) for this is

$$A(q) = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{B.10}$$

Successive rotations can be accomplished by multiplying the attitude matrices together. Thus for two successive rotations, the attitude matrix that would accomplish both would be:

$$A(q'') = A(q') \cdot A(q) \tag{B.11}$$

Thus, the above equation will provide the quaternion with the two successive rotations already embedded into it. Had the two euler attitude matrices been calculated and multiplied together, that would have resulted in twenty-seven multiplication operations rather than the sixteen multiplication operations from quaternion multiplication.

Hopefully it can be seen that quaternions are better than other Euler operations to determine attitude. Quaternions lack singularities due to one redundant element. They also lack the computationally intensive trigonometric functions, and contain a simplified way to determine successive rotations about an arbitrary axis.

APPENDIX C: Fuzzy Logic

Fuzzy logic was biologically inspired by a world of living organisms which do not face life in the classical binary logic sense of true or false. Living organisms deal with sometimes subtle, gradual changes of truth, where there is no crisp black and white, but rather shades of gray. Fuzzy logic is a method which allows computers to deviate from the classical binary decision making method, and embrace the shades of gray world of human existence.

Fuzzy logic is a vast and complex topic. Thus, the purpose of this section is not to cover all aspects of fuzzy logic, but rather introduce an unfamiliar reader to the concept of fuzzy logic and therefore lay a foundation for material throughout the rest of this report. For more information on fuzzy logic, the reader is referred to Kosko [43] and Leonid [44].

Classical and Fuzzy Set Theory

In the sections below, a description of classical set theory, fuzzy set theory and fuzzy logic as a whole is presented. With this basic understanding of the difference between classical and fuzzy, a better understanding of the under lying principles of fuzzy logic should emerge.

Classical Set Theory.

Classical set theory is marked by a true or false, on or off, 1 or 0 relationship. There is a clear and unambiguous line in the sand where false ends and truth begins. For example, if Y is a classical set composed of real numbers whose value is greater than 10, then

$$Y = \{x \mid x > 10\}$$

There is a crisp boundary where 10.00000 does not belong to Y, but 10.0000001 does belong to Y. This type of logic is simple for computers to understand and use, but classical sets are non-intuitive to humans who do not see real life situations in black and white but rather shades of gray.

Fuzzy Set Theory.

Fuzzy set theory on the other hand is quite different from the classical form. Fuzzy set theory introduces these shades of gray that is inherent to human perception of everyday life and allows computers to participate in this ambiguity of life. For example, suppose that Y is the set tall trees and x is the height of an individual tree greater than ten meters tall. The notation for fuzzy set theory is:

$$Y = \mu_{x>10}(x)$$

Another example of fuzzy set theory is, if fifty people where asked what they considered to be fast driving, there would be fifty different answers. This is because the human experience is a very subjective one. What one person considers fast, someone else might think is too fast or not fast enough. This also shows the shades of gray that humans see, but classically logical computers do not.

Fuzzy Rule Base

Now we need to define the framework for how we are going to use fuzzy logic. Membership functions are fuzzy sets (as described previously) and fuzzy rules are a method of joining sets together. A series of rules define what is referred to as the rule base. In controls, this would be your input/output control surface.

From the previous examples, let X be the sample space (universe of discourse) from which x is drawn and $Y \subset X$ where Y is a collection of elements of x. In the classical instance, x will either belong to or not belong to Y. Thus for each element x in X, a set of ordered pairs which will denote the degree of membership, (x,0) and (x,1), can be constructed to represent the classical set Y.

Now in fuzzy logic, a similar pairing will occur to denote the degree of membership when given a fuzzy set Y in X where X contains a collection of elements of x. Now each element x in X can be mapped to Y by a set of ordered pairs:

$$Y = \{(x, \mu_Y(x)) \mid x \in Y\}$$

where $\mu_Y(x)$ is the membership function (MF) for fuzzy set Y. The MF will map each element x to a membership value between 0 (false) and 1 (true).

Fuzzy Inference Fuzzification and Fuzzy Reasoning.

Fuzzification

The fuzzy logic process starts with obtaining some crisp values from user input or sensor on the system that is being controlled. The first thing that has to be done is to fuzzify these crisp numbers into fuzzy numbers. This process is of course accomplished by use of the appropriate MFs to which the crisp numbers belong.

Once all of the inputs are fuzzified, the rules need to be applied to the now fuzzy numbers. In fuzzy logic, all rules are executed in parallel; thus no rules in the rule base are ever not evaluated.

Rules.

Fuzzy logic rules describe the relationship between the input and the output, and (from a controls standpoint) defines an input/output surface of control effort. This surface has been used to quantify stability by Petroff et al [30] of fuzzy controllers. Fuzzy rules take the standard if then form, i.e.

IF (Y is x) then (B is a) IF (antecedent) then (premise)

where x is an input, Y and B are a fuzzy sets, and a is an output. A fuzzy system can have multiple inputs and multiple outputs (MIMO), where the inputs and outputs are joined together by ANDs and ORs.

IF (A is a) AND (B is b) AND (C is c) AND ... then (AA is aa) AND ...

In addition, many other operators can be used to combine and manipulate fuzzy rules. One example is hedges (i.e. Not, Very, Somewhat, below, above, etc) [43]. These hedges will not be described here since they were not used in this work.

Defuzzification.

Once the fuzzy reasoning is complete, a fuzzy output can be produced. However, this output is only meaningful to humans and not to computers and other classical logic equipment. Thus, a method is necessary to transform the resulting fuzzy output into a crisp numerical output. There are several methods to accomplish this. Some of the most meaningful schemes are described below and illustrated in Figure C-1.

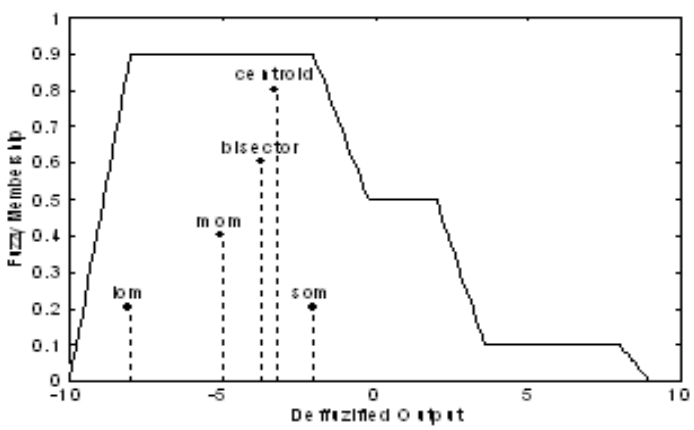


Figure C-1. Comparison of various defuzzification techniques [29]

- **Winner takes all / Singleton** - this is the simplest defuzzification method to implement. The output MF with the highest membership wins.
- **Mean of max (mom)** - this is the average.
- **Bisector of area** - this is a partition of the resulting area into two regions.
- **Centroid of area** - this is the most popular defuzzification method. This method finds the center of mass of the output region.

The entire process is shown in Figure C-2. The n inputs on the left are fuzzified by MFs. Then each of the fuzzified values is put into the rules that produce m outputs. Each of these m outputs are combined (there are various methods) and the defuzzification of the resulting area is found. Now the whole process of fuzzification, rule application, and defuzzification can be put together.

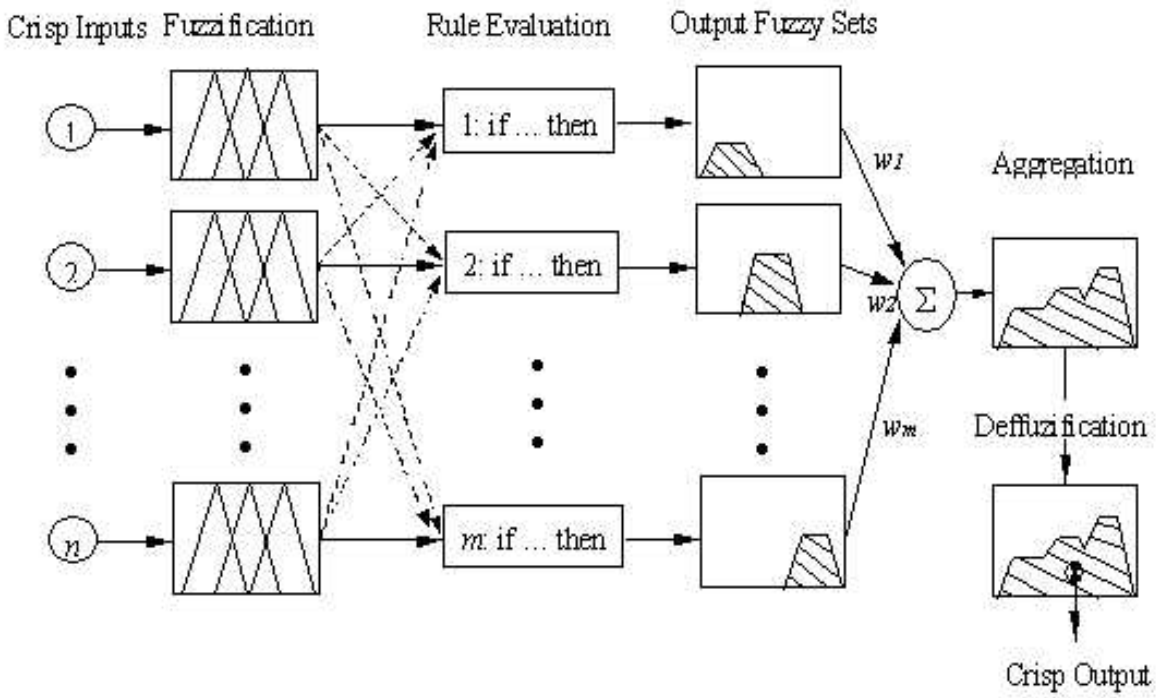


Figure C-2. Fuzzy Logic from input to output [29]

APPENDIX D: Source Code

Main Program

```
% func( whatSimType, ControllerType, satellite,
%other)
%
%**********************************************
% kevin's satellite simulator
%**********************************************
% Xn - current state [qn;Ln]
% qn - current quaternion
% Ln - angular momentum
%
%
% This simulation allows you to simulate
%attitude maneuvers on three different
%satellites, using three different controllers.
%It also allows you to do step or path
%following, with noise or disturbances
%included.
%
% misc variables:
% m - length of simulation in steps
% dt - time of each step (seconds)
% ii - current step
% tn - current time (seconds)
% e_p - attitude error of satellite
% e_i - integral of error
% e_d - velocity error of satellite
% uc - control effort
%
%
%*******************************
% simulation types
%*******************************
%  step - step
%  path - follows a path
%*******************************
%
%*******************************
% other
%*******************************
%  normal - nothing
%  noise - includes noise
%  disturbance - includes disturbance
%*******************************
%
%*******************************
% satellites
%*******************************
%   samepx
%   maps
%   smart
%*******************************
%
%*******************************
% controller types
%*******************************
%  fuzzy - fuzzy controller
%  pd - normal PD controller
%  pid - normal PID controller
%  slide - sliding mode controller
%  fuzslide - fuzzy sliding mode controller
%*******************************

function output=func(whatSimType,ControllerType,satel-
lite,other)

m=1000;                 % stop iteration
ii=0;                   % start iteration
wd=[0;0;0];

TD=1.0;
dt=1;                   % time step
tn=0;                   % start time
FLCSM='try12_sat';
FLC='try4_sat';

e_i=0;

%**** initial states ******
direction=[1;1;1;1];
qn=[direction/norm(direction)];   % orientation
Ln=[1e-15;1e-15;1e-15];          % angular momentum
Xn=[qn;Ln];                      % state vector

if strcmp(ControllerType,'fuzzy')
  % Define fuzzy control
  fismatrix=readfis(FLC);

elseif strcmp(ControllerType,'slide') | strcmp(Controller-
Type,'fuzslide')
  % Define fuzzy control
  fismatrix=readfis(FLCSM);
end

% inertia matrices
if strcmp(satellite,'sampex')
  iinertia=[0.06444960041248 0 0;
        0 0.04625661524405 0.00058906284347;
        0 0.00058906284347 0.06565014298225];

  inertia=[15.51599999999938 0 0
```

```
        0 21.62100000000062 -0.19400000000166
        0 -0.19400000000166 15.2340000000057];

elseif strcmp(satellite,'maps')
  inertia=[3.99e2 -2.81 -1.31
          -2.81 3.77e2 2.54
          -1.31  2.54 3.77e2];

  iinertia=inv(inertia);

elseif strcmp(satellite,'smart')
   inertia=[5.65e-1 4.21e-4 -4.86e-2;
            4.21e-4 7.19e-1 -5.8e-3;
           -4.86e-2 -5.8e-3 4.54e-1];

  iinertia=inv(inertia);
else
  disp(sprintf('ERROR!))
end


%***** desired states ******
if strcmp(whatSimType,'step')
   Ld=[0;0;0];              % angular momentum
  axis=[1;1;1];
  axis=axis/norm(axis);
  angle=pi;

  qdirection=[axis(1)*sin(angle/2);
axis(2)*sin(angle/2);
axis(3)*sin(angle/2);
   cos(angle/2)];

  direction=qmult(qn,qdirection);
  qd=[direction/norm(direction);Ld];

elseif strcmp(whatSimType,'path')
  s=sin(2*pi/700*(0:dt:dt*(m+1)));
  zs=zeros([1,m+1]);
  ds=diff(s)/dt;
  for i=1:m+1,
    % L = I*w angular momentum
    dsp(:,i)=inertia*[ds(i);ds(i);zs(i)];
  end
  qpath=[e2q(s(1:m+1),s(1:m+1),zs(1:m+1),11); dsp];
  qd=qpath(:,1);
%******** disturbance **********************
  if strcmp(other,'disturbance')
    if strcmp(satellite,'smart')
      disturb=[3.5e-4*ones([3,1])];
    elseif strcmp(satellite,'sampex')
      disturb=[.005*ones([3,1])];
    else
      disturb=[.1*ones([3,1])];
    end
%******************noise *******************
  elseif strcmp(other,'noise')
    noise=.001*randn([4,m+1]);
  end
else
  disp(sprintf('ERROR!));
end

%************  main  Loop  *****************
for ii=1:m,

%***** calculate the control effort  *******
  tn=tn+dt;                        % time

  if strcmp(whatSimType,'path')
    qd=qpath(:,ii);
  end

  Ln=Xn(5:7);
  Ld=qd(5:7);


  if strcmp(other,'noise')
    Xn(1:4)=Xn(1:4)+noise(:,ii);
    Xn(1:4)=Xn(1:4)/norm(Xn(1:4));
  end

%****** calc e_d ***
  e_d=(Ln-Ld);
  save_ed(:,ii)=e_d;
  save_norm_ed(ii)=norm(e_d');

%****** calc e_p ***
% error = inv(qd)*q
  qinverse=[-qd(1);-qd(2);-qd(3);qd(4)];
  e_p=qmult(qinverse,Xn(1:4));

  save_ep(:,ii)=e_p;
  save_norm_ep(ii)=norm(e_p(1:3));
  save_itae(ii)=tn*save_norm_ep(ii);

%****** calc e_i ***
  e_i = e_i + (e_p(1:3)*dt)/30;
  save_ei(:,ii) = e_i;
  save_norm_ei(ii) = norm(e_i);

  % fuzzy controller
  if strcmp(ControllerType,'fuzzy')
    inputs = [norm(e_p(1:3)') norm(e_i')  norm(e_d')];          %
input into fuzzy

%%%%%%%%%%%% saturation %%%%%%%%%%%%%%%
    if inputs(1)>1.1
      inputs(1)=1.1;
    end
    if inputs(2)>.5
      inputs(2)=.5;
    end
    if inputs(3)>.5
      inputs(3)=.5;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    [fuz] = evalfis(inputs, fismatrix
    kp=fuz(1);
    TD=1;
    save_fuz(:,ii)=fuz';
    un=-50*kp*([0;e_p(1:3)/norm(e_p)] + TD*[0;e_d]);

  % normal PD controller
  elseif strcmp(ControllerType,'pd')
    kp=.5;
    TD=1;
    un=-kp*([0;e_p(1:3)]+[0;TD*e_d]);
```

```
  % sliding mode controller
  elseif strcmp(ControllerType,'slide') | strcmp(Controller-
Type,'fuzslide')
    alpha = .1;
    gamma = .5;

    if strcmp(satellite,'sampex')
      inertiah=[15  0   0
               0  21   0
               0   0  15];

    elseif strcmp(satellite,'maps')
      inertiah=[3.99e2 0   0
               0  3.77e2  0
               0  0    3.77e2];

    elseif strcmp(satellite,'smart')
      inertiah=.9*[5.65e-1 0 0;
                  0 7.19e-1 0;
                  0 0 4.54e-1];
    else
      disp(sprintf('ERROR!'))
    end

    iinertiah=inv(inertiah);

    w=iinertiah*Ln;
    omega = -[0     w(3) -w(2);
            -w(3)  0     w(1);
             w(2) -w(1)   0 ];
    qe = e_p;

    wd = iinertiah*Ld;
    we = w - wd;
    s =  we+alpha*qe(1:3);
    save_s(:,ii) = s;

   q_dot = .5*omega*Xn(1:3)+.5*Xn(4)*w;


  if strcmp(ControllerType,'slide')
    F=(inertiah*s);
  elseif strcmp(ControllerType,'fuzslide')
    for i=1:3,
      inputs = [abs(alpha*e_p(i)) abs(e_d(i))];
      if inputs(1)>1.1     % saturation
        inputs(1)=1.1;
      end
      if inputs(2)>.5
        inputs(2)=.5;
      end
      fuz=evalfis(inputs, fismatrix);
      F(i,1)=2*fuz(1);
    end
    F=inertiah*F;
  end
  save_F(:,ii)=F;

  su = -omega*inertiah*w - gamma*inertiah*s - alpha*iner-
tiah*q_dot - diag(F)*sat(s,.1);
un=[0;su];

  else
    disp(sprintf('ERROR!'))
  end

  % Disturbance input
  if strcmp(other,'disturbance') & ii>300 & ii<600
    dn=disturb;
  else
    dn=[0;0;0];
  end

   %******* saturate output ************
  if strcmp(satellite,'smart')
    un = saturateOutput(un,7.5e-4);
  elseif strcmp(satellite,'sampex')
    un = saturateOutput(un,.01);
  else
    un = saturateOutput(un,.2);
  end

   save_uc(:,ii)=un;          % Store control effort
   save_norm_uc(:,ii)=norm(un);  % Store the norm of the effort

%*******  RK45  ******
   k1x=dt*sam_ctrl(tn,Xn,un,dn,inertia,iinertia);
  k2x=dt*sam_ctrl(tn+.5*dt,Xn+.5*k1x,un,dn, inertia,iinertia);
  k3x=dt*sam_ctrl(tn+.5*dt,Xn+.5*k2x,un,dn, inertia,iinertia);
  k4x=dt*sam_ctrl(tn+dt,Xn +k3x,un,dn,inertia, iinertia);
  Xn= Xn +(1/6)*(k1x + 2*k2x + 2*k3x + k4x );

%******  Feedback  and error terms  ******

  saveXn(:,ii)=Xn;        % save state vector

  t(ii)=tn/60;                % save time
  saveqd(:,ii)=qd;            % save desired
end
%********* end main loop *****************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Draw Graphs        %
%      Printing results      %
%      not shown here        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Euler to Quaternion: e2q.m

This function converts euler angles into quaternions. It was used to calculate the desired path of the satellites to follow.

```
function q=e2q(phi,theta,psi,flag)
%function q=e2q(phi,theta,psi,flag)
%
% This m-file transforms Euler angle to
% quaternions with the following rotations.
% Theta, phi, and psi are in radians.
%
%  The input are:
%    flag = 1 for 1-2-1
%         = 2 for 2-3-2
```

```
%       = 3 for 3-1-3
%       = 4 for 1-3-1
%       = 5 for 2-1-2
%       = 6 for 3-2-3
%       = 7 for 1-2-3
%       = 8 for 2-3-1
%       = 9 for 3-1-2
%       = 10 for 1-3-2
%       = 11 for 2-1-3
%       = 12 for 3-2-1

% John L. Crassidis 4/24/95

t1=phi;t2=theta;t3=psi;

if flag==1 | flag==2 | flag==3 | flag==4 | flag==5 | flag ==6
 q4=cos(t2/2).*cos((t1+t3)/2);
end

if flag==1,
 q1=cos(t2/2).*sin((t1+t3)/2);
 q2=sin(t2/2).*cos((t1-t3)/2);
 q3=sin(t2/2).*sin((t1-t3)/2);
elseif flag==2,
 q1=sin(t2/2).*sin((t1-t3)/2);
 q2=cos(t2/2).*sin((t1+t3)/2);
 q3=sin(t2/2).*cos((t1-t3)/2);
elseif flag==3,
 q1=sin(t2/2).*cos((t1-t3)/2);
 q2=sin(t2/2).*sin((t1-t3)/2);
 q3=cos(t2/2).*sin((t1+t3)/2);
elseif flag==4,
 q1=cos(t2/2).*sin((t1+t3)/2);
 q2=sin(t2/2).*sin((t3-t1)/2);
 q3=sin(t2/2).*cos((t3-t1)/2);
elseif flag==5,
 q1=sin(t2/2).*cos((t3-t1)/2);
 q2=cos(t2/2).*sin((t3+t1)/2);
 q3=sin(t2/2).*sin((t3-t1)/2);
elseif flag==6,
 q1=sin(t2/2).*sin((t3-t1)/2);
 q2=sin(t2/2).*cos((t3-t1)/2);
 q3=cos(t2/2).*sin((t3+t1)/2);
end

if flag==7 | flag==8 | flag==9
 q4=cos(t1/2).*cos(t2/2).*cos(t3/2)-sin(t1/2).*sin(t2/2).*sin(t3/
2);
end
if flag==10 | flag==11 | flag==12
 q4=cos(t1/2).*cos(t2/2).*cos(t3/2)+sin(t1/2).*sin(t2/2).*sin(t3/
2);
end

if flag==7,
 q1=sin(t1/2).*cos(t2/2).*cos(t3/2)+cos(t1/2).*sin(t2/2).*sin(t3/
2);
 q2=cos(t1/2).*sin(t2/2).*cos(t3/2)-sin(t1/2).*cos(t2/2).*sin(t3/
2);
 q3=cos(t1/2).*cos(t2/2).*sin(t3/2)+sin(t1/2).*sin(t2/2).*cos(t3/
2);
elseif flag==8,
 q1=cos(t1/2).*cos(t2/2).*sin(t3/2)+sin(t1/2).*sin(t2/2).*cos(t3/
2);
 q2=sin(t1/2).*cos(t2/2).*cos(t3/2)+cos(t1/2).*sin(t2/2).*sin(t3/
2);
 q3=cos(t1/2).*sin(t2/2).*cos(t3/2)-sin(t1/2).*cos(t2/2).*sin(t3/
2);
elseif flag==9,
 q1=cos(t1/2).*sin(t2/2).*cos(t3/2)-sin(t1/2).*cos(t2/2).*sin(t3/
2);
 q2=cos(t1/2).*cos(t2/2).*sin(t3/2)+sin(t1/2).*sin(t2/2).*cos(t3/
2);
 q3=sin(t1/2).*cos(t2/2).*cos(t3/2)+cos(t1/2).*sin(t2/2).*sin(t3/
2);
elseif flag==10,
 q1=sin(t1/2).*cos(t2/2).*cos(t3/2)-cos(t1/2).*sin(t2/2).*sin(t3/
2);
 q2=cos(t1/2).*cos(t2/2).*sin(t3/2)-sin(t1/2).*sin(t2/2).*cos(t3/
2);
 q3=cos(t1/2).*sin(t2/2).*cos(t3/2)+sin(t1/2).*cos(t2/2).*sin(t3/
2);
elseif flag==11,
 q1=cos(t1/2).*sin(t2/2).*cos(t3/2)+sin(t1/2).*cos(t2/2).*sin(t3/
2);
 q2=sin(t1/2).*cos(t2/2).*cos(t3/2)-cos(t1/2).*sin(t2/2).*sin(t3/
2);
 q3=cos(t1/2).*cos(t2/2).*sin(t3/2)-sin(t1/2).*sin(t2/2).*cos(t3/
2);
elseif flag==12,
 q1=cos(t1/2).*cos(t2/2).*sin(t3/2)-sin(t1/2).*sin(t2/2).*cos(t3/
2);
 q2=cos(t1/2).*sin(t2/2).*cos(t3/2)+sin(t1/2).*cos(t2/2).*sin(t3/
2);
 q3=sin(t1/2).*cos(t2/2).*cos(t3/2)-cos(t1/2).*sin(t2/2).*sin(t3/
2);
end

q=[q1; q2; q3; q4];
```

Quaternion Multiplication: qmult.m

This function performed quaternion multiplication.

```
function out=qmult(q1,q2)

out=[ q1(4) -q1(3)  q1(2) q1(1);
      q1(3)  q1(4) -q1(1) q1(2);
     -q1(2)  q1(1)  q1(4) q1(3);
     -q1(1) -q1(2) -q1(3) q1(4)]*q2;
```

Equations of Motion: sat_cntl.m

This function contained the equations of motion, used by the RK45 section of the simulation to calculate the new states.

```
%
%           This program simulates the Any satellite
%
%           Syntax [sys]=thing1(t,x,u,dn,inertia,iinertia);
%
% where       u(:,1) = the reaction wheel input
%           u(:,2) = the L1 angular momentum input
%           u(:,3) = the L2 angular momentum input
%           u(:,4) = the L3 angular momentum input
%           iinertia = the inverse of the inertia matrix
%

function [sys]=sam_ctrl(t,x,u,dn,inertia,iinertia)


d=zeros(7,1);

w=iinertia*([x(5) x(6) x(7)]'-[0 0.0041488*u(1)*0 0]');

sys(1,1)=0.5*(w(3)*x(2)-w(2)*x(3)+w(1)*x(4));
sys(2,1)=0.5*(-w(3)*x(1)+w(1)*x(3)+w(2)*x(4));
sys(3,1)=0.5*(w(2)*x(1)-w(1)*x(2)+w(3)*x(4));
sys(4,1)=0.5*(-w(1)*x(1)-w(2)*x(2)-w(3)*x(3));

sys(5,1)=w(3)*x(6)-w(2)*x(7)+u(2) + dn(1);
sys(6,1)=-w(3)*x(5)+w(1)*x(7)+u(3) + dn(2);
sys(7,1)=w(2)*x(5)-w(1)*x(6)+u(4) + dn(3);
```

Sliding Mode Saturation Function: sat.m

This function was the saturation function in the sliding mode controllers.

```
function out=sat(in,boundry)

for i=1:length(in),
  if(abs(in(i))<(boundry))
    out(i,1) = in(i);
  else
    out(i,1) = sign(in(i))*boundry;
  end
end
```

Control Effort Saturation: saturate-Output.m

This function saturated the output of the controller to the proper level since each satellite could produce different control effort levels.

```
function uout = saturateOutput(uin,sat)

howMany = length(uin);

for i=1:howMany,
  if uin(i)>sat
    uin(i) = sat;
  elseif uin(i)<-sat
    uin(i) = -sat;
  end
end

uout = uin;
```

APPENDIX E: OPEN PUBLICATION LICENSE Draft v1.0, 8 June 1999

I. REQUIREMENTS ON BOTH UNMODIFIED AND MODIFIED VERSIONS

The Open Publication works may be reproduced and distributed in whole or in part, in any medium physical or electronic, provided that the terms of this license are adhered to, and that this license or an incorporation of it by reference (with any options elected by the author(s) and/or publisher) is displayed in the reproduction.

Proper form for an incorporation by reference is as follows:

> Copyright (c) <year> by <author's name or designee>. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, vX.Y or later (the latest version is presently available at http://www.opencontent.org/openpub/).

The reference must be immediately followed with any options elected by the author(s) and/or publisher of the document (see section VI).

Commercial redistribution of Open Publication-licensed material is permitted.

Any publication in standard (paper) book form shall require the citation of the original publisher and author. The publisher and author's names shall appear on all outer surfaces of the book. On all outer surfaces of the book the original publisher's name shall be as large as the title of the work and cited as possessive with respect to the title.

II. COPYRIGHT

The copyright to each Open Publication is owned by its author(s) or designee.

III. SCOPE OF LICENSE

The following license terms apply to all Open Publication works, unless otherwise explicitly stated in the document.

Mere aggregation of Open Publication works or a portion of an Open Publication work with other works or programs on the same media shall not cause this license to apply to those other works. The aggregate work shall contain a notice specifying the inclusion of the Open Publication material and appropriate copyright notice.

SEVERABILITY. If any part of this license is found to be unenforceable in any jurisdiction, the remaining portions of the license remain in force.

NO WARRANTY. Open Publication works are licensed and provided "as is" without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement.

IV. REQUIREMENTS ON MODIFIED WORKS

All modified versions of documents covered by this license, including translations, anthologies, compilations and partial documents, must meet the following requirements:

1. The modified version must be labeled as such.
2. The person making the modifications must be identified and the modifications dated.
3. Acknowledgement of the original author and publisher if applicable must be retained according to normal academic citation practices.
4. The location of the original unmodified document must be identified.
5. The original author's (or authors') name(s) may not be used to assert or imply endorsement of the resulting document without the original author's (or authors') permission.

V. GOOD-PRACTICE RECOMMENDATIONS

In addition to the requirements of this license, it is requested from and strongly recommended of redistributors that:

1. If you are distributing Open Publication works on hardcopy or CD-ROM, you provide email notification to the authors of your intent to redistribute at least thirty days before your manuscript or media freeze, to give the authors time to provide updated documents. This notification should describe modifications, if any, made to the document.
2. All substantive modifications (including deletions) be either clearly marked up in the document or else described in an attachment to the document.
3. Finally, while it is not mandatory under this license, it is considered good form to offer a free copy of any hardcopy and CD-ROM expression of an Open Publication-licensed work to its author(s).

VI. LICENSE OPTIONS

The author(s) and/or publisher of an Open Publication-licensed document may elect certain options by appending language to the reference to or copy of the license. These options are considered part of the license instance and must be included with the license (or its incorporation by reference) in derived works.

A. To prohibit distribution of substantively modified versions without the explicit permission of the author(s). "Substantive modification" is defined as a change to the semantic content of the document, and excludes mere changes in format or typographical corrections.

To accomplish this, add the phrase `Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.' to the license reference or copy.

B. To prohibit any publication of this work or derivative works in whole or in part in standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

To accomplish this, add the phrase 'Distribution of the work or derivative of the work in any standard (paper) book form is prohibited unless prior permission is obtained from the copyright holder.' to the license reference or copy.

OPEN PUBLICATION POLICY APPENDIX:

(This is not considered part of the license.)

Open Publication works are available in source format via the Open Publication home page at http://works.opencontent.org/.

Open Publication authors who want to include their own license on Open Publication works may do so, as long as their terms are not more restrictive than the Open Publication license.

If you have questions about the Open Publication License, please contact TBD, and/or the Open Publication Authors' List at opal@opencontent.org, via email.