3-24-2016

Evaluation of Verification Approaches Applied to a Nonlinear Control System

Kerianne H. Gross

EVALUATION OF VERIFICATION APPROACHES APPLIED TO NONLINEAR SYSTEM CONTROL

THESIS

Kerianne Hobbs Gross

AFIT-ENY-MS-16-M-214

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENY-MS-16-M-214

EVALUATION OF VERIFICATION APPROACHES APPLIED TO NONLINEAR SYSTEM CONTROL

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Astronautical Engineering

Kerianne Hobbs Gross, B.S.A.E.

March 2016

AFIT-ENY-MS-16-M-214

EVALUATION OF VERIFICATION APPROACHES APPLIED TO NONLINEAR SYSTEM CONTROL

THESIS

Kerianne Hobbs Gross, B.S.A.E.

Committee Membership:

Eric D. Swenson, PhD
Chairman

Fred Leve, PhD
Member

Darryl Ahner, PhD
Member

James Overholt, PhD
Member

Matthew Clark, MS
Member

Abstract

As the demand for increasingly complex and autonomous systems grows, designers may consider computational and artificial intelligence methods for more advanced, reactive control. While the performance gained by such increasingly intelligent systems may be superior to traditional control techniques, the lack of transparency in the systems and opportunity for emergent behavior limits their application in the field. New verification and validation methods must be developed to ensure the output of such controllers do not put the system or any people interacting with it in danger. This challenge was highlighted by the former Air Force Chief Scientist in his 2010 Technology Horizons Report, stating "It is possible to develop systems having high levels of autonomy, but it is the lack of suitable [verification and validation] (V&V) methods that prevents all but relatively low levels of autonomy from being certified for use."

Exhaustive test of complex and autonomous systems is intractable and cost prohibitive; however, design analysis techniques such as formal methods and design methodologies such as Run Time Assurance (RTA) could provide supplementary certification evidence early in system design. Incorporating formal methods analysis throughout the system design process provides a means to identify faults as they are introduced to drastically reduce the overall system development cost. RTA is a proposed methodology to allow unproven autonomous controllers to perform within a predetermined envelope of acceptable behavior, allowing the burden of testing to be spread across the entire system lifecycle. The performance of the system is monitored by a decision module that includes a set of acceptable conditions and behaviors. If the system operates outside of pre-determined conditions or any of the acceptable

behavior ranges are violated, the decision module switches control from the unproven autonomous controller to an appropriate recovery controller that may be fully proven using conventional design time verification methods, coupled with formal methods or other alternative verification techniques. Supplementing traditional verification tools with enhanced alternative tools such as formal methods or online verification methods as RTA will enable verification of more complex nonlinear problems that the United States Air Force (USAF) is facing.

In this research, a nonlinear model was chosen because it stresses the formal methods analysis tools to expose areas where they could be enhanced. A 6U CubeSat Attitude Control Subsystem (ACS) is used as a challenge problem to evaluate the application of non-traditional verification methodologies such as formal methods and run time assurance architectures in conjunction with more traditional verification techniques. Thirteen hypothetical requirements are presented and formally defined. Strengths and weaknesses of the verification techniques are exposed in order to recommend capability expansions for further development. In analyzing the application of different formal methods tools, a new approach to verification was created to provide evidence of requirement satisfaction that leverages the capabilities of formal methods in conjunction with traditional verification techniques such as simulation cases, space filling experimental design simulation, and mathematical feasibility analysis.

To Mom

I wouldn't be where I am today without your loving support and encouragement.

Acknowledgements

First, I would like to thank my advisor Dr. Eric Swenson, for his mentoring throughout my time at AFIT and the foundational spacecraft attitude dynamics and control knowledge I gained in his classes. Thank you for working with me as a civilian student to create the perfect set of course sequences for my research interests with an odd schedule and a research agenda of my own. I really appreciate your support on what became a very cross-disciplinary research topic and all your feedback on the numerous thesis drafts and three conference papers I presented in the last few months.

I'd like to thank my thesis committee members Dr. Eric Swenson, Dr. Darryl Ahner, Dr. Jim Overholt, Dr. Fred Leve, and Mr. Matt Clark for their support during my research and evaluating the final product. I'd also like to thank Dr. Richard Cobb for the foundational controls knowledge I gained from him in his classes and extra hours with questions in his office, and Col. Jeremy Agte for his support of the related research paper in optimal control I published with Lt. Ryan Patrick.

My master's degree would not have been possible without the help of Mr. Howard Emsley and Dr. Andy Sparks whose support as my Branch Chief and Division Chief was critical when I started my graduate studies, Dr. Jeff Tromp whose support as my Branch Chief was vitally important over the last year when I was balancing a very challenging course load with my full time responsibilities at AFRL, and Mr. Matt Clark whose support as my Team Lead and now Branch Chief has provided key technical direction to my research. I really appreciated all of your support and the flexibility you gave me to align my research at AFIT with my work at AFRL and to schedule my work around my courses during the day. Mr. Matt Clark, Dr. Andy Sparks, Dr. Jim Overholt, Mr. Howard Emsley, Mr. Joe Nalepka and Dr. Elizabeth Beecher have all provided valuable mentorship and perspectives over the last few years that have helped guide me down a path to completing a Master's Degree in

Astronautical Engineering, and pursuing my PhD next fall.

I'd also like to thank my classmates Capt. Eric Basset and 1st Lt. Ryan Patrick for sharing in a spacecraft attitude control related thesis research journey, and Matt Lippert for helping me understand the C code for the CubeSat Testbed while balancing all his other projects. I'd like to thank Jon Hoffman, Aaron Fifarek, Lucas Wagner, and Mike Whalen for helping me learn what formal methods are and how to use many of the formal methods tools I applied in this research. I'd like to thank Mr. Tyler Vick for helping me set up my batch simulation in about a tenth of the time it would have taken me on my own, Dr. Darryl Ahner for his advice and help to generate a space filling experimental design for batch simulation runs and analyze the results, and Dr. Stan Bak for introducing me to hybrid systems reachability.

Finally, I'd like to thank my husband Justin Gross for proofreading my technical conference papers related to this research, keeping me fueled nearly every night with his delicious restaurant quality dinners, and sacrificing so much of our time together while I pursued this degree.

Kerianne Hobbs Gross

Table of Contents

List of Figures

List of Symbols

List of Abbreviations

EVALUATION OF VERIFICATION APPROACHES APPLIED TO NONLINEAR SYSTEM CONTROL

I. Introduction

1.1 Motivation

On September 16, 2014, Major General Thomas Masiello, commander of the Air Force Research Laboratory (AFRL), announced autonomy as one of AFRL's three "Game Changing Technologies."[67] The introduction of unmanned systems in modern warfare has significantly improved warfighter capabilities by providing greater access and intelligence in hazardous environments. Autonomous control systems that sense their environment, compensate for system failures, and respond in spite of uncertainty by rapidly deciding and acting on their own are highly desired across many disciplines and organizations. The definition of autonomy is heavily debated;[73] however, most agree that a spectrum exists between automation and increasingly higher levels of autonomy and machine "self consciousness."

Increasingly automatic and autonomous control systems promise better performance in dynamic environments; however, traditional exhaustive verification and validation (V&V) techniques are insufficient and cost prohibitive to evaluate the software underlying these complex systems with extremely large or infinite state spaces. The cost to develop and test new complex systems for the Air Force is on an exponential curve that inspired Norman Augustine, former Chairman of Lockheed Martin Corporation, to famously predict that "in the year 2054, the entire defense budget will purchase just one aircraft."[15] New approaches to V&V must be developed to deal

with new highly complex systems. The former Air Force Chief Scientist highlighted the need for alternative verification and validation methods in his 2010 Technology Horizons Report, stating "It is possible to develop systems having high levels of autonomy, but it is the lack of suitable V&V methods that prevents all but relatively low levels of autonomy from being certified for use."[39]

While certification of autonomous systems encompasses both hardware and software V&V, the software that defines the autonomous decision making capability arguably presents the largest challenge. Software certification is challenging even without the incorporation of autonomy. The Systems Engineering "V" with descriptions of when software faults are introduced, found, and the relative cost to fix the fault is shown in Figure 1.

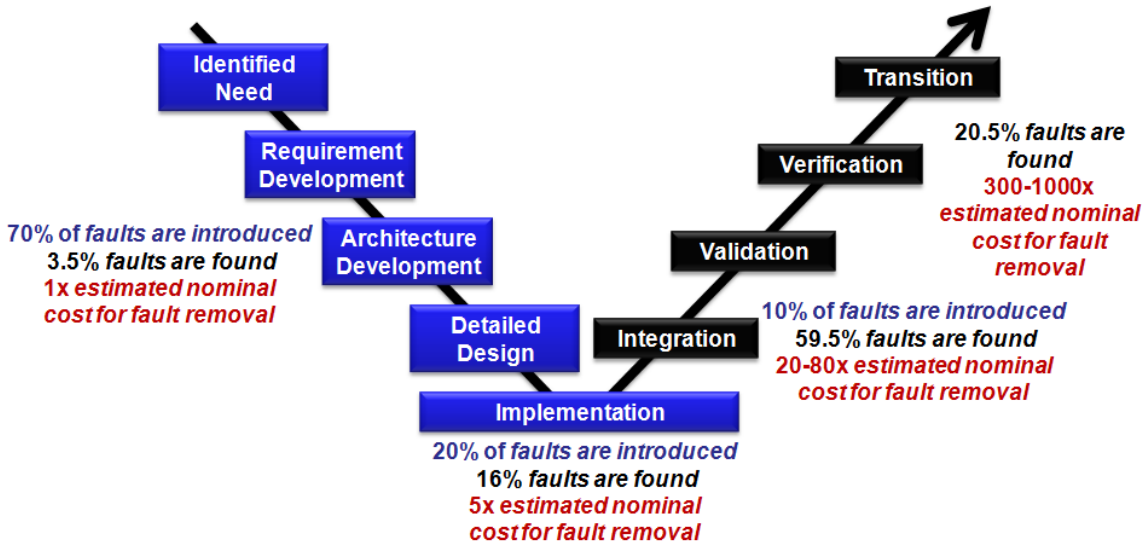

Figure 1. Introduction, Identification, and Cost to Correct Software Faults [47][10][32]

Feiler [47] estimates that 70% of software cost is rework and certification.[47] In addition, 70% of software faults are introduced early in the development cycle, but only 3.5% of those faults are found early; over 80% of faults are found after unit test when the estimated nominal cost for fault removal is 20-1000 times higher

than at early design stages.[10] Investing in V&V earlier in the design enables earlier error detection when the costs associated with a design change are much lower than later in the design process. In a 2014 report [32], the Verification and Validation of Complex and Autonomous Systems (VVCAS) Team, which the author of this thesis is a member of, at AFRL's Aerospace Systems Directorate described the need to expand the current V&V paradigm and the benefits that may be realized from expansion.

In traditional systems engineering practices, V&V is conducted after the system design is complete via modeling, simulation, and test. As software complexity increases, depending on traditional V&V practices alone becomes intractable and infeasible. A combination of run time assurance (RTA) design principles, correct-by-construction code synthesis, and formal methods-based analytical proofs could supplement and reduce reliance on traditional simulation and flight test evidence. The VVCAS team envisions a systems engineering practice that supplements traditional modeling, simulation, test, and evaluation techniques with analytical V&V evidence generated throughout the system engineering process and run time monitoring and bounding of system performance. Following publication of the 2014 report [32], this vision was also adopted by the Assistant Secretary of Defense/Research and Engineering (ASD/R&E) endorsing the Department of Defense (DoD) Autonomy Test and Evaluation, Verification and Validation (TEVV) Investment Strategy.[11]

1.2 Problem Statement

The objective of this research is to evaluate formal methods analysis (FMA) approaches and RTA in the context of solving the exhaustive testing problem. RTA architectures and FMA are not widely used and are still areas of active research. Applying a compositional RTA architecture backed by FMA during the requirements, architecture, and modeling design phases could provide reusable verification evidence

that a controller implementation meets a set of requirements. A formally verified RTA architecture provides a foundation for the insertion of a nondeterministic or intelligent controller that employs artificial intelligence and learning. The online verification provided by an RTA controller at runtime is predicated on offline verification of its verified backup controller and decision module subcomponents. The backup controller subcomponent is verified to provide a safe alternative control solution if an intelligent controller fails, while the decision module subcomponent is verified to switch to the backup controller if the unverified intelligent controller approaches a boundary that would violate one of the system's safety properties. To generate a verified backup controller, formal methods analysis techniques are applied to a non-linear proportional, integral, derivative (PID) control system to supplement traditional simulation techniques. This research will provide insight into the capabilities and gaps of RTA architectures and formal methods V&V techniques when applied to the control design of a nonlinear system. A 6U CubeSat Attitude Control Subsystem (ACS) is selected as the nonlinear control challenge problem.

1.3 Research Focus

The primary objective of this research is evaluate the combination of traditional simulation analysis with non-traditional verification techniques to generate verification evidence that a nonlinear controller will meet performance requirements and never exceed safety constraints. The combination of traditional verification methods, including feasibility analysis (FA) to see if a requirement is possible, a simulation case (SIM) and space filling experimental design simulation (SFEDS), and non-traditional verification methods, including RTA and FMA, conducted in this research has not previously been applied to a nonlinear control system design to the best of the author's knowledge. The strengths and limitations of each of these analysis techniques will

be evaluated and recommendations will be made based on the results for appropriate use of each technique in control systems analysis. In addition, recommendations will be made for generating appropriate requirements to enable the use of the variety of techniques explored in this research.

1.4 Methodology

A combination of RTA, FMA, SIM, and SFEDS are applied to a nonlinear control system in the following steps:

1. The requirements and derived requirements resulting from design decisions (such as the type of actuator) are generated for the control system, and specific safety requirements are identified. These requirements are formalized by assigning precise mathematical and logical definitions.

2. FMA is performed during requirements, architecture, and modeling phases of the nonlinear PID controller design to prove that the PID controller implementation will not violate safety requirements. The Specification and Analysis of Requirements (SpeAR) framework is used to apply model checking and limited theorem proving analysies to the controller requirements to show that the derived requirements of the system implementation satisfy the safety requirements of the system. The Assume Guarantee Reasoning Environment (AGREE) annex is used to complete assume-guarantee contract analysis on the architecture of the controller written in the Architecture Analysis & Design Language (AADL) to show that the architecture implementation satisfies safety requirements. Simulink Design Verifier (SLDV) is used to apply FMA to the Simulink model of the controller to prove that the modeled implementation of the controller satisfies the safety requirements.

3. SIM is applied to the nonlinear PID control system to show satisfaction of the performance requirements.

4. SIM is expanded through the selection of 257 test points and application of SFEDS to analyze under what conditions the performance requirements are held by the nonlinear PID control system over a set of initial and final system states.

5. The PID controller is inserted as the verified controller in a RTA controller implementation. The RTA controller monitors the output of a "black box," "gray box," or "white box" unverified controller with a decision module component that switches to the formally verified PID controller output if any performance boundaries are expected to be violated within the next timestep. FMA is applied to the verified controller and decision module outputs of the RTA controller in requirements, architecture, and model phases using SpeAR, AGREE, and SLDV to prove safety requirements are not violated. FMA is then applied to the composition of the RTA controller as a system with three subcomponents in the requirements and architecture with SpeAR and AGREE to show that the combination of the three subcomponents will never violate the controller safety requirements.

These steps are visualized in Figure 2, where the requirements, implementation, and verification portions of the research are organized in vertical column-like orientations identified by three brackets at the top of the figure. The different verification portions of the research are further identified as "traditional" versus "non-traditional" using brackets on the right side of Figure 2. The numbers in circles indicate portions of the diagram identified earlier in this section. For instance, step 5 includes the development of the decision module, composition of the RTA, and FMA of the RTA

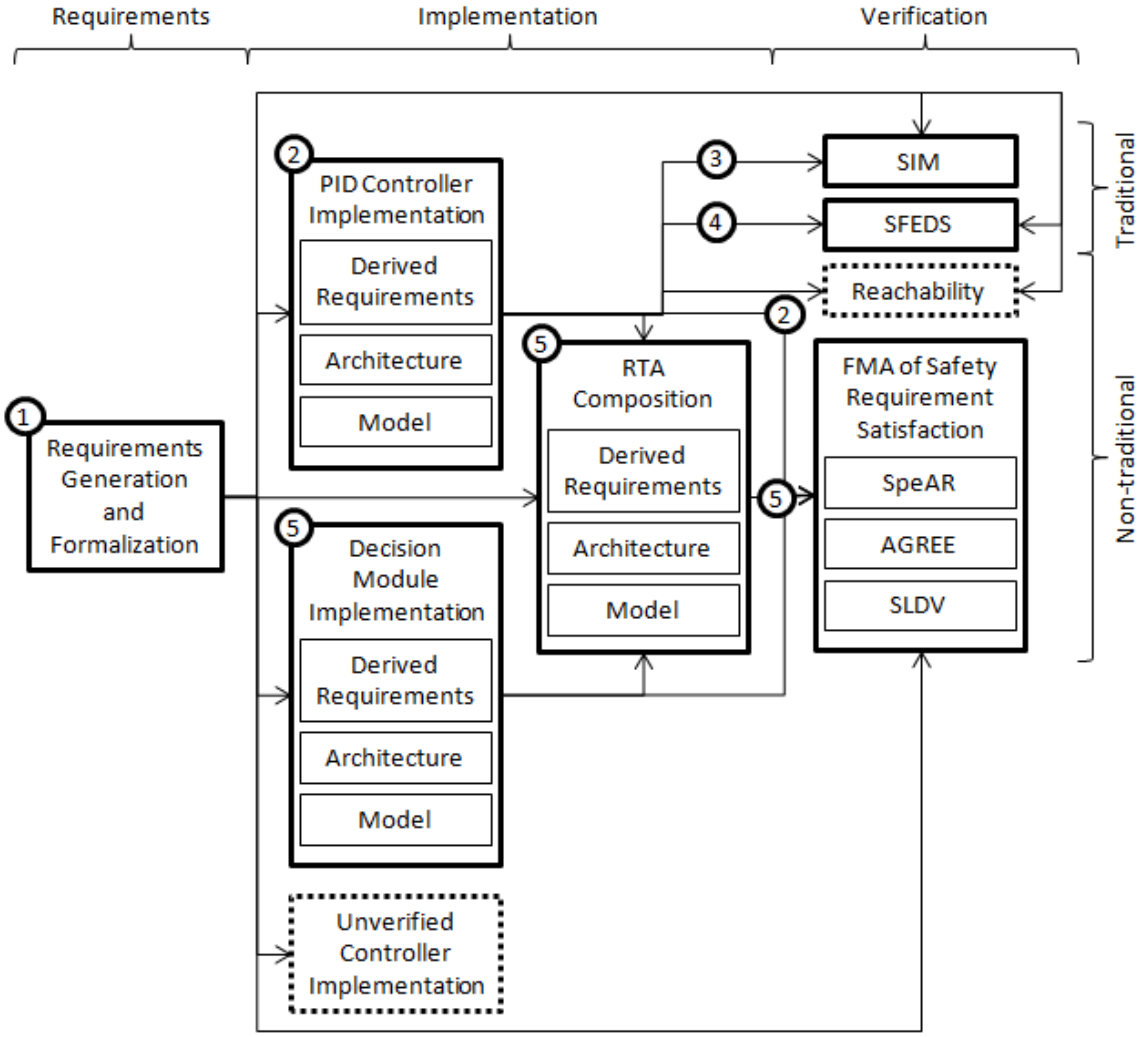

Figure 2. Development and Analysis Flow Diagram

and its three subcomponents. The two boxes with dotted outlines are left for future work, although a placeholder for an intelligent unverified controller is included in all the models in step 5.

A summary of the models developed as a part of this research and the types of analysis conducted on them is shown in Figure 3.

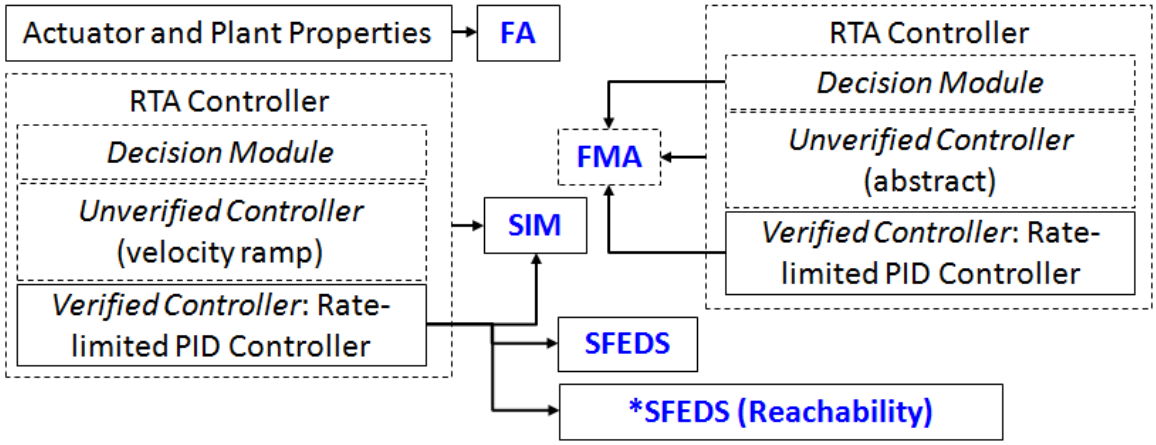

Figure 3. Thesis Research Models and Analysis Techniques

As seen in Figure 3, three models are used in the analysis: a model of the actuator and plant properties only, an RTA controller with an abstract unverified controller component, and an RTA controller with a velocity ramp substituted for the unverified controller. Also seen in Figure 3, five types of analysis techniques are conducted. Arrows from the models to the analysis type are used to clarify what portions of the model are analyzed with the technique, where arrows from the RTA controller model boxes indicate that the RTA controller and all of its subcomponents are analyzed using that technique. Traditional models and analyses are contained in solid boxes while nontraditional models and analyses are contained in dotted line boxes.

A 6U CubeSat ACS is selected as the nonlinear controller challenge problem to evaluate the non-traditional V&V tools and techniques applied in this research. The complex control algorithms and equations of motion that describe the 6U CubeSat

ACS challenge the limits of the non-traditional verification tools and provide a robust example on which to evaluate the feasibility of the approach and identify gaps in tool technique capabilities, as well as appropriateness of the tools for nonlinear control system analysis.

1.5 Assumptions and Limitations

The RTA, FMA, SIM, and SFEDS tools and techniques applied to the analysis of the nonlinear controller do not provide complete coverage of the design. A nonlinear control design was selected because it is expected to be on the edge of the analysis capabilities of the FMA and RTA tools and techniques used in this research. This research is limited to a theoretical and simulated spacecraft attitude control system. There are additional limitations in the physical modeling of the system that will be described in Section 4.4.3.

1.6 Scope and Contribution

The scope of this research is verification of the early design stages of a theoretical RTA controller design. The RTA controller's verified controller and decision module subcomponents and the composition of the RTA are analyzed with FMA to prove that safety requirements of the system are never violated. In addition, a nonlinear PID controller is developed and its performance is analyzed with SIM and SFEDS to understand under what conditions performance requirements may be violated.

The contributions of this work are:

1. The first application and evaluation of SpeAR, AGREE, and SLDV to provide traceable verification evidence for a nonlinear system control in the requirements, architecture and modeling design phases;

2. An exploration of how to formally describe different classes of requirements typically associated with control systems design;

3. Recommendations for the appropriate use of RTA, FMA, SIM, and SFEDS in control system analysis;

4. Recommendations for requirements specification best practices to enable FMA, SIM, and SFEDS;

5. Recommendations for improvements in FMA tools that would allow more analysis coverage of control system design space; and

6. Development of a systematic methodology to combine RTA, FMA, SIM, and SFEDS in a modular design that enables reuse of verification evidence from component requirements, architectures and models in the development of new and modified systems.

1.7 Preview

Chapter I provided the motivation, problem statement, research focus, methodology, assumptions and limitations, and scope of the research. Chapter II provides background information on cyber-physical systems theory that could be used to describe most modern control systems, requirements specification best practices, FMA techniques and tools, examples of spacecraft failures that could have been prevented had formal methods been applied, a description and history of RTA, and background on SFEDS. Chapter III describes AFIT's CubeSat Testbed and ACS components, as well as mathematical descriptions of the underlying spacecraft attitude dynamics for the challenge problem. Chapter IV introduces and explores how to formally describes the requirements set used in the verification of the nonlinear control design, presents the architecture and model of the PID and RTA controller designs, and describes the

set up of the FMA, SIM, and SFEDS used to determine requirement satisfaction. Chapter V presents the results from the FMA, SIM, and SFEDS. Chapter VI provides conclusions from the research, discusses possible extensions and future work, discusses gaps in toolset capabilities, and makes recommendations for improvements to the analysis toolsets.

II. Specification and Verification Techniques

Chapter II provides background on specification and analysis techniques that are used in this research. First, an introduction to cyber-physical systems theory is presented with concepts such as component composition that are used to design models of the nonlinear control system and its environment, finite state machines that are used as part of the FMA, determinism and nondetermism that are used to design the SFEDS analysis, and requirements categories that are used to describe each of the requirements introduced in the next chapter. In addition, characteristics of good requirements are presented that are used to develop and describe the requirements in the next chapter. Requirements verification is defined and several methods are presented to show requirement satisfaction, including all of the methods that are implemented in this research. The foundational principles behind formal methods such as propositional logic, formal languages and FMA analysis capabilities and limitations are described. The FMA analysis tools SpeAR, AGREE, and SLDV used in this research are introduced. The importance of FMA as a non-traditional analysis technique is described by providing examples of catastrophic spacecraft failures that occurred because of design faults that can now be caught automatically with formal methods. The history and application of RTA is described, which is used later in this research to develop a control architecture that can facilitate an intelligent controller within a construct that is proven to never violate safety properties. SFEDS is described, which is applied to the design to verify many of the requirements that cannot be proven using formal methods.

2.1 Cyber-Physical Systems

The spacecraft attitude control design used as a challenge problem for verification and validation techniques is an example of a cyber-physical system. The field of cyber-physical systems research emerged in the early 2000s when processing capabilities, wireless communication, and sensors had matured enough technologically and became more affordable for widespread use. Rajeev Alur states that a cyber-physical system "consists of a collection of computing devices communicating with one another and interacting with the physical world via sensors and actuators in a feedback loop."[14] This description fits most modern control systems.

Cyber-physical systems have five key features: Reactive computation, concurrency, feedback control, real time computation, and safety critical applications.[14] Reactive computation describes a cyber-physical system's interaction with the outside world in terms of inputs and outputs. These systems compute an output based on an input signal using a specific function. Concurrency describes the ability of these systems to conduct multiple threads of computation in the form of components or processes simultaneously, in contrast to sequential systems which execute a sequence of instructions one at a time. Concurrent systems may be synchronous where all computations progress in fixed frames coordinated by the system's clock, or asynchronous where computations may be executed at independent time intervals. Feedback control refers to the discipline of control systems engineering. While traditional control theory focuses on designs for continuous-time systems, cyber-physical systems consist of discrete software executing concurrent computations that interacting with a continuously evolving physical environment. In this way, cyber-physical systems can also be described as hybrid systems because they contain a mix of discrete and continuous dynamics. Real time computation refers to the study of timing delays, time-depended coordination protocols, and resource-allocation strategies to

meet requirements and ensure predictability. Safety-critical applications are those where performance and development cost may be prioritized lower than the safety of the system. Cyber-physical systems theory is relevant to this research because the nonlinear control system in this design features reactive computation, concurrency, and feedback control, and could be considered a safety critical application in terms of safety of the hardware components from damage. This research does not investigate any real time computation concerns.

2.1.1 Synchronous Cyber-Physical Systems Models

Synchronous cyber-physical system models are comprised of discrete components that conduct computations in a sequence of frames. In each frame, components read their input(s), compute output(s) based on the input(s) and update the components internal state(s). The synchrony hypothesis is the assumption that the system is ready to processes new inputs when external inputs change and these external inputs do not change during a computational frame, is based on an assumption that all the computations and communications required to determine variable values occurs instantaneously.[14] This assumption simplifies designs and allows designers better predictability. Synchronous models are comprised of functional and reactive components. Functional components compute an output when supplied by inputs and reactive components interact with other components and maintain and internal state. The system model used in this research is assumed to be a synchronous cyber-physical system, and a unit delay between the environment and controller subsystems segments model computation into discrete frames. In this research, the functional component is the controller, which contains the control software, and the reactive component is the environment, which contains the actuators and plant.

Table 1. Variable and Expression Types[14]

Type	Description
nat	natural number
int	integer
real	real number
bool	boolean value in set of 0,1
enumerated	finite number of symbolic constants in a set x_1,x_2, ... ,x_n

Table 2. Logical Operators[14]

Symbol	Name	Evaluation	Description
$\neg$	negation	$\neg e$ is 1 when e is 0	"not"
$\wedge$	conjunction	$e_1 \wedge e_2$ is 1 when e_1 and e_2 are both 1	"and"
$\vee$	disjunction	$e_1 \vee e_2$ is 1 when at least e_1 or e_2 are 1	"or"
$\rightarrow$	implication	$e_1 \rightarrow e_2$ is 1 when e_1 is 0 or e_2 is 1	"implies"

2.1.2 Components

Alur describes components are in terms of variables, valuations, expressions, inputs, outputs, and states, which are all an integral part of this research. Variables and expressions are assigned types, described in Table 1. A valuation is an assignment of a value to a variable consistent with the variable type. Expressions e may be numerical when assigned a type of nat, int, or real and are constructed with constants; primitive operations, such as addition and multiplication; comparison operators, such as $\leq$; and appropriately-typed variables. Expressions may also be Boolean and constructed with logical operators in addition to the components of numerical expressions. These logical operators are summarized in Table 2. All of the requirements analyzed in this research are formalized as Boolean expressions. In general, a cyber-physical component $C = (I, O, S, Init, React)$ is comprised of sets of typed input variables I, typed output variables O, typed state variables S, initial states $Init$, and a reaction description $React$, where Q_I is the set of all possible inputs to C ($I \in Q_I$), Q_O is the set of all possible outputs from C ($O \in Q_O$), and Q_S is the set of all possible states of C

($S \in Q_S$). In finite-state components, each set of variables is finite. If a component C only has input variables I and output variables O, but no state variables S, such as an if-else statement that combines the inputs to define the output, it is considered a combinational component. Some components may only conduct computations in a frame when triggered by one of the inputs, rather than in every frame. These components are called event-triggered and consist of a subset J of the input variables I that act as triggers.[14]

In this research, the system is broken down into components C as described in Section 4.3.1. Much of the analysis in this research is focused on analysis the typed inputs I and outputs O, such as whether the outputs of the controller components will violate safety constraints. The typed state variables S in this research correspond to four states that describe the orientation of the spacecraft, three states that describe the angular velocity of the spacecraft and four states that describe the angular velocity of the four reaction wheel actuators, as described in Section 3.2.5. Where appropriate, assumptions are used to constrain the set of all possible inputs Q_I and the set of all possible states Q_S to ranges of physically possible to focus FMA and eliminate spurious results. The FMA is used to determine whether the set of all possible outputs Q_O contain values that violate safety constraints. While there are not event-triggered components in this research, future extensions of this research may include event-triggered components that are only used when faults occur or in other specific conditions.

2.1.3 Importance of Initialization

Initial states and how components react to inputs and each state must be specified in cyber-physical systems theory.[14] The initial value of a variable should be assigned when the variable is declared. Some programming languages and styles allow multiple

initial values to be specified using a choose function which randomly assigns a chosen value from the argument set. These argument sets could include a set or a range of values. Initialization is an important concept used in the FMA portion of this research.

2.1.4 Composition of Components

Three main operations are used to compose components of cyber-physical systems to create larger, more capable components: instantiation, parallel composition, and output hiding.[14] In instantiation, multiple instances of the same component may be used with different input and output variable names.[14] In this research, a single model of a reaction wheel is created and multiple instances with unique input and output names are used to create an array model.

Parallel composition combines two or more individual components or instances of components into a single component that captures the synchronous interaction of each individual component running concurrently.[14] In this research, parallel composition is used in the abstraction of sensor components. The sensor and actuator components can be composed in parallel because the output of the actuator is the same as the input to the sensor. If the communication is synchronous, as it is in this research, the actuator model reads its input, produces an output and updates its internal state to record the current value in the same round that the sensor model reads its input from the actuator, computes its output, and updates its internal state. Parallel composition features the following properties: it is commutative, associative, and finite or deterministic when comprised of finite or deterministic components, and the number of states is a product of the number of states of each component.

Output hiding combines two or more individual components so that only one input and output are shown to the outside world and the remaining inputs and outputs

that are not showing are local variables.[14] Output hiding allows for the hierarchical composition of components rather than a single monolithic design that places all components at the same level. In this research, output hiding is used at the system level to examine inputs and outputs of the RTA controller and the environment, while hiding the inputs and outputs of the internal components of the RTA controller and the environment.

2.1.5 State-Space Explosion Problem

One of the consequences of composing states is that the number of states grows exponentially with the number of components. For instance, if you were to compose n instances of a delay component C_1 with n_1 states in a chain that outputs the input of the chain from n frames earlier it would have n_1^n states. The exponential relationship between states and the number of components is sometimes referred to as the state-space explosion problem which has limited the scalability of analysis tools. In this research, composition is used to mitigate state-space explosion by conducting analysis at the component level, and using the results of that analysis to constrain analysis for other components. For example, in this research, a rate-limited PID controller is used as the verified controller component of an RTA controller and is analyzed in isolation to ensure that it meets safety constraints placed on the output. The proof results of the rate-limited PID controller analysis are used as assumptions on the verified controller input to the RTA controllers decision to constrain the range of values used in the FMA analysis. This assumption allows proof to be generated that the decision module component output will not violate safety constraints and prevents identification of a spurious counterexample showing that the decision module violates the safety constraints based on an infeasible input from the rate-limited PID controller.

2.1.6 Finite State Machines

Model-based design often uses finite state machines (FSM) to describe the behavior of the system.[14] Each state variable is a mode of the state machine, and are drawn as circles in state machine diagrams. Arrows are used to show transitions between the modes, with a single sourceless arrow used to describe the initial mode. Extended state machines use additional state variables to augment the modes of the machine. Mode-switches are used to specify the reactions of the extended state machine and are depicted as an edge between two modes with a guard condition and expression to update the variables. Finite and extended state machines are essential concepts in cyber-physical systems as well as the FMA techniques used in this research.

2.1.7 Determinism and Nondeterminism

An algorithm is said to be deterministic if it produces a unique and repeatable result given the same set of inputs. In cyber-physical systems, a component is deterministic if "for a given sequence of inputs, the component has a unique execution producing a unique sequence of outputs. Such deterministic behavior is ensured if the component has a single initial state, and in every state, for a given input, there is exactly one possible reaction."[14] In contrast, nondeterministic algorithms and components may produce different output sequences given the same input sequence. Nondeterministic models are useful for modeling an environment which cannot be completely captured by the model. The notion of non-deterministic algorithms were first introduced in computer science by Robert Floyd in a 1967 when he demonstrated the use of a non-deterministic algorithm to solve the problem of placing 8 queens on a chessboard so that no two were in the same row, same column or diagonal from one another. Two things that set nondeterministic algorithms apart from deterministic

algorithms are their use of some sort of chose function and their branching rather than sequential computational strategy. In the algorithms presented by Floyd, the end of each branch is labeled as a success or failure. Floyd differentiates non-deterministic algorithms from probabilistic, random, and Monte Carlo algorithms, by describing the non-deterministic algorithms as representations of systematic search procedures that are governed by some final goal.[48] The controller and dynamics models used in this research are deterministic.

2.2 Requirements Specification

Wiegers [99] states that quality requirements should be correct, feasible, necessary, prioritized, unambiguous, verifiable, complete, consistent, modifiable and traceable. In this research, formalizing and analyzing requirements helps to ensure that requirements are unambiguous, consistent, and verifiable, and a naming convention described in Section 4.1.3 is used to ensure the requirements are traceable. Ensuring that the requirements are correct, feasible, necessary, prioritized, complete, and modifiable is done with the help of a subject matter expert (SME), and is not the focus of this research.

2.2.1 Requirement Categories

Alur [14] categorizes requirements as safety or liveness. Safety requirements could be thought of as stating "nothing bad ever happens" while liveness requirement could be thought of as stating "something good eventually happens."[14] Safety requirements classify states of the system as safe or unsafe and assert that unsafe states should not be reachable. A safety requirement is often an invariant of the system, which is defined as property of the system which is satisfied in every reachable state.[14] Transition systems may be used to describe how a state variables update in

each frame. Alur defines transitions systems $T = (S, Init, Trans)$ as having a finite set of typed state variables S within a set of all possible states Q_S ($S \in Q_S$), an initial set of states $Init$, and a transition description $Trans$. Reachable states may then be defined as all possible states that can be achieved given an initial state and transition description.[14] A property φ is a Boolean-valued expression e over the state variables of the transition system T. If no reachable state of T violates φ, then φ is considered an invariant property. If at least one reachable state of T satisfies φ then it is called a reachable property, or a liveness requirement.[14] Liveness and invariant requirements are specified using temporal logic.[14]

In this research, the majority of the requirements listed in Section 4.1.1 fall into the categories of liveness or invariants, with the exception of a few requirements that deal with specific control systems characteristics such as settling time, percent overshoot, and rise time. In addition, the system used in this research could be considered a transition system T with 11 states S describing the position and angular velocity of the spacecraft and the angular velocity of the reaction wheels, a range of possible values for each state Q_S, an initial set for each state $Init$, and a transition description $Trans$ provided by the model of the controller and system dynamics.

2.2.2 Requirements Verification

The Defense Acquisition University (DAU) states verification "confirms that a system element meets design-to or build-to specifications. Throughout the systems life cycle, design solutions at all levels of the physical architecture are verified through a cost-effective combination of analysis, examination, demonstration, and testing, all of which can be aided by modeling and simulation."[94] In other words, verification confirms that a system implementation satisfied requirements. As discussed in this definition of verification, there are many ways to show requirement satisfaction. In

this research, the following verification methods are used to show requirement satisfaction:

1. Feasibility Analysis (FA): Show that it is mathematically possible for the system as designed to satisfy a particular requirement.

2. FMA: Use a formal methods analysis technique such as model checking to formally prove that the requirement will never be violated.

3. SIM: Show that the requirements are met in a representative simulation case.

4. SFEDS: Determine whether the requirements are satisfied in multiple simulations across a set of test points and identify regions where the requirement is or is not satisfied.

2.3 Formal Methods

The NASA Langley Formal Method Group defines formal methods as "mathematically rigorous techniques and tools for the specification, design and verification of software and hardware systems."[25] This field of study relies on the foundational belief that mathematically rigorous statements (i.e., well-formed mathematical logic) used to form specifications of a system can be verified through an exhaustive process to determine the correctness of a system. FMA is used in this research to prove that the controller cannot violate safety constraint requirements. In this section, propositional logic, theories, and processes that provide the foundation for FMA are described and the capabilities and limitations of formal methods tools are discussed.

2.3.1 Propositional Logic

Propositional logic forms the basis for formal methods analysis. A proposition is a declarative statement that makes a claim which can be proven true or false.

The proposition should be atomic or indecomposable. These atomic propositions, or "atoms" typically represented symbolically as p, q, or r, can be composed to convey a more complex concept. Natural deduction allows the inference of a conclusion from propositions with a set of proof rules. Applying proof rules to formulas, represented with φ and often referred to as "premises" or "propositions," one may obtain more formulas until a "conclusion," represented by ψ, can be proven.

2.3.2 Temporal Logic

Temporal logic is a methodology to represent and reason about systems in time by discretizing time into time steps.[90] Linear temporal logic (LTL), models sequences of events in a linear progression along a single path; "all" and "exists" operators have no meaning in LTL.[16] Computational tree logic (CTL) models time in a branching tree-like structure, where the branches represent different decisions that may be made; "there exists a path" and "along all paths" may be defined in CTL.[31] Timed Computational Tree Logic (TCTL) is a CTL extension that includes continuous time propositions, and is the "mathematical foundation of timed-automata finite state machines." [31][16] Requirements specified in formal methods tools such as SpeAR or AGREE are translated into temporal logic as a part of the analysis process.

2.3.3 Formal Methods Theories and Processes

A formally designed system can be checked by an algorithmic process (typically with model checkers or theorem provers). Model checking was introduced Clarke and Emerson [33] and Queille and Sifakis [75] in early 1980s as an automatic verification technique for concurrent systems with finite states. Model checking is capable of evaluating if a model satisfies a requirement expressed in temporal logic.[53] The technique utilizes graph theory to build finite models to exhaustively examine and

verify the correctness of decision mode logic and discrete state machines.[34] Model checking tools utilize various techniques to be able to reason on these systems such as binary decision diagrams (BDDs) [13], k-induction [42], and property directed reachability [46]. Model checking produces either a proof or a counter example. A proof is produced if no violation of a formal specification was discovered in the exploration of the FSM. A counterexample is a trace through the FSM that illustrates how the formalized specification could be violated. Model checkers describe the techniques to examine the FSM, not how the individual case is determined to be upheld or is valid. Feasibility at specific state machine locations is leveraged off of the use of satisfiabilty (SAT) solvers or Satisfiability Modulo Theory (SMT) solvers that add modulo theory to SAT capabilities. These tools perform the specific node analysis to provide the model checker results that can be utilized in many of the techniques mentioned previously. Some examples of solvers include Yices1[44], Yices2, CVC4[19], Z3[41], MathSAT[24], and SMTInterpol[28].

2.3.4 Formal Methods Analysis Capabilities and Limitations

Formal methods analysis techniques are capable of generating comprehensive proof that some types of requirements will never be violated. While formal methods techniques are best suited for high level mode logic, they can be adapted to prove properties of control systems such as a requirement that a calculation will never exceed a particular value. These proof capabilities are severely limited for nonlinear systems; however, formal methods are highly capable of analyzing non-functional requirements, as will be discussed in this section.

2.3.4.1 Nonlinear Systems Analysis

Linear functions are proportional to the input variables of the function, such as $f(x) = mx + b$ where m and b are constants and x is the variable. In other words, if a function only contains addition and multiplication by a constant value, it is linear; however, if a function contains two variables multiplied together, creating a polynomial with an order of two or higher, it is nonlinear. The controller and dynamics equations used in this research are nonlinear, meaning that they are products of multiple variables. These nonlinear equations are not well suited for analysis in SpeAR, AGREE, or Simulink Design Verifier because none of these tools currently support nonlinear math; however, SpeAR and AGREE allow users to select which SMT solver to use for the formal methods analysis, and Z3 is an option that provides some support for nonlinear math, so it is possible to expand SpeAR and AGREE capabilities to leverage more of Z3's capabilities in the future. While many of the requirements can be expressed in the tools, very limited analysis can be completed because nonlinear control equations and equations of motion define whether or not the requirements are met; however future versions of SpeAR, AGREE, and Simulink Design Verifier may be able to analyze some low order nonlinear systems.

2.3.4.2 Non-Functional Requirement Analysis

While functional requirements describe the behavior of a system or subsystem, non-functional requirements describe properties such as cost, power, and weight that can often be described mathematically. Analysis was conducted on a sample attitude control subsystem that contained a reaction wheel array, listed as RWA, and a magnetic torque actuator, described as MT. In SpeAR, the high level subsystem requirements are listed as "Properties," while derived requirements that describe the components selected to meet subsystem requirements are listed as "Requirements."

In this example, the cost, mass, and power of the RWA and MT are shown to meet the total subsystem requirements. The analysis shows that all properties are proven valid, as shown in Figure 4. The requirements and properties written in SpeAR are shown in Figure 5.

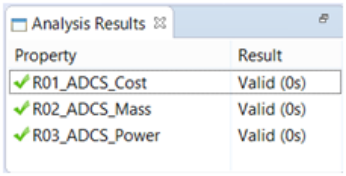

Figure 4. Non-functional Requirement Analysis Results in SpeAR

```
Requirements: //Derived Requirements at the subsystem level

    // The RWA shall not cost more than $210K excluding software development and labor
    R01_ADCS_Cost_RWA = global :: always cost_RWA <= 210000.0;
    // The RWA shall not exceed a mass of 1.1 kg
    R02_ADCS_Mass_RWA = global :: always mass_RWA <= 1.1;
    // The RWA shall not require more than 16 W of power
    R03_ADCS_Power_RWA = global :: always power_RWA <= 16.0;

    // The MT shall not cost more than $210K excluding software development and labor
    R01_ADCS_Cost_MT = global :: always cost_MT <= 210000.0;
    // The MT shall not exceed a mass of 1.1 kg
    R02_ADCS_Mass_MT = global :: always mass_MT <= 1.1;
    // The MT shall not require more than 30 W of power
    R03_ADCS_Power_MT = global :: always power_MT <= 13.0;

    //Total ADCS Subsystem Power
    R01_ADCS_Power_total = global :: always power == power_RWA+power_MT;
    R02_ADCS_Cost_total = global :: always cost == cost_RWA+cost_MT;
    R03_ADCS_Mass_total = global :: always mass == mass_RWA+mass_MT;

Properties: //System Level Requirements
    // The ADCS shall not cost more than $420K excluding software development and labor
    R01_ADCS_Cost = global :: always cost <= 420000.0;
    // The ADCS shall not exceed a mass of 2.3 kg
    R02_ADCS_Mass = global :: always mass<= 2.3;
    // The ADCS shall not require more than 30 W of power
    R03_ADCS_Power = global :: always power <= 30.0;
```

Figure 5. Non-functional Requirements Analysis Example in SpeAR

2.4 Formal Methods Tools

This section provides a description of the three formal methods tools used in this research to analyze the requirements, architecture, and model of the nonlinear controller. A description of how the tools can be used together in an incremental design process is presented.

2.4.1 Requirements Analysis with SpeAR

There are currently two competing approaches to generating formal requirements: parsing natural language requirements documents and manually constructing formal requirements using patterns and templates. The first approach, language parsing, is the subject of research programs such as Automatic Requirements Specification Extraction from Natural Language (ARSENAL), which uses semantic parsing and refinement techniques to generate a formal model in LTL.[50] The advantage of the parsing approach is that it does not require SMEs to learn a formal specification language, and existing requirements documents can be automatically parsed rather than regenerated by hand in a specification language. The disadvantage of parsing is that it requires that the natural language requirements document have all the content necessary to construct a formal model in LTL. The second approach, manual construction of formal requirements specifications, is the subject of a the research tool underdevelopment by Rockwell Collins and AFRL called SpeAR, which allows subject matter experts (SMEs) to write system requirements in peer-reviewed specification templates [45] that facilitate formal analysis. Manually constructing formal requirements using patterns and templates may require a SME to learn these patterns, however, this method has two major advantages. First, expressing requirements in a formalized notation gives them a precise, mathematical meaning, and requires SMEs to unambiguously state what the true meaning of each requirement is so that the

intent of the requirement is not lost. Second, using a tool such as SpeAR to write formalized requirements, allows translation of the requirements into multiple temporal logic languages that can be input into a variety of model checking or theorem proving engines.

By comparing system requirements to high level system properties in a model checker, requirements and derived requirements added throughout the design of the system can be analyzed for completeness and consistency. The properties and requirements documented in SpeAR are translated into a Lustre [56] model and sent to JKind,[49] a k-induction model checker.[42] JKind uses Satisfiability Modulo Theory (SMT) solvers, such as Z3,[41] to inductively prove that a given model meets its formal specification. In the instance of SpeAR, the model is defined by the formalized system requirements and the specification defined by its properties. JKind will either report that the systems properties are satisfied by the given requirements, or it will provide a sequence of inputs, referred to as a counterexample, that demonstrates how a property can be violated. This counterexample can be used by domain experts to refine either the requirements or the properties until a proof is obtained.

All of the requirements in SpeAR are Boolean expressions that can be evaluated as true or false. An example requirement written in SpeAR is shown in Figure 6.

```
//R01 The commanded change of angular velocity from the
control algorithm shall not exceed the maximum allowable
angular acceleration of the reaction wheel.
r_01a_ctrl = global ::  always psi_dot_comm_1 <= MAX_PSI_DOT_RW
and psi_dot_comm_1 >= -MAX_PSI_DOT_RW; //[rad_p_s_2]
```

Figure 6. SpeAR Example

The first requirement of the reaction wheel attitude control system is that the acceleration command from the controller should not exceed the maximum allowable acceleration of the reaction wheel. As will be discussed in Section 4.1.1, requirement

R01 is a safety property set by the physical limitations of the motor, control system, and power supply, and is designed to prevent damage to the reaction wheel assembly. The requirement is broken out for each individual reaction wheel, and the first wheel sub requirement is documented as `r_01a_ctrl`, which is a naming convention where `r` indicates that it is a requirement versus an assumption, `01` is the number of the requirement, `a` is the first sub requirement, and `ctrl` denotes that it is a controller subsystem requirement. The requirement is interpreted formally as: under any conditions (globally) the commanded change of angular velocity (represented here as `psi_dot_comm_1`) is always less than or equal to the maximum allowable angular acceleration of the system (`MAX_PSI_DOT_RW`) and greater than or equal to the negative of the maximum allowable angular acceleration. Both the negative and positive limits are included because they represent the maximum velocity in different directions.

One of the strengths of SpeAR is the ability to express and analyze non-functional requirements that describe non-functional properties. For instance, satellites usually have strict requirements on maximum weight, cost, and power consumption. If the satellite includes multiple subsystems with multiple components, designers can specify requirements for each subsystem and its components and verify that the sum each of the individual subsystems and their components do not violate the overall spacecraft requirements, e.g. the total cost of the individual subsystems is less than the allowable cost for the subsystem. While this capability may seem trivial, it is valuable on a larger scale for complex systems that may have thousands of individual components distributed among dozens of subsystems. The compositional nature of SpeAR allows designers to pull component-level verification results up to complete subsystem-level verification, and subsystem-level verification up to complete system-level verification. Another valuable analysis feature of note in SpeAR is the ability to define units used

throughout the system and analyze compliance. SpeAR is a new tool that is remains under development. Outside of this research, SpeAR has only been used to analyze requirements of an academic coupled tanks system example.[54]

2.4.2 Architecture Analysis with AGREE

The properties and requirements expressed in SpeAR are then used to develop the system architecture in the Architecture Analysis & Design Language (AADL). AADL was standardized by SAE in 2004 as an architecture modeling framework to allow analysis of system designs prior to detailed development.[82] AADL allows for the inclusion of annexes for additional functionality. The Assume Guarantee Reasoning Environment (AGREE) AADL annex[97] was developed by Rockwell Collins and University of Minnesota as part of the DARPA High Assurance Cyber Military Systems (HACMS) program to use assume-guarantee contracts to evaluate the behavior of the subsystem components with the greater system. Properties in SpeAR are documented as guarantees of the system in AGREE, and requirements developed in SpeAR are used to build the system behavior. Like SpeAR, AGREE utilizes JKIND and Z3 to conduct model checking and theorem proving to assure that the guarantees hold. Again, like SpeAR, Z3 outputs proofs or counterexamples that your architecture fulfills the guarantees with the given assumptions. Just as they are in SpeAR, all of the assumptions and guarantees in AGREE are Boolean expressions that can be evaluated as true or false. The example requirement in Figure 6 is expressed in AGREE as shown in Figure 7. In the AGREE annex, the statement in quotation marks is what is output as proven or falsified in the analysis, but is not actually analyzed. It is proven if the statement listed below the quoted section is proven to be true by the model checker. If the guarantee is falsified, a counterexample is produced showing how the system architecture violated the guarantee.

```
-- R01 The commanded change of angular velocity from the
control algorithm shall not exceed the maximum allowable
angular acceleration of the reaction wheel.
guarantee    "r_01a_ctrl:  RW1 commanded angular acceleration
shall not exceed maximum allowable angular acceleration" :
(psi_dot_comm_1 <= MAX_PSI_DOT_RW)    and    (psi_dot_comm_1 >=
-MAX_PSI_DOT_RW);
```

Figure 7. AGREE Example

2.4.3 Component Verification with SLDV

Simulink Design Verifier (SLDV) uses formal methods to detect design errors and requirement violations in a Simulink model. SLDV can also detect integer overflow, dead logic, array access violations, division by zero with static analysis.[91] The primary function of interest for this research is SLDV's ability to detect requirement violations in the model with formal methods. The assumptions and guarantees developed during the architecture phase are used to manually write the SLDV verification functions. The model is analyzed using SLDV to prove that the requirements (documented as guarantees in AGREE, and as properties and requirements in SpeAR) hold true throughout the modeling phase. First, the assumptions and requirements must be stated formally. This step was completed in the requirements design phase of this research. Just as it was in the requirements and architecture phase, all the requirements in SLDV are Boolean expressions that can be evaluated as true or false. The same requirement example from Figure 6 and Figure 7 is carried through into Figure 8, which shows the requirement specified as a property in SLDV.

```
%R01 The commanded change of angular velocity from the
control algorithm shall not exceed the maximum allowable
angular acceleration of the reaction wheel.
r_01a_ctrl = (psi_dot_comm_1 <= MAX_PSI_DOT_RW) &&
(psi_dot_comm_1 >= -MAX_PSI_DOT_RW);
```

Figure 8. Simulink Design Verifier Example

2.5 Historic Spacecraft Software Failure Cases Addressable by Formal Methods

This section describes historical examples of spacecraft failures attributed to different types of software errors that are identifiable by formal methods analysis.[68] Errors such as divide by zero, overflow, mixed precision, out-of-bounds array access, unreachable code, requirement conflictions, and unit inconsistency can be caught using a variety of formal methods tools and techniques.[55]

Figure 9. Mission Specialists Grapple Intelsat During STS-49 [7]

One historical example of a mixed precision spacecraft software failure occurred on STS-49. During this Space Shuttle mission to rendezvous with and repair the Intelsat satellite, pictured in Figure 9, the Lambert Targeting Routine used to calculate rendezvous firings contained a mixed precision error that prevented the routine from converging and nearly caused the mission to be aborted.[8] While the state vector variables were double precision, the limits used to bound the calculation were single precision. The mission was recovered by a workaround that allowed state vector information to be relayed from the ground. McAllister stated at the 2014 NASA Formal

Methods Symposium that later analysis showed formal methods would have found this error prior to flight.[68]

Figure 10. Ariane 5 Launch [2]

Figure 11. Ariane 5 Explosion [1]

Ariane 5 also had a catastrophic failure that could have been prevented with the use of formal methods design analysis. The Ariane 5 is a European heavy lift launch vehicle, pictured in Figure 10, that reused portions of Ariane 4 code on its first flight, Ariane 501. The software contained a bad 64 to 16 bit conversion which caused the vehicle trajectory to veer off course and prompted self-destruction of the

system, pictured in Figure 11.[43] Lacan et al presented research at the 1998 Data Systems in Aerospace Conference showing how static formal-methods based analysis tools were successfully applied to formally verify portions of code for Ariane 502 and the following Ariane flights [63].

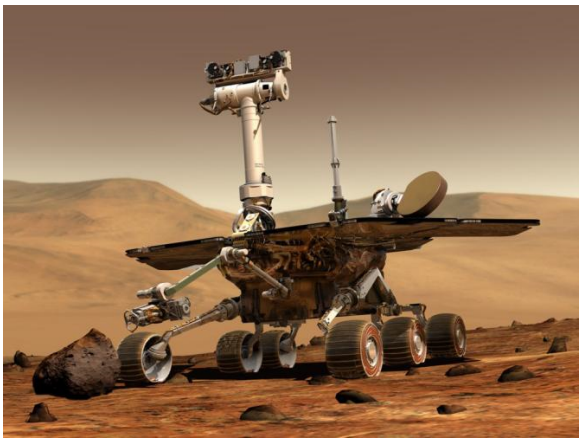

Figure 12. An Artist's Concept Portrays a NASA Mars Exploration Rover on the Surface of Mars [3]

In 2004, the Mars Exploration Rover, Spirit, pictured in Figure 12, was unexpectedly shut down for 10 days after a parameter in the software permitted unlimited consumption of system memory as flash memory was exhausted. Denny cites this error as one that could have been caught had formal methods been applied. [43]

Perhaps one of the most famous spacecraft software failures occurred on the Mars Climate Orbiter (MCO) program, pictured in Figure 13, in which a conflict between English and metric units in the software resulted in loss of the system. According to the Mishap Investigation Board Phase I Report [9]:

> "The [Mars Climate Orbiter Mishap Investigation Board] has determined that the root cause for the loss of the MCO spacecraft was the failure to use metric units in the coding of a ground software file,"Small Forces," used in trajectory models. Specifically, thruster performance data in English units instead of metric units was used in the software application code."

The requirements for the system specified that the data should be provided in metric

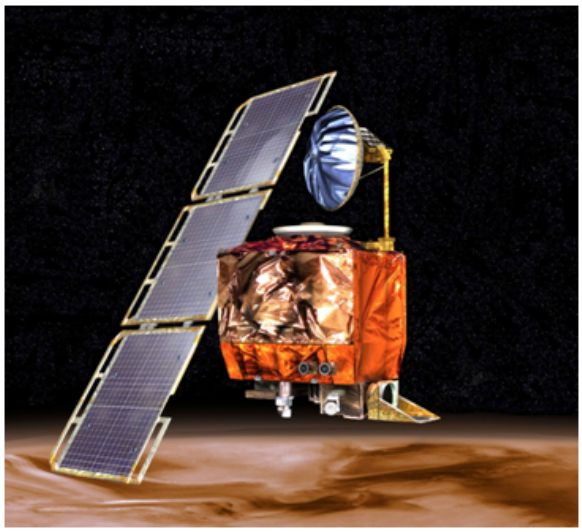

Figure 13. Artists Rendering of the Mars Climate Orbiter [4]

units. Many modern formal methods tools, such as the Specification and Analysis of Requirements (SpeAR) tool, provide unit checking capabilities to aid in identification of errors such as this.

The Mars Polar Lander mission provides an example of conflicting requirements. One requirement specified that alternative communication methods should be tested if no commands were received after 24 hours, while another requirement instructed the system to go into a "sleep mode after 24 hours to conserve battery, making the alternate communication requirement unreachable. [61] Formal methods tools can implement model checking to check for inconsistent requirements by evaluating realizability of a model given the constraining requirements.

The examples listed in this section are just a few of the many examples of software failure in spacecraft. Application of formal methods could have been used to identify many of the historical software errors that have appeared in recent decades.[52]

2.6 Run Time Assurance

RTA is a verification methodology that provides online verification of an "unverified controller" design predicated on offline verification of "verified controller" and "decision module" components, as well as a proposed control architecture, as described in Section 2.6.2. RTA allows unverified controllers, such as an intelligent controller, to perform within a predetermined envelope of acceptable behavior. Inspired by software sandboxing techniques and the simplex architecture, the initial RTA concept was developed through a series of AFRL Certification of Flight Critical Systems (CerTA FCS) programs. The objective of the first CerTA FCS program with prime contractor Northrop Grumman was to identify certification challenges for unmanned systems. The program concluded with a determination that some of the certification challenges would require run time verification methods, among other recommendations.[77] Building on the foundation of the CerTA FCS program, the CerTA FCS Challenge Problem Integration (CPI) with the Boeing Corporation as contractor focused on the verification challenge associated with adaptive control strategies. While the program successfully demonstrated time-delay stability for a specific L1 adaptive control approach, the technique could not be generalized, and did not ensure prevention of control surface rate limit failures or aircraft structural mode excitation caused by the actuators.[74][31] The CerTA FCS CPI effort with Lockheed Martin, Barron Associates, and Rockwell Collins as contractors developed an idea they called Run Time Verification and Validation (RTVV) as a cost saving solution to implement advanced systems that are not certifiable by current methods due to learning, nondeterminism, high criticality, or high complexity.[83] In the follow on CerTA FCS Challenge Problem Demonstration (CPD) program, Barron Associates introduced the conceptual "Run Time Assurance" framework used in this research in 2011. The program developed a simulated RTA autoland system with a predefined

safety boundary, violation predication, and switching to a reversionary controller (called the "verified controller" in this research, and referred to as the "safety controller" in the Simplex Architecture). Barron Associates has continued the research on RTA through a series of Small Business Innovative Research (SBIR) grants managed by AFRL. A thorough survey of RTA methodologies was completed by in April 2013 to provide an understanding of the state of the art in RTA development across a range of contributing technologies.[30]

In contrast to offline verification, RTA provides online constraining of system performance. RTA controllers feature three distinct characteristics: [30][11]

1. A run time or real-time monitoring and prediction scheme designed to detect failures not previously identified (software failure) or anticipated (bad environmental design assumptions) before they are about to occur;

2. A failsafe recovery or steering mechanism guaranteed to recover the system in the event the monitor detects an eminent failure;

3. A structured argument (or assurance case), supported by evidence, justifying that a system is acceptably safe not only through offline tests, but also through reliance on real-time monitoring, prediction, and failsafe recovery.[30][11]

The performance of the unverified control system is monitored by a decision module that includes a set of acceptable conditions and behaviors. If the system is forced to operate outside of pre-determined conditions due to an unexpected environmental change or any of the acceptable behavior ranges are violated, the decision module switches control from the unverified controller to an appropriate verified backup controller, which could be thought of as the controller's "Plan B." The system may have one or more verified backup controllers capable of tasks such as gracefully transitioning control from the unverified controller function to another verified controller

function, maintaining a safe system state (i.e. avoiding an obstacle, or performing a holding pattern), or performing the same tasks expected of advanced controller with less efficiency. The designed system of verified backup controllers and decision modules can be analyzed using formal methods to ensure that none of the pre-determined safety conditions are violated.

2.6.1 Sandboxing and the Simplex Architecture

RTA control architectures are based on concepts from software sandboxing and the Simplex Architecture. Sandboxing is a technique of isolating sections of code to limit the sections ability to cause critical system errors and is often used in software and web-based security testing to isolate untested and untrusted code from active critical resources.[76] Sandboxing is similar to RTA which isolates the unverified controller from the rest of the control system. The Simplex Architecture [81] provides protection to the plant by isolating the complex controller with the addition of a decision module and safety controller, as depicted in Figure 14. Whenever the complex controller threatens the safety of the plant, the decision module activates, switching control to a proven safety controller. The "complex controller" in the Simplex Architecture is analogous to the RTA "unverified controller" and the "safety controller" corresponds to the "verified controller."

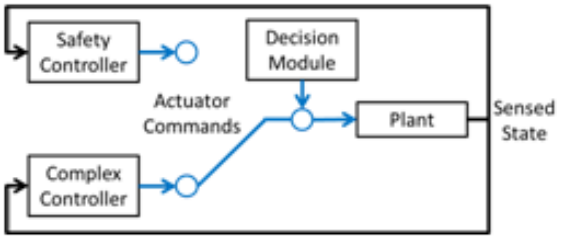

Figure 14. The Simplex Architecture

The Simplex Architecture has been successfully applied to systems such as a fleet of remote-controlled cars [38], a pacemaker [17], a set of advanced aircraft maneuvers

[80], and an autonomous waypoint tracking system [18]. While sandboxing techniques and Simplex Architectures are sometimes used only in testing environments, the RTA architecture is envisioned to be deployed with the operational system to ensure safe behavior of the system in its environment.

2.6.2 RTA Control Architecture

In the proposed RTA architecture pictured in Figure 15, the system state is sent to each of the subcomponents: the unverified controller, the verified controller, and the decision module. The verified and unverified controller each provide actuator commands to the decision module. The decision module monitors the unverified controller, and if a safety property is violated, control is switched to the verified controller. The verified controller has been shown to meet safety properties through some offline verification method such as formal methods analysis, simulation, and flight test. The RTA control architecture is not limited to one verified safety or recovery controller. An RTA controller implementation might be devised such that it has multiple recovery controllers and switches to the appropriate one based on the state of the system.

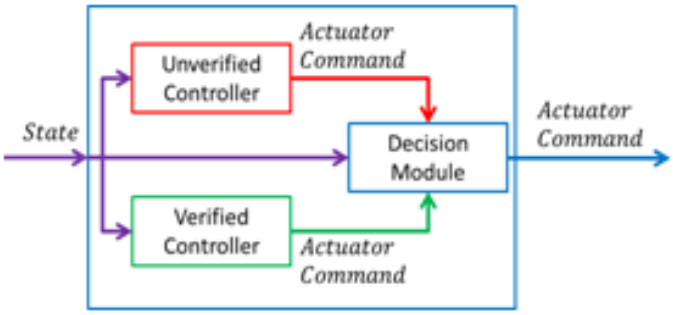

Figure 15. The Run Time Assurance Architecture

2.6.3 Auto GCAS as an Example RTA with Complex Switching Logic

The decision module is not limited to switching only when a specific safety property is violated either; the decision to switch could be made on a much more complex set of criteria. For example, the F-16 Automatic Ground Collision Avoidance System (Auto GCAS), developed by AFRL and Lockheed Martin,[89] could be considered an RTA control system if one examines its core functionality. Auto GCAS depends on the pilot to be the primary controller of the aircraft. In the RTA control architecture analogy, the pilot could be thought of as the "unverified controller" because he or she is subject to situations such as gravity-induced loss of consciousness, target fixation, task saturation, or loss of situational awareness that may compromise his or her ability to control the aircraft. Continuing with the RTA control architecture analogy, the prescripted automatic recovery maneuver Auto GCAS uses to avoid terrain would be considered the "verified controller." This automatic maneuver rolls the aircraft to wings level and pulls up to 5 g's (5 times the force of gravity) to avoid colliding with the terrain. The Auto GCAS algorithm serves as the "decision module" in the RTA control architecture analogy with the responsibility to monitor the path of the aircraft, switch control to an automatic recovery controller when a collision is imminent, and switch control back to the pilot when the aircraft has cleared the terrain. The Auto GCAS "decision module," uses a complex algorithm that compares a projection of the automatic recovery maneuver to a two dimensional projection of the terrain with buffers ahead of the aircraft to determine if a ground collision is imminent. When the projection of the recovery maneuver intersects the projection of the terrain with safety buffers as show in Figure 16, the Auto GCAS "decision module" switches control from the pilot (unverified controller) to the automatic recovery maneuver (verified controller). Once the terrain is cleared, the Auto GCAS "decision module" switches control from the automatic recovery maneuver back to the pilot.

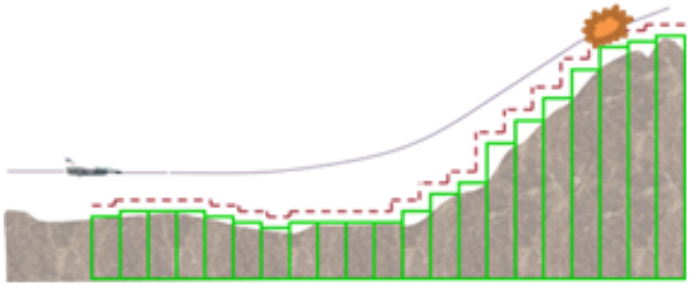

Figure 16. The Auto GCAS Automatic Maneuver and Terrain Projection Comparison

This example simplifies the Auto GCAS functionality, but illustrates an alternative implementation of an RTA control architecture than the case study presented in this thesis.

2.7 Space Filling Experimental Design Simulations (SFEDS)

Conducting batch simulations that cover a wide swath of the design space expands the analysis of a single simulation case to enable more comprehensive analysis. However, conducting simulations of every possible input may be extremely computationally expensive and infeasible. While the simulated equations of motion and controller used in this research are deterministic (given the same set of inputs, they will produce the same result every time), the possible initial and final spacecraft orientations that the system could encounter are non-deterministic, situational-dependent values. The initial orientation could be the result of a previous command, random perturbations on the satellite, or a number of other variables. The final orientation could be a vector corresponding to aspects of the mission, such as an orientation that places a sensor in a specific direction or the solar panels in a position to absorb maximum sunlight, that are subject to change and reprioritization. In addition, the nonlinear nature of the system makes it difficult to predict which initial and final pointing angles of the satellite could lead to violations of the requirements. Using SFEDS allows for a section of the design space to be explored in a manageable set of

100s to 10,000s of simulations.

SFEDS is similar to Monte Carlo simulation. According to the Palisade Corporation, "Monte Carlo simulation performs risk analysis by building models of possible results by substituting a range of values for any factor that has inherent uncertainty" and using latin hypercube (LH) sampling rather than a random sampling can produce a more accurate set of results from the entire range of possible values.[36] The uncertain factors in the SFEDS conducted in this research are the initial and final orientation of the spacecraft, and the risk of interest is that the requirements of the control system are violated. Wolfram Research Inc. defines Monte Carlo methods as "any method which solves a problem by generating suitable random numbers and observing that fraction of the number obeying some property or properties."[95] In this research, the suitable "random" numbers generated correspond to descriptions of the initial and final orientation of the satellite and the properties of interest are the requirements of the controller. In this context SFEDS may be used to evaluate what fraction of possible initial and final pointing angle conditions result in a violation of the requirements during a maneuver with a nonlinear PID controller. What separates the SFEDS in this research from Monte Carlo simulation is that the system studied in this research is deterministic and system variables are not being modified during the simulation. SFEDS are better suited for identifying unknown response surfaces where complex forms and localized effects are possible.[70]

An SFEDS may be conducted to determine if there is a relationship between the explanatory variables, also known as independent variables, and the response variables, also known as dependent variables. The number of explanatory variables in the simulation is referred to as the number of factors and the relationship between the explanatory and response variables is referred to as a response surface.[20] There are a variety of methods used to generate a matrix of explanatory variables for an SFEDS;

however, LH designs, first proposed in 1979,[69] have become the predominant design choice for computer simulation experiments.[26] When all of the input variables of an LH have zero correlation, it is referred to as an orthogonal LH (OLH).[59] Hernandez defines a nearly OLH (NOLH) as an LH in which no two variables have a correlation of more than 0.05.[58] One of the advantages of the NOLH experiment designs is that they enable identification of non-linear relationships between the explanatory, or independent variables, and response, or dependent variables.[23] For this research the space-filling design of NOLH was selected to provide an exploration of the response surface.

The response surface generated by the SFEDS may be characterized by a variety of methods, including graphically or statistically using software such as JMP Statistical Discovery Software from SAS. JMP can be used to generate a statistical model of the response curve with a coefficient of determination R^2 value that indicates how well the statistical data fits the model generated. An R^2 value of 1 indicates that the model fits the data perfectly while an R^2 value of 0 indicates that data doesnt fit the model at all.

2.8 Summary

Chapter II began by providing background on cyber-physical systems theory as it relates to modern control theory, before describing characteristics of good requirements, requirement types, and requirement verification. The foundational theories and techniques behind FMA were presented and the tools used to conduct FMA of the requirements, architecture, and model of the nonlinear control system were described. Additional analysis techniques including SFEDS were introduced. The topics of Chapter II provide context and background information on the analysis techniques used in this research.

III. CubeSats and Spacecraft Attitude Dynamics and Control

Chapter III provies background in CubeSats, reference frames, and spacecraft attitude dynamics and control.

3.1 6U CubeSat Application

The AFIT 6U CubeSat ACS is used as a challenge problem for non-traditional verification techniques including formal methods, run time assurance, and reachability analysis. This section provides background on CubeSats, the AFIT 6U CubeSat, reaction wheel dynamics, magnetic torque coil actuation, the equations of motion for the system, and a set of hypothetical requirements for a spacecraft attitude control subsystem.

3.1.1 CubeSats

CubeSats are small satellites built in units, or "U," which are nominally 10 x 10 x 10 cm^3, or 1 liter, in volume and have a mass of 1.3 kilograms (3 pounds) or less. CubeSats initially began as a set of amateur radio experiments in the 1970s - 1980s, matured as university experiments in the 1990s - 2000s, and began being utilized for technology experiments and to carry out real missions in the 2010s.[78] More recently, CubeSats have been considered for increasingly complex missions including interplanetary and lunar missions such as NASA's Lunar Flashlight mission, which seeks to determine presence or absence of exposed water ice on our Moon's south pole.[35] Complex missions can have precise pointing requirements, depending on the requirements of the satellite payload. While attitude control actuators exist for small 1-3U CubeSat designs, larger actuators and accompanying control algorithms are needed to provide greater control authority over larger 6-27U CubeSat designs.

3.1.2 CubeSats for DoD Missions

CubeSats are generally much less expensive to design, test, transport and launch than larger traditional satellites and therefore present a potential time and cost effective approach to meet DoD project needs.[96] However, the limited size and weight of smaller CubeSats provide relatively little space for mission-specific payloads and the advanced subsystems required to support them. To meet the increasing needs associated with anticipated DoD missions, AFIT's Center for Space Research and Assurance (CSRA) is currently developing 6, 12, and 27U CubeSats which promise a larger payload to bus ratio than the traditional 1U and 3U versions.[87]

3.1.3 AFIT 6U CubeSat Testbed

AFIT's 6U CubeSat Attitude Determination and Control System (ADCS) Testbed is part of a engineering development unit (EDU) for classroom use only that is incrementally built by students in a master's level course called ASYS 632 Satellite Design and Test.[12] The approximate mass of the CubeSat bus and chassis without cross braces has a mass of approximately 4 kg, and a volume of approximately 2000 cm^3, leaving approximately 8 kg of mass and 5000 cm^3 of volume for additional subsystems and payload. The complete Attitude Determination System (ADS) of the EDU may contain up to six sun sensors, one or two Earth sensors, one or two star sensors, two magnetometers, and an IMU. The complete ACS may contain a reaction wheel array (RWA) and three magnetic torque coils (MTC). The complete ACS software is envisioned to have multiple control modes including a sun-soak mode that orients the solar cells for maximum power generation, an alignment mode that orients the spacecraft body vector with an inertial vector, and an alignment mode that orients a spacecraft body vector with a nadir vector. The attitude control algorithm is assumed to be a PID controller. The 2015 ADCS Testbed, as pictured in

Figure 17, includes the following components and subsystems: ADCS board, motor control board, RWA, electrical power system(EPS) board, battery pack, command data and handling (CDH) board with WiFly capability, and a laser pointer.

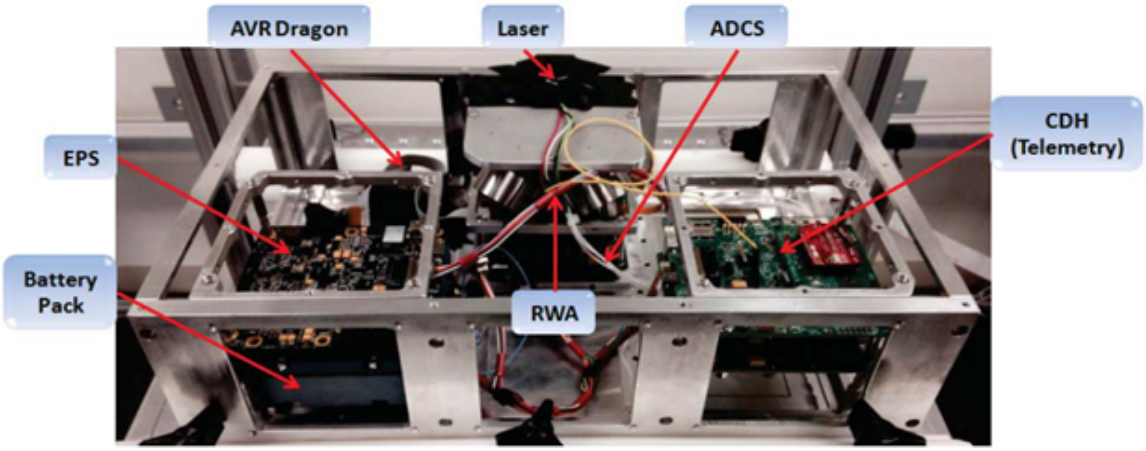

Figure 17. AFIT 6U ADCS Testbed[92]

3.1.4 AFIT Four Wheel Reaction Wheel Array

The AFIT RWA implemented on the ADCS test bed includes four wheels in a pyramid configuration that provides redundancy if a failure were to occur in one of the wheels. The AFIT CubeSat four wheel RWA is shown in Figure 18.

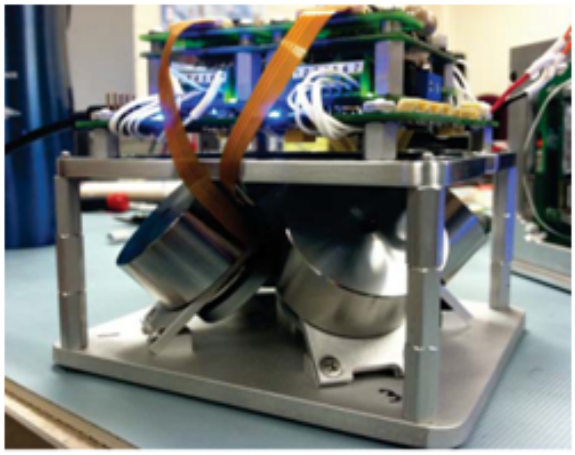

Figure 18. AFIT's CubeSat Four Wheel Pyramid Reaction Wheel Assembly[40]

The RWA is not flight qualified because it cannot survive expected launch vehicle vibration loads; however, its low friction and redundancy make it an excellent system for the laboratory. A CAD Model of just the RWA and its frame without the electronics stack is shown in Figure 19. The numbering scheme and coordinate system that describe orientation of the individual reaction wheels with respect to the RWA assembly is shown in Figure 20.

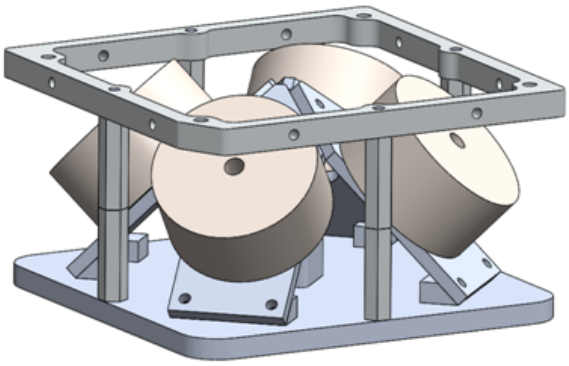

Figure 19. CAD Model of AFIT Four Wheel RWA

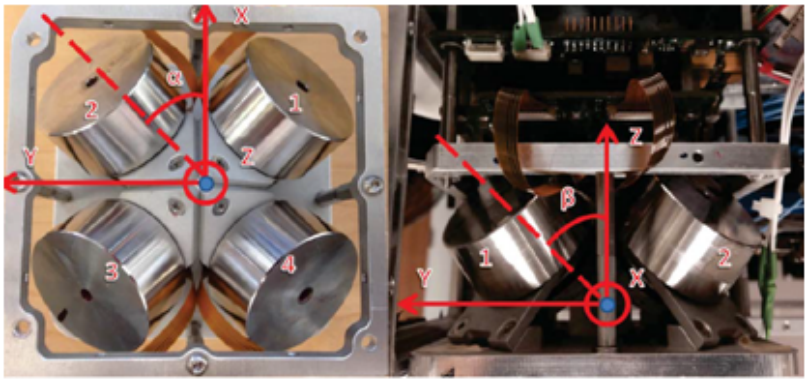

Figure 20. Four Wheel Pyramid Reaction Wheel Array Coordinate System[92]

The angle between the reaction wheel angular momentum vector and the x-axis, represented as α, for wheels 1, 2, 3, and 4 is 315, 45, 135, and 225 degrees, respectively. The angle between the reaction wheel angular momentum vector and the z-axis, represented as β, is 45 degrees.

3.2 Spacecraft Attitude Dynamics and Control

The purpose of an ACS is to control the orientation of the spacecraft relative to other objects. Examples include the need to point an onboard sensor or communication antenna at a specific place on the Earth, or to orient the spacecraft solar panels orthogonal to the vector from the satellite to the sun for maximum power generation. In this section, the kinematics and kinetics of spacecraft attitude are discussed, example control actuators are described, a PID control algorithm is presented, and the equations of motion for spacecraft attitude control are derived.

3.2.1 Kinematics

Kinematics provide a mathematical description of the relationship between spacecraft orientation and angular velocity. A variety of spacecraft orientation representations and a select set of spacecraft attitude kinematic equations are presented in this section.

3.2.1.1 Rotation Matrices

A rotation matrix, also known as a Direction Cosine Matrix, is one way to convert vectors from one coordinate frame to another. In spacecraft attitude dynamics, a vector in the spacecraft body frame $\vec{v}_b$ may be found from a vector in the spacecraft inertial frame $\vec{v}_i$, as using a rotation matrix $\boldsymbol{R}^{bi}$ as

$$\vec{v}_b = \boldsymbol{R}^{bi}\vec{v}_i. \tag{1}$$

The entries of the rotation matrix are dot products of the vertices.[86] It can be shown that the inverse of a rotation matrix is equivalent to the transpose of the rotation matrix ($\boldsymbol{R}^{-1} = \boldsymbol{R}^{T}$).[86] Thus rotation matrices are orthonormal, meaning

that each column has a magnitude of 1 ($||\hat{i}_1|| = ||\hat{i}_2|| = ||\hat{i}_3|| = 1$) and all the columns are mutually independent ($\hat{i}_1 \cdot \hat{i}_2 = \hat{i}_1 \cdot \hat{i}_3 = \hat{i}_2 \cdot \hat{i}_3 = 0$). The rotation matrix can be written as a concatenation of 3×1 column matrices corresponding to one frame's unit vectors and another frame's components.[84] The rotation matrix notation, equation, matrix of components, and a concatenation of 3×1 column matrices may be expressed as

$$\boldsymbol{R}^{bi} = \hat{b} \cdot \hat{i}^T = \begin{bmatrix} \hat{b}_1 \cdot \hat{i}_1 & \hat{b}_1 \cdot \hat{i}_2 & \hat{b}_1 \cdot \hat{i}_3 \\ \hat{b}_2 \cdot \hat{i}_1 & \hat{b}_2 \cdot \hat{i}_2 & \hat{b}_2 \cdot \hat{i}_3 \\ \hat{b}_3 \cdot \hat{i}_1 & \hat{b}_3 \cdot \hat{i}_2 & \hat{b}_3 \cdot \hat{i}_3 \end{bmatrix} = \begin{bmatrix} \left[\hat{i}_1\right]_b & \left[\hat{i}_2\right]_b & \left[\hat{i}_3\right]_b \end{bmatrix}. \tag{2}$$

Because a general rotation has three degrees of freedom and a rotation matrix contains 9 components, there are 6 constraints corresponding to its orthonormal properties ($||\hat{i}_1|| = ||\hat{i}_2|| = ||\hat{i}_3|| = 1$ and $\hat{i}_1 \cdot \hat{i}_2 = \hat{i}_1 \cdot \hat{i}_3 = \hat{i}_2 \cdot \hat{i}_3 = 0$).[84]

3.2.1.2 Euler Angles

Rotation matrices form the bases of Euler angles, which describe 3 successive simple rotations about individual axes. Leonhard Euler first suggested using a sequence of rotations to convert an orientation in an orbital frame to an inertial frame in the 18th century. It can be shown that three independent parameters are the minimum required to fully describe any rotation.[84] The order in which the rotation matrices are multiplied is key, and the first rotation must be on the right most side of the triple product. An example of a 3-2-1 sequence is shown here as

$$\mathbf{R}^{3-2-1} = \mathbf{R}_1(\theta_3)\mathbf{R}_2(\theta_2)\mathbf{R}_3(\theta_1)$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_3) & \sin(\theta_3) \\ 0 & -\sin(\theta_3 & \cos(\theta_3) \end{bmatrix} \begin{bmatrix} \cos(\theta_2) & 0 & -\sin(\theta_2) \\ 0 & 1 & 0 \\ \sin(\theta_2) & 0 & \cos(\theta_2) \end{bmatrix} \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) & 0 \\ -\sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (3)$$

There are several disadvantages to using Euler angle representations. First, Euler angle representations are subject to singularities when the second angle is equal to 0, 90, 180, 270, or 360 degrees ($0, \pi/2, \pi, 3\pi/2$, or 2π radians) depending on the sequence. These singularities are of particular concern for spacecraft, which experience large angle motion and may be tumbling and frequently entering attitudes where the singularities are present. Second, trigonometric functions have a high computation cost which could tax space-hardened processors that are several generations behind modern processors, although spacecraft processing power is improving and the computational cost of trigonometric functions is becoming less important.[84]

3.2.1.3 Euler Axis Rotation

Euler axis rotations, also known as eigenaxis rotations, represent rotations as a single rotation about a fixed Euler axis ($\hat{a}$), and a principal Euler angle of rotation (Φ). For pure rotations where the Euler axis passes through the origin,[84] the motion can be described by a rotation matrix that satisfies

$$\boldsymbol{R}^{bi}\hat{a} = \hat{a}. \quad (4)$$

The Euler axis is the eigenvector of the rotation matrix associated with an eigenvalue of 1. The Euler axis and principle Euler angle representation contains four

parameters but only three are independent with a constraint that the product of transpose of the Euler axis with the Euler axis must equal 1 ($\hat{a}^T\hat{a} = 1$). The Euler axis and principle Euler angle [84] may be computed from

$$\Phi = \cos^{-1}\Big[\frac{1}{2}(\text{trace}(\boldsymbol{R}) - 1)\Big]. \tag{5}$$

$$\hat{a}^{\times} = \frac{1}{2\sin(\Phi)}(\boldsymbol{R}^T - \boldsymbol{R}) \tag{6}$$

where $\hat{a}^{\times}$ is a skew-symmetric matrix comprised of components of $\hat{a}$. The Euler axis and Euler angle representation is still subject to a singularity when the sine of the principle Euler angle is equal to 0 or 2π ($\sin(\Phi) = 0$ or 2π).[84]

3.2.1.4 Quaternions

A quaternion, also known as a Euler parameter set, is a four-parameter array ($\bar{q}$) with a unit norm constraint and is comprised of vector component $\vec{q}_{123}$ and a scalar component q_4. The quaternions may be calculated from the Euler axis and principle Euler angle using

$$\vec{q}_{123} = \hat{a}\sin\Big(\frac{\Phi}{2}\Big) \tag{7}$$

and

$$q_4 = \cos\Big(\frac{\Phi}{2}\Big). \tag{8}$$

A rotation matrix can be generated from

$$\boldsymbol{R} = (q_4^2 - (\vec{q}_{123})^T\vec{q}_{123})\boldsymbol{U}_{3\times3} + 2\vec{q}_{123}(\vec{q}_{123})^T - 2q_4(\vec{q}_{123})^{\times} \tag{9}$$

where $\boldsymbol{U}_{3\times3}$ is a 3×3 identity matrix, and $(\vec{q}_{123})^{\times}$ is a skew-symmetric matrix with

components of $\vec{q}_{123}$. The quaternion can be generated from a rotation matrix with

$$q_4 = \pm 0.5\sqrt{1 + \text{trace}(\boldsymbol{R})} \tag{10}$$

and

$$\vec{q}_{123} = \frac{1}{4q_4}\begin{bmatrix} R_{23} - R_{32} \\ R_{31} - R_{13} \\ R_{12} - R_{21} \end{bmatrix}. \tag{11}$$

3.2.1.5 Quaternion Kinematics

The time rate of change of a quaternion $\dot{\bar{q}}$ is a function of the current quaternion and the spacecraft angular velocity vector $\vec{\omega}$, and may be calculated from

$$\dot{\bar{q}} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2}\boldsymbol{Q}_{4\times 3}\vec{\omega} = \frac{1}{2}\begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{12}$$

One of the primary advantages of quaternions is that quaternion differential or kinematic equations do not contain any singularities.

3.2.2 Kinetics

Kinetics describe the relationship between torque, angular velocity, and angular acceleration. The angular momentum of the spacecraft $\vec{h}_{sc}$ is given by

$$\vec{h}_{sc} = \boldsymbol{I}\vec{\omega} \tag{13}$$

where $\boldsymbol{I}$ is the mass moment of inertia (MOI) matrix of the spacecraft. The MOI matrix contains scalar moments of inertia on the diagonal, and products of inertial

on the off diagonal. When the body frame axes are aligned with the principal axes, the MOI matrix of the body of the spacecraft can be diagonalized, so that the MOI of the spacecraft can be entirely described by scalar moments of inertia on the diagonal, in the form

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}. \tag{14}$$

Spacecraft experience two types of torque: internal and external. Internal torque usually comes from actuators such as reaction wheels, momentum bias wheels, and control moment gyros, while external torque could be applied from orbital perturbations, aerodynamic drag or control actuators such as thrusters, magnetic torque rods, magnetic torque coils, or solar or aerodynamic torque actuators. In the absence of external torques, the total angular momentum $\vec{H}_{tot}$ of the spacecraft is a sum of the angular momentums of the spacecraft $\vec{h}_{sc}$ and any internal torque actuators $\vec{h}_{internal}$, given by

$$\vec{H}_{tot} = \vec{h}_{sc} + \vec{h}_{internal}. \tag{15}$$

When external torques are present, the relationship between the external moments $\vec{M}$ and the time rate of change of angular momentum $\dot{\vec{H}}$ must be computed in an inertial frame,[57] as described in the Euler's equation by

$$\vec{M} = \dot{\vec{H}} = \frac{\mathrm{d}^{(i)}}{\mathrm{dt}}(\vec{H}). \tag{16}$$

The transport theorem [85] may be used to write Euler's equation in a body-fixed frame, as

$$\vec{M} = \dot{\vec{H}} = \dot{\vec{h}}_{sc} + \vec{\omega}^{\times}\vec{h}_{sc} + \dot{\vec{h}}_{rwa} + \vec{\omega}^{\times}\vec{h}_{rwa} = \mathbf{I}\vec{\omega} + \vec{\omega}^{\times}\mathbf{I}\vec{\omega} + \dot{\vec{h}}_{rwa} + \vec{\omega}^{\times}\vec{h}_{rwa} \tag{17}$$

where $\vec{\omega}^{\times}$ is the skew-symmetric matrix of spacecraft angular rates.

3.2.3 Reaction Wheel Actuation

Reaction wheels are common actuators used for precision spacecraft attitude control that work by exchanging momentum with the spacecraft. By contrast, actuators such as thrusters or magnetic torque devices interact with the environment to apply an external torque to the spacecraft, which changes the spacecraft total angular momentum $\vec{H}_{tot}$. In the absence of external torques, the total angular momentum of the spacecraft remains constant, and the total angular momentum of the RWA is equal and opposite to the angular momentum that it exchanges with the spacecraft. One disadvantage of RWAs is their vulnerability to a phenomenon known as saturation, which occurs once the maximum angular velocity rates one or more reaction wheels are reached and the RWA is no longer able to control the spacecraft attitude in certain directions. RWAs are one of the most precise attitude actuators with accuracy of up to 0.001 degree [64] in some cases, and are less complex than actuators such as control moment gyros, but they are one of the heavier actuator options available.[96][22][51]

3.2.3.1 Individual Reaction Wheel Dynamics

The individual reaction wheel's angular momentum $\vec{h}_{rw}$ is

$$\vec{h}_{rw} = D\vec{\psi}_i \tag{18}$$

where D is the wheel's mass moment of inertia, $\vec{\psi}_i$ is the rate of spin about the axis of the spin of reaction wheel i.

3.2.3.2 3-Wheel Reaction Wheel Array Dynamics

When the RWA has three wheels, each identical and aligned with the body axes of the spacecraft, the angular momentum of the RWA $\vec{h}_{rwa}$ can be written as

$$\vec{h}_{rwa} = \vec{h}_{rw1} + \vec{h}_{rw2} + \vec{h}_{rw3} = D \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix}. \tag{19}$$

3.2.3.3 4-Wheel Pyramid Reaction Wheel Array Dynamics

In the case of a 4-wheel pyramid RWA, like that simulated in this research, a 4×3 torque mapping matrix is needed to transform the torque vector of each wheel to the body frame. The angular momentum of the RWA $\vec{h}_{rwa}$ is given by:

$$\vec{h}_{rwa} = \boldsymbol{S}\vec{\psi} \tag{20}$$

where $\vec{\psi}$ is an array containing the angular velocity of each of the reaction wheels, and $\boldsymbol{S}$ is a matrix that orients the angular momentum contribution of each reaction wheels with the body axes of the spacecraft. The $\boldsymbol{S}$ matrix is defined as:

$$\boldsymbol{S} = D \begin{bmatrix} \cos(\alpha_1)\sin(\beta) & \cos(\alpha_2)\sin(\beta) & \cos(\alpha_3)\sin(\beta) & \cos(\alpha_4)\sin(\beta) \\ \sin(\alpha_1)\sin(\beta) & \sin(\alpha_2)\sin(\beta) & \sin(\alpha_3)\sin(\beta) & \sin(\alpha_4)\sin(\beta) \\ \cos(\beta) & \cos(\beta) & \cos(\beta) & \cos(\beta) \end{bmatrix} \tag{21}$$

where α is the angle between a particular reaction wheel's angular momentum vector and the spacecraft's x-axis and β is the angle between the reaction wheel angular momentum vector and the z-axis.

3.2.3.4 Momentum Exchange

As discussed earlier, RWAs are momentum exchange actuators that work by exchanging momentum with the spacecraft while not changing the total angular momentum of the spacecraft. The total angular momentum of the spacecraft with RWA actuation is given by

$$\vec{H}_{tot} = \vec{h}_{sc} + \vec{h}_{rwa}. \tag{22}$$

3.2.4 PID Spacecraft Attitude Control

The first step in the control process is determining the difference between the present and desired orientation. This difference is represented by quaternion error $\bar{q}_e$ and is computed from

$$\bar{q}_e = \begin{bmatrix} q_{4c} & -q_{3c} & q_{2c} & q_{1c} \\ q_{3c} & q_{4c} & -q_{1c} & q_{2c} \\ -q_{2c} & q_{1c} & q_{4c} & q_{3c} \\ -q_{1c} & -q_{2c} & -q_{3c} & q_{4c} \end{bmatrix} \bar{q}_p \tag{23}$$

where q_{ic} terms are the commanded quaternions, and $\vec{q}_p$ is the present quaternion array. The controller used in this work is a PID controller introduced by Wie [98] in which the calculated change in angular momentum of the RWA $\dot{\vec{h}}_{rwa}$ is from

$$\dot{\vec{h}}_{rwa} = \mathbf{K}_P \mathbf{I} \vec{q}_{e_{123}} + \mathbf{K}_D \mathbf{I} \vec{\omega} + \mathbf{K}_{I11} \Delta t (q_{e_1} + q_{e_2} + q_{e_3}) - \vec{\omega}^{\times} (\mathbf{I}\vec{\omega} + \vec{h}_{rwa}) \tag{24}$$

where $\mathbf{K}_P$, $\mathbf{K}_I$, and $\mathbf{K}_D$ are the proportional, integral, and derivative gains of the controller, $\vec{q}_{e_{123}}$ is a vector part of the error quaternion, $\mathbf{I}$ is the spacecraft 3×3 matrix mass moment of inertia, $\vec{\omega}$ is the spacecraft angular velocity vector, Δt is the time step of the simulation, $\vec{\omega}^{\times}$ is a skew-symmetric matrix with off-diagonal components comprised of the angular velocity vector elements, and $\vec{h}_{rwa}$ is the RWA

angular momentum. The gains of this controller are calculated from

$$\mathbf{K}_P = \mathbf{I}\left(\omega_n^2 + \frac{2\zeta\omega_n}{T}\right), \tag{25}$$

$$\mathbf{K}_I = \mathbf{I}\left(\frac{\omega_n^2}{T}\right), \tag{26}$$

$$\mathbf{K}_D = \mathbf{I}\left(2\zeta\omega_n + \frac{1}{T}\right), \tag{27}$$

and

$$T = \frac{10}{\zeta\omega_n}. \tag{28}$$

The resulting reaction wheel commanded angular acceleration $\dot{\vec{\psi}}_{com}$ is calculated from

$$\dot{\vec{\psi}}_{com} = \boldsymbol{S}^{+}\dot{\vec{h}}_{rwa} \tag{29}$$

where $\boldsymbol{S}^{+}$ is the pseudoinverse of the $\boldsymbol{S}$ matrix that orients the angular momentum contribution of each reaction wheels with the body axes of the spacecraft. Rate limits are placed on each of the wheels that limit the maximum acceleration angular acceleration and angular velocity of the wheels that can be produced from the commanded acceleration given the current angular velocity of the reaction wheels. Integrator windup from the PID controller is abstracted out of this research and could be explored in future work.

3.2.5 Equations of Motion

In this section, the state derivative equations of motion for a system with RWA are presented.

The state vector $\bar{x}$ is defined here as

$$\bar{x} = \begin{bmatrix} \bar{q} & \vec{\omega} & \vec{\psi} \end{bmatrix}^T = \begin{bmatrix} q_1, q_2, q_3, q_4, \omega_1, \omega_2, \omega_3, \psi_1, \psi_2, \psi_3, \psi_4 \end{bmatrix}^T \tag{30}$$

where the q_i terms are the elements of the quaternion describing the orientation, the ω terms are the spacecraft angular velocity about each principal axis, and the ψ terms are the angular velocity of each of the reaction wheels. The time rate of change of the state can then be expressed as: [88][60]

$$\dot{\bar{x}} = \begin{bmatrix} \mathbf{0}_{4\times4} & 0.5\boldsymbol{Q}_{4\times3} & \mathbf{0}_{4\times4} \\ \mathbf{0}_{3\times4} & -\boldsymbol{I}^{-1}\omega^{\times}\boldsymbol{I} & -\boldsymbol{I}^{-1}\omega^{\times}\boldsymbol{S} \\ \mathbf{0}_{4\times4} & \mathbf{0}_{4\times3} & \mathbf{0}_{4\times4} \end{bmatrix} \bar{x} - \begin{bmatrix} \mathbf{0}_{4\times4} \\ \boldsymbol{I}^{-1} \\ \mathbf{0}_{4\times3} \end{bmatrix} \boldsymbol{S}\dot{\vec{\psi}} + \begin{bmatrix} \mathbf{0}_{4\times4} \\ \boldsymbol{I}^{-1} \\ \mathbf{0}_{4\times3} \end{bmatrix} \vec{M} + \begin{bmatrix} \mathbf{0}_{4\times4} \\ \mathbf{0}_{3\times4} \\ \boldsymbol{U}_{4\times4} \end{bmatrix} \dot{\vec{\psi}} \tag{31}$$

where $\vec{M}$ is a vector of external torques acting on the spacecraft body axes, $\boldsymbol{U}_{4\times4}$ is a 4×4 identity matrix, $\mathbf{0}_{n\times m}$ is a $n \times m$ matrix of zeros, and $\mathbf{Q}_{4\times3}$ is

$$\boldsymbol{Q}_{4\times3} = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix}. \tag{32}$$

IV. Methodology

Chapter IV begins by describing the verification strategy to show requirement satisfaction for the PID and RTA controller designs. Next, a set of hypothetical requirements are presented and then formally described for the 6U CubeSat attitude control subsystem. The strategy used to provide traceability of the requirements is described. Next, development and application of FMA techniques to the derived requirements, architecture, and model are described. Finally, techniques from traditional verification techniques, such as mathematical feasibility analysis (FA), SIM, and SFEDS, are presented. Between FMA, SIM, SFEDS, and FA requirement satisfaction may be shown for all requirements except those dealing with reachability.

4.1 Verification Strategy

In this section, a hypothetical set of CubeSat ACS requirements are presented, and the verification and traceability strategies used to show requirement satisfaction in each design phase are introduced.

4.1.1 Requirements Set

Hypothetical requirements are presented here for the 6U CubeSat highlighting the limitation of the actuators, the pointing accuracy of the control system, slewing range and rate, drift, settling time, rise time, and percent overshoot. The Defense Acquisition University (DAU) defines derived requirements as requirements that "arise from constraints, consideration of issues implied but not explicitly stated in the requirements baseline, factors introduced by the selected architecture, cyber security requirements and the design. Derived requirements are definitized through requirements analysis as part of the overall systems engineering process (SEP) and are part

of the allocated baseline."[93] The first two requirements in the set presented here are derived requirements that arise once a RWA is selected to actuate the satellite. Requirements R01 and R02 constrain the output of the control algorithm to prevent damage to the reaction wheel motors:

- R01 The commanded time rate of change of angular velocity shall not exceed the maximum allowable angular acceleration of the reaction wheel.
- R02 The commanded angular velocity shall not exceed the maximum allowable angular velocity of the reaction wheel.

The remaining requirements describe the expected performance of the attitude control subsystem in the context of its ability to control the 6U CubeSat attitude:

- R03 The pointing accuracy shall be at least 1 degree as a threshold and 0.08 degrees as an objective.
- R04 The pointing range about the z axis shall be 0 to 360 degrees.
- R05 The pointing range about the y axis shall be 0 to 360 degrees.
- R06 The pointing range about the x axis shall be 0 to 360 degrees.
- R07 The maximum slew rate shall be > 3 deg/sec as a threshold and > 7 deg/sec as an objective.
- R08 After settling, the drift rate shall be < 3 deg/min as a threshold and < 1 deg/min as an objective.
- R09 After settling, the total drift shall be $\leq$ 0.5 degrees as a threshold and $\leq$ 0.1 degrees as an objective.

- R10 The 5% settling time shall be $\leq$ 5 minutes as a threshold and $\leq$ 2 minutes as an objective.
- R11 The 2% settling time shall be $\leq$ 7 minutes as a threshold and $\leq$ 3 minutes as an objective.
- R12 The rise time shall be $\leq$ 5 minutes as a threshold and $\leq$ 2 minutes as an objective.
- R13 The percent overshoot shall be $\leq$ 50% as a threshold and $\leq$ 25% as an objective.

4.1.2 Requirement Analysis Stragety

Not all analysis techniques are appropriate for different types of requirements. For example, FMA is currently limited to analysis of linear systems with very little capability to analyze nonlinear systems. For this reason, FMA is best suited to showing satisfaction of the first two requirements, which are derived requirements that describe the rate limiting of the control actuators. Application of SFEDS provides insight, though not proof, into the satisfaction of the remaining requirements. The techniques applied to each of the requirements are summarized in Table 3.

The * on the SFEDS for requirements R04, R05, and R06 SFEDS indicates that these requirements are not analyzed in the main SFEDS. A seperate SFEDS for requirements R04, R05, and R06 analysis consists of batch simulations from an initial state of 0 degrees about each axis to final states from 0 to 360 degrees about the x, y, and z-axes, one at a time for a total of 1080 simulations.

Table 3. Strategy for Requirements Analysis

Req't	FA	FMA	SIM	SFEDS
R01		×	×	×
R02		×	×	×
R03			×	×
R04				×*
R05				×*
R06				×*
R07	×		×	×
R08			×	×
R09			×	×
R10			×	×
R11			×	×
R12			×	×
R13			×	×

4.1.3 Traceability of Requirements and Assumptions

Requirements traceability from the requirements, through the architecture, and into the modeling phase is achieved by a strict naming convention developed for this research. This naming convention was first introduced in Section 2.4.1. Each assumption and requirement is first assigned an `a` for assumption or an `r` for requirement. Next, the assumption or requirement is numbered with two digits, although with more than 99 assumptions or requirements, this number could be modified to include three or more digits. Next, if the requirement applies to more than one component, it is broken down for each component and assigned a unique letter to follow the requirement number (`a`, `b`, `c`, etc). Finally, a four letter description of the component that the requirement applies to is appended to the end, such as `ctrl` for controller. The initial letters, numbers, and descriptions of each requirement or assumption are separated by an underscore. For example, the first wheel sub requirement is documented as `r_01a_ctrl`, where `r` indicates that it is a requirement versus an assumption, `01` is the number of the requirement, `a` is the first sub requirement (it applies to wheel

1), and `ctrl` denotes that it is a controller component requirement.

4.2 Requirements Development and Formalization

In this section, a formal representation of each of the 13 hypothetical ACS requirements is discussed and values assigned to the formal requirements are presented.

4.2.1 Formal Representation of the Requirements

In order to analyze a requirement with a formal method, such as model checking, the requirement must be expressed a precise mathematical syntax, such as that provided by a temporal logic based language. The formal methods tools used in this research implement a scope and predicate pairing. The scope provides a temporal or conditional description of when the requirement should hold true. The predicate defines the condition of interest that should hold true within the defined scope.

For the following requirement definitions, it is assumed that only one maneuver is conducted, and the total scope of all of the requirements is the total time of the simulation. If however, it would be desired to analyze these requirements over multiple maneuvers in succession, the scope of all the requirements would be the time of a single maneuver which would restart when each new maneuver is commanded. The scope, and analysis of the requirements within that scope, would reset as soon as the next commanded orientation was received.

4.2.1.1 R01 and R02: Reaction Wheel Angular Acceleration and Velocity Limitation

Requirements R01 and R02 are examples of invariant properties because they are satisfied by showing that no reachable states of the system violate the requirement. For requirements R01 and R02, the scope is "always," because there is never a situa-

tion in which the reaction wheels should exceed the maximum acceleration or velocity. The predicate for R01 is that the acceleration commanded by the controller should never exceed the maximum acceleration for each reaction wheel ($|\dot{\psi}_{com}| \leq \dot{\psi}_{max}$), and the predicate for R02 is that the velocity resulting from the acceleration commanded by the controller should never exceed the maximum velocity for each reaction wheel ($|\psi_{pre} + \dot{\psi}_{com}\Delta t| \leq \psi_{max}$).

4.2.1.2 R03: Pointing Accuracy

A pointing accuracy requirement is another example of an invariant property because it is satisfied by showing that within the scope, no reachable states of the system violate the requirement. The scope of the pointing error requirement is after the system has settled ($t \geq T_s$). When the system has settled can be determined in a variety of ways including the time after the 5% or 2% settling time requirement has been met. For this research, settled is defined by a spacecraft angular acceleration and angular velocity range, where the angular acceleration must be less than approximately 0.01 degrees per second squared or 2.9089×10^{-6} radians per second squared ($||\dot{\vec{\omega}}|| \leq 2.9089 \times 10^{-6}$) and the spacecraft angular velocity must be less than 1 degree per minute or approximately 2.9089×10^{-4} radians per second ($||\vec{\omega}|| \leq 2.9089 \times 10^{-4}$). The first step in mathematically stating a pointing accuracy predicate is defining the pointing error.

The pointing error is derived from the definition of the quaternion which contains a 3x1 vector component $\vec{q}_{123}$ and a scalar component q_4 which is calculated from

$$q_{(4)} = \cos\left(\frac{\Phi}{2}\right) \tag{33}$$

where Φ is the Euler angle. Eqn. 33 can be rearranged to calculate the Euler angle Φ from the quaternion with

$$\Phi = 2\text{acos}(q_4), \tag{34}$$

which is desirable if the model uses quaternions to update the orientation state, like the model used in this research. The pointing error e_p may be calculated by subtracting the final Euler angle Φ_f from the current Euler angle Φ, written here as

$$e_p = \Phi - \Phi_f. \tag{35}$$

The predicate for pointing accuracy states that the pointing error e_p must not exceed the pointing requirement e_r ($e_p \leq e_r$).

4.2.1.3 R04-R06: Reachability

Requirements R04, R05, and R06 essentially deal with reachability. For a linear system, all states could be said to be reachable if the system is completely controllable. This can be determined by calculating the rank of the controllability matrix. If the rank of the controllability matrix M_c, defined as

$$M_c = [B, AB, ..., A^{\text{n-1}}B]_{n\times(nl-l^2+1)}, \tag{36}$$

is n for a system with l inputs, then the system is completely controllable. For nonlinear systems, determining controllability is a non-trivial task. However, tools such as Flow* [27] have emerged in the last few years to aid in calculation of reachable states for nonlinear systems. In this research, insight is gained into whether the system is completely controllable by determining if the pointing error at the end of the simulated maneuver $e_p(t_f)$ is less than a pointing error requirement e_r ($e_p(t_f) \leq e_r$). In simulation, the scope of this maneuver is the final time ($t = t_f$). In a simulation with no noise, like those conducted in this research, the final pointing error at the

end of the simulation should be zero.

4.2.1.4 R07: Maximum Slew Rate

A maximum slew rate requirement is an example of a liveness property because it is met if at least one reachable state satisfies the requirement. In simulation, the slew rate requirement is satisfied if it is eventually reached. The slew rate is the magnitude of the spacecraft's instantaneous angular velocity during a slewing maneuver. The scope of the slew rate would be from the start of the maneuver until the spacecraft is settled ($t \leq T_s$). To formally prove that a slew rate requirement is met, it must be proven that at some point during a slewing maneuver from one position at rest to another position at rest the angular velocity of the spacecraft will exceed the minimum slew rate stated in the requirement. A formal verification method such as model checking is not a good verification method for a slew rate requirement because there are many maneuvers in which it would be desirable for the angular velocity of the spacecraft not to exceed the slew rate. For instance if the spacecraft is required to change the pointing angle by one degree, reaching the slew rate specified by the requirement during the maneuver would overshoot the desired angle and take longer to settle.

The true concern being expressed in a maximum slew rate requirement is whether the actuator selected is appropriately sized to be responsive to reorientation commands. Therefore, the best ways to show satisfaction of a slew rate requirement are to show it is mathematically possible to achieve the desired slew rate, or to demonstrate a case in which the spacecraft angular velocity appropriately exceeds the slew rate requirement, or to analyze the maximum slew rate for a range of initial and final conditions in a SFEDS. For a RWA attitude actuator, it can be shown that a desired slew rate is possible by analyzing the maximum angular momentum the RWA can

generate. RWAs actuate the satellite through momentum exchange, where the change in angular momentum of the RWA is equal and opposite to the change in angular momentum of the spacecraft. The maximum angular momentum that the reaction wheels can generate would occur when the reaction wheels start at their maximum velocity in one direction and accelerate to their maximum velocity in the opposite direction.

Given a selected RWA design, the RWA and spacecraft angular momentum equations can be set equal to one another rearranged to solve for maximum achievable spacecraft angular velocity $\vec{\omega}_{max}$ from rest using

$$\vec{\omega}_{max} = \boldsymbol{I}^{-1}\boldsymbol{S}\vec{\psi}_{max}. \tag{37}$$

Satisfaction of the slew rate requirement is shown by comparing the maximum achievable spacecraft angular velocity to the required slew rate ω_r to show that the maximum velocity from rest eventually meets or exceeds that required ($||\vec{\omega}_{max}|| \geq \omega_r$).

4.2.1.5 R08 and R09: Drift Rate and Total Drift

Requirements R08 and R09 are additional examples of invariant properties because they are satisfied by showing that within scope, no reachable states of the system violate the requirement. Drift rate is a measure of how quickly a system deviates from the commanded orientation once the commanded orientation has been achieved and total drift is the amount the spacecraft orientation has deviated from the commanded orientation. Like the pointing accuracy requirement, the scope of drift rate and total drift requirements is the time after the spacecraft has settled ($t \geq T_s$). The drift rate is the angular velocity after the spacecraft has settled, and the total drift is the pointing error after the spacecraft has settled.

Drift rate and total drift are concerns in cyber-physical systems where measure-

ment errors in physical sensors and true continuous dynamics make constantly maintaining an exact desired position impossible with a discrete control algorithm. Attempting to maintain the position exactly may result in a series of very abrupt maneuvers that create high frequency noise as the satellite constantly attempts to jerk right back to the correct position. Rather than maintaining an exact orientation, a deadband is defined so that no action will take place within a certain range of the correct position. Small drift rates and amounts of drift should not impact the mission as long as they remain within the drift rate ω_{dr} and total drift e_{tdr} requirements; therefore, the predicate for the drift rate requirement states that the spacecraft angular velocity should not exceed the drift rate ($||\vec{\omega}|| \leq \omega_{dr}$) and the predicate for total drift states that the spacecraft pointing error should not exceed the total drift requirement ($e_p \leq e_{tdr}$).

4.2.1.6 R10 and R011: 5% and 2% Settling Time

Traditionally, 5% and 2% settling time are used to describe when the response of a second order system to a step input reaches and stays within 5% or 2% of the steady state value, and may be estimated by a function of the systems damping ratio ζ and natural frequency ω_n from

$$T_{s2\%} = \frac{\ln(0.02\sqrt{1-\zeta^2})}{\zeta\omega_n} \tag{38}$$

and

$$T_{s5\%} = \frac{\ln(0.05\sqrt{1-\zeta^2})}{\zeta\omega_n}. \tag{39}$$

Since the system under analysis is not a linear, second order system, the traditional estimation equations Eq.(38) and Eq.(39) do not apply. The system must be simulated in order to determine the settling time. In this research, the settling time definitions

are generalized to when the orientation of the spacecraft reaches and stays within 5% or 2% of the desired final orientation. The 2% error $e_{2\%}$ and 5% error $e_{5\%}$ ranges are based on the initial pointing error angle between the initial and final positions $e_p(t_0)$.

$$e_{2\%} = (0.02)2\text{acos}\Big(2e_p(t_0)\frac{180}{\pi}\Big) \tag{40}$$

$$e_{5\%} = (0.05)2\text{acos}\Big(2e_p(t_0)\frac{180}{\pi}\Big) \tag{41}$$

As the settling time is a temporal property, it does not have a temporal scope like other requirements described. The predicate is that the 5% or 2% settling time does not exceed the required settling time, respectively ($T_{s2\%} \leq T_{s2\%r}$ and $T_{s5\%} \leq T_{s5\%r}$).

4.2.1.7 R12: Rise Time

Like settling time, rise time is typically used to describe the time of the first peak T_p of the step response of a linear, second order system. The rise time requirement T_{rt} is a way to specify the responsiveness of the system. This definition is generalized here to first peak of the pointing error of the spacecraft, which eventually reaches a steady state value of 0 with some deadband when the maneuver is complete. The first peak is determined algorithmically by finding the first inflection point in the pointing error data. The rise time also doesn't have a temporal scope, like the settling time requirements. The predicate for the rise time states that the time of the first pointing error peak does not exceed the required rise time ($T_p \leq T_{rt}$).

4.2.1.8 R13: Percent Overshoot

In traditional control systems analysis, the percent overshoot is linked to the peak time T_p, and describes the height of the first peak as a percentage of the steady-

state response. For a step response from 0 to a steady-state value of 1, the percent overshoot $\%OS$ can be estimated as a function of the system's damping ratio ζ from

$$\%OS = e^{(\zeta\pi/\sqrt{1-\zeta^2})} \times 100\%. \tag{42}$$

This estimate is not appropriate for the attitude control system in this research because the system being evaluated in this research is not a second order, linear, or performing a step response. Because the percent overshoot requirement $\%OS_r$ is primarily concerned with how far the spacecraft passes the steady state value once it has been reached the first time, the pointing error of the system is used to analyze this requirement. For the purposes of this research, the overshoot is defined as the magnitude of the first peak in pointing error. The time T_p and magnitude of the first peak $e_p(T_p)$ are determined algorithmically at the inflection point of the first peak. Because the pointing error is at its maximum at the start of the maneuver, the starting error $e_p(t_0)$ is used to normalize the percent overshoot in

$$\%OS = e_p(T_p)/e_p(t_0) \times 100\%. \tag{43}$$

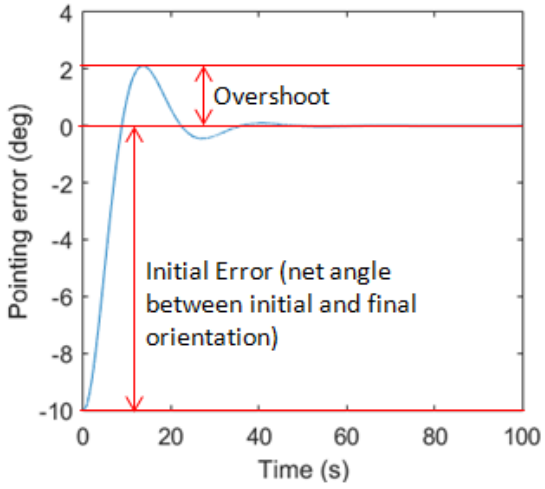

Figure 21. Overshoot and Initial Error $e_p(t_0)$ for an Example Slewing Maneuver

The overshoot and initial error $e_p(t_0)$ for an example slewing maneuver may be seen in Figure 21. Like the settling time and rise time requirements, percent overshoot does not have a specific scope. The predicate of the requirement is that the percent overshoot must be less than the percent overshoot requirement ($\%OS \leq \%OS_r$).

4.2.1.9 Requirements Formalization Summary

As stated earlier, it is assumed that the larger scope of all of the requirements is from the time that a maneuver is commanded to the time that a new maneuver is commanded. Within the maneuver scope, each requirement may have a smaller scope. The applicable scope and predicate of each requirement that were formalized in this section are summarized in Table 4.

Table 4. Requirement Scope and Predicate Summary

Req't	Scope	Predicate
R01	always	$\lvert\dot{\psi}_{com}\rvert \leq \dot{\psi}_{max}$
R02	always	$\psi_{pre} + \dot{\psi}_{com}\Delta t\rvert \leq \psi_{max}$
R03	$t \geq T_s$	$e_p \leq e_r$
R04*	$t = t_f$	$e_p \leq e_r$
R05*	$t = t_f$	$e_p \leq e_r$
R06*	$t = t_f$	$e_p \leq e_r$
R07	$t \leq T_s$	eventually $\lvert\lvert\vec{\omega}_{max}\rvert\rvert \geq \omega_r$
R08	$t \geq T_s$	$\lvert\lvert\vec{\omega}\rvert\rvert \leq \omega_{dr}$
R09	$t \geq T_s$	$e_p \leq e_{tdr}$
R10	-	$T_{s2\%} \leq T_{s2\%r}$
R11	-	$T_{s5\%} \leq T_{s5\%r}$
R12	-	$T_{\omega_{max}} \leq T_{rt}$
R13	-	$\%OS \leq \%OS_r$

The * on requirements R04-R06 indicate that the scope and predicate given are only valid for simulation cases, and may not be used to demonstrate proof that these requirements are met.

4.2.2 Values Assigned to the Formal Requirements

The values of the requirements used to evaluate the ACS in this research are shown in Table 5.

Table 5. Formalized Requirement Values

Requirement	Variable	Description	Value	Units
R01	$\dot{\psi}_{max}$	Maximum reaction wheel angular acceleration	181.3	rad/s^2
R02	ψ_{max}	Maximum reaction wheel angular velocity	576.0	rad/s
R03	e_{r_t}	Pointing accuracy threshold	1	deg
R03	e_{r_o}	Pointing accuracy objective	0.1	deg
R07	ω_{r_t}	Slew rate threshold	3	deg/s
R07	ω_{r_o}	Slew rate objective	7	deg/s
R08	ω_{dr_t}	Drift rate threshold	3	deg/min
R08	ω_{dr_0}	Drift rate objective	1	deg/min
R09	e_{tdr_t}	Total drift threshold	0.5	deg
R09	e_{tdr_o}	Total drift objective	0.1	deg
R10	$T_{s5\%r_t}$	5 percent settling time threshold	5	min
R10	$T_{s5\%r_o}$	5 percent settling time objective	2	min
R11	$T_{s2\%r_t}$	2 percent settling time threshold	7	min
R11	$T_{s2\%r_o}$	2 percent settling time objective	3	min
R12	T_{rt_t}	Rise time threshold	5	min
R12	T_{rt_o}	Rise time objective	2	min
R13	$\%OS_{r_t}$	Percent overshoot threshold	50	%
R13	$\%OS_{r_o}$	Percent overshoot objective	25	%

4.3 Architecture Development and Formal Verification

In this section, the compositional structure of the ACS architecture and the dynamics of the spacecraft controller are presented. Descriptions of the PID and RTA control architectures as well as the design of the RTA controller's decision module are provided. Finally, the safety properties of the PID and RTA controllers, the approach used to verify that the safety properties are satisfied by the PID and RTA controller implementations, and assumptions used in the FMA verification are provided.

4.3.1 Compositional Architectecture

Breaking up a system by components to the lowest compositional level enables formal methods analysis at the lowest functional levels of the design to be composed into subsystem and system level certification arguments. As was introduced in Sections 2.1.4 and 2.3, a compositional architecture is critical for efficient FMA. In addition, a compositional structure promotes modularity and reuse of verification results by allowing components such as an actuator or sensor to be changed and analysis to be completed on directly impacted portions of the system only up through the subsystem to the system level rather than completing all of the analysis for the entire system again.

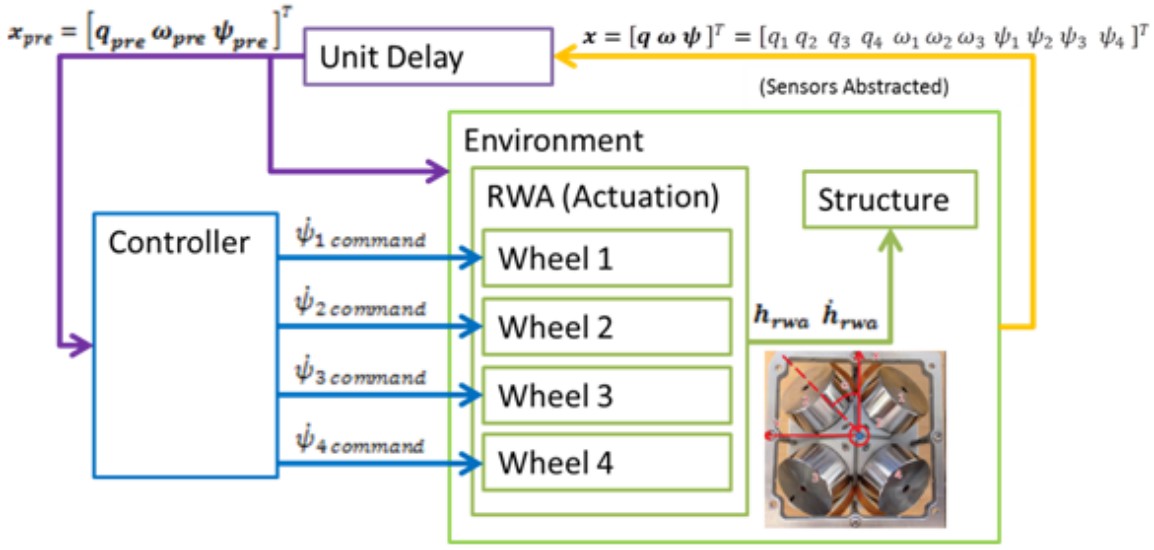

Figure 22. 6U CubeSat Singal Flow for Model with RWA Control Actuator

For this research, the satellite is broken up compositionally into a controller, which contains the controller software, and the environment, which includes all the hardware that the controller interacts with. The environment is further functionally decomposed into the RWA and structure models. The sensors have been abstracted and it is assumed that the controller receives perfect truth data, which could be modified in future work. The reaction wheel is broken down one more level to individual reaction

wheel models. The flow of signals between the components is depicted in Figure 22. The simulated environment outputs the current system state including the orientation of the spacecraft in a quaternion vector, the rotation rates of each of the major axes of the spacecraft, and the rotation rates of each of the individual reaction wheels. At each step in the simulation of the system, a unit delay is placed between the output of the environment and the controller. The controller receives the state information from the previous time step and uses it to calculate a commanded angular acceleration. In the basic model, a PID controller calculates a commanded angular acceleration for each of the reaction wheels in order to approach and eventually achieve the desired spacecraft orientation. The alternative RTA controller architecture is described in the Section 4.3.2. The angular acceleration commands are received and implemented by each of the individual reaction wheels. The reaction wheels then contribute angular momentum to the RWA, which exchanges its total angular momentum with the spacecraft structure. The simulated structure model then updates the state of the spacecraft and the environment model outputs the system state.

4.3.2 Run Time Assurance Architecture

In the RTA control architecture, the controller block of the previous section is replaced by an RTA controller that is decomposed into the unverified controller, verified controller, and decision module, as described in Section 2.6.3. The unverified controller provides the primary control commands of the RTA control architecture and could contain any control design, including controllers that use artificial intelligence and machine learning. The verified controller provides the backup control commands and contains a rate-limited version of the PID controller described in Section 3.2.4, which may be proven using formal methods not to violate the two safety properties governing reaction wheel angular velocity and acceleration, as is shown in Section 5.1.

The decision module contains the logic which determines which control signal, verified or unverified, is sent to the actuator. The system state is received by the unverified controller, verified controller, and decision module. The decision module also receives a commanded reaction wheel acceleration vector from each of the controllers. The signal flow in the RTA controller is summarized in Figure 23.

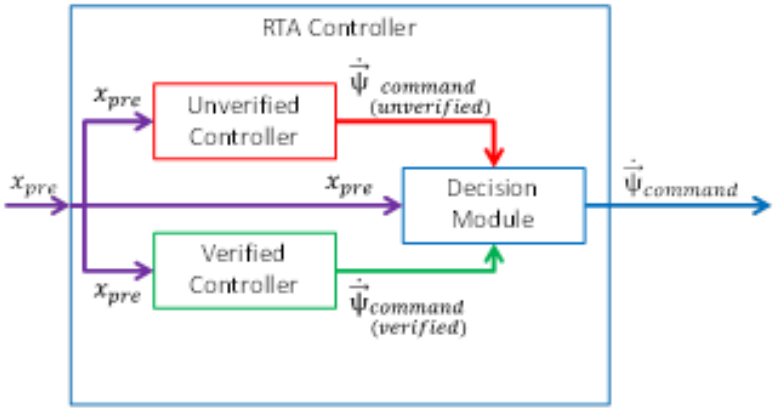

Figure 23. Signal Flow in the RTA Controller

4.3.3 Decision Module Design

The key component of the RTA controller is the decision module, which can be thought of as a switching mechanism. The decision module receives the current state of the system and the reaction wheel angular acceleration commands from each of the two controllers. Aside from allowing the use of an intelligent or learning controller, an RTA controller could be used to safely generate proof that an unproven control system will not violate specific constraints over the course of hundreds, thousands, or more hours of performance. After a period of time in service, a sufficient amount of evidence may be generated that an unverified controller will not violate the safety properties, and the decision module and verified controller backup could potentially be removed.

In this research, desired safety properties are used to constrain the operating re-

gion of the unverified controller and inform the decision to switch to a verified backup controller. The safety properties are arranged in if-else logic, which provides several advantages. During the design process, a designer may want to check to see if the unverified controller violates other properties of interest, such as performance properties, beyond those used to constrain the operating envelope of the unverified controller for safety reasons. An if-else decision tree enables the addition of a new constraint to the unverified controller without compromising the other system constraints. In addition, breaking properties into like groups in if-else statements allows reviewers to quickly inspect, debug and isolate problems in the code.

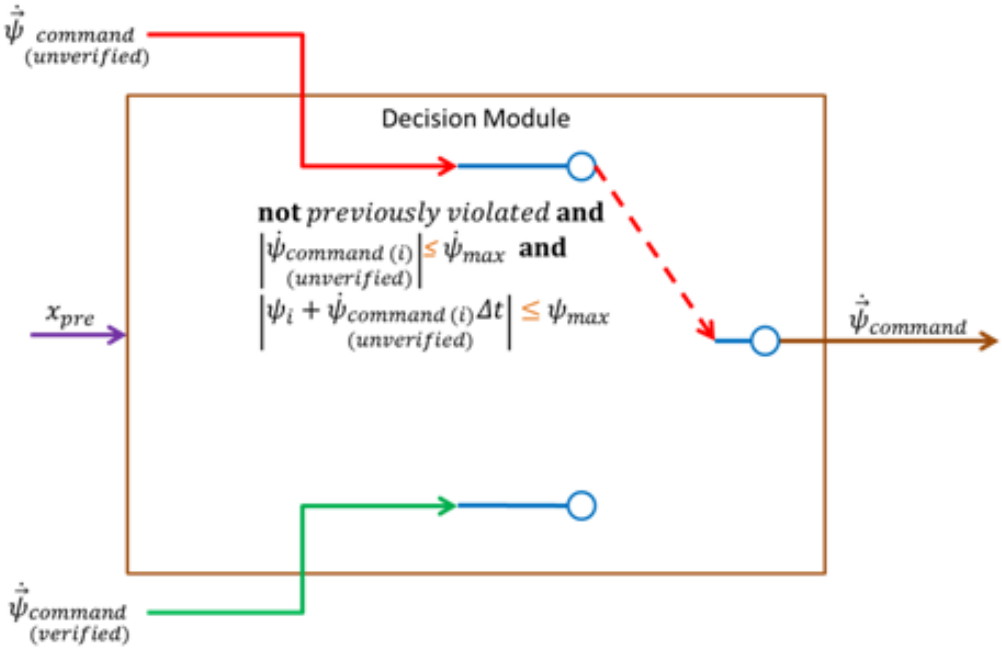

Figure 24. Signal Flow in Decision Module when the Safety Properties of the System are not Violated by the Unverified Controller

In this implementation, the decision module first checks to see if the unverified controller has violated any of the safety properties in the past. The rationale for checking previous instances of violations first is that if the unverified controller has previously violated a safety property, it cannot be trusted until a reset, redesign, or other change is implemented. Mode logic to incorporate unverified controller resets is

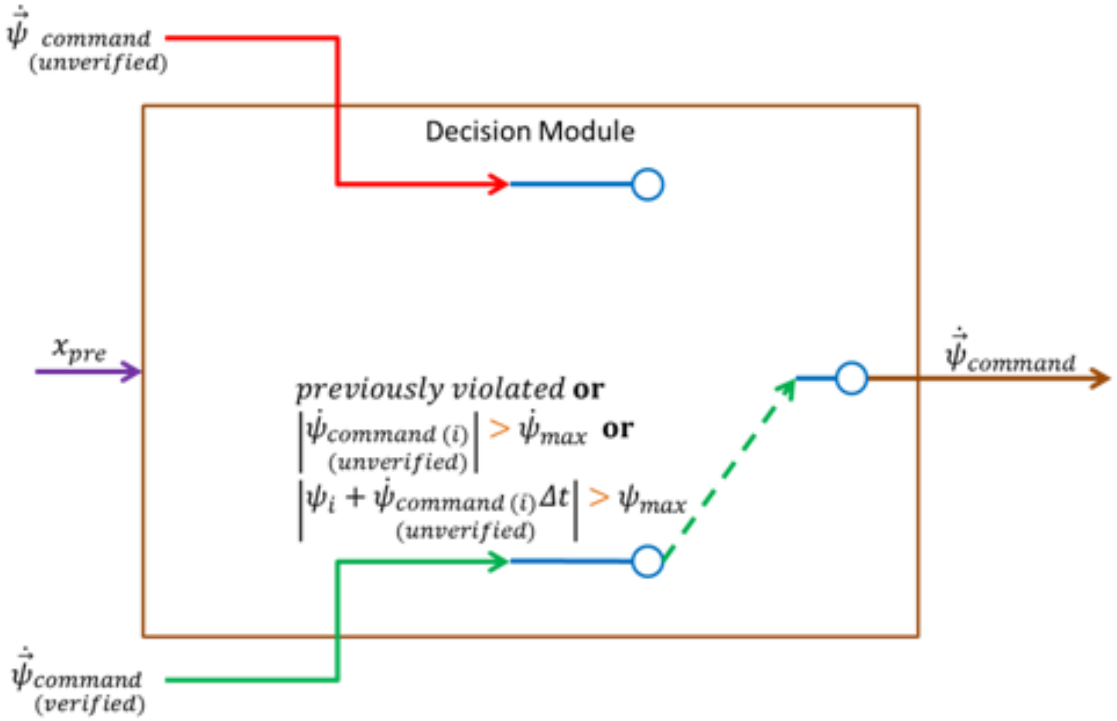

Figure 25. Signal Flow in Decision Module when the Safety Properties of the System are Violated by the Unverified Controller

left for future work. Second, the decision module then checks to see if the unverified controller violates the maximum commanded acceleration property for any of the four wheels in the array. Third, the decision module checks that the commanded angular acceleration will not result in a predicted angular velocity that exceeds the maximum velocity property for any of the four wheels. As depicted in Figure 24, if no violation of the safety properties has occurred, and the commanded angular acceleration to each reaction wheel from the unverified controller does not exceed the maximum allowable limit, and the angular velocity of each wheel resulting from the commanded angular acceleration does not exceed the maximum allowable angular velocity of the reaction wheel array, then the decision module allows the output from the unverified controller to pass through to the actuators. If however, a violation has previously occurred, or the output command of the unverified controller exceeds angular acceleration limits, or the commanded acceleration will result in an angular velocity limit violation in any

of the wheels, then the decision module sends the output of the verified controller to the actuators, as depicted in Figure 25.

4.3.4 Safety Properties Used in Formal Methods Analysis

In this research, formal methods are used to prove that the controller will not violate two safety properties: the maximum angular velocity and the maximum angular acceleration of the reaction wheels, as presented in Figure 26, where the subscript i corresponds to the individual reaction wheels. These two safety properties are derived requirements identified after an RWA is selected as the control actuator and are designed to protect the motors of the reaction wheel assembly from damage. These derived requirements, used as safety properties in the analysis are requirements R01 and R02 in Section 4.1.1. In this work, these two safety properties are used to formally analyze the requirements, architecture, and model of the RTA controller. When applied to each of the 4 reaction wheels, a total of 8 safety properties are evaluated in the output of the RTA controller and its components.

Safety Properties:
1) $\left|\dot{\psi}_i\right| \leq \dot{\psi}_{max}$
2) $\left|\psi_i + \dot{\psi}_i \Delta t\right| \leq \psi_{max}$
For $i = 1, 2, 3, 4$

Figure 26. Desired Safety Properties of the Controller

It is possible that timing delays and drift from the unit delays used in the architecture to sample the system state could result in violation of the second safety property in a continuous physical system. In this research the plant and controller are both simulated with the same discrete timestep. Future work could investigate the impact of the sampling delays on violation of the second property, and possibly use more conservative switching logic in the decision module.

4.3.5 Formal Methods Verification Approach for RTA Controller Requirements and Architecture

As explained in Section 2.6.3, the unverified controller, which is treated as a "black box" in this research cannot be proven to hold the desired safety properties in all cases using formal methods. The verified controller, however, should be proven using formal methods or an alternative verification method, such as extensive simulation, flight test, or a long period of operational service. In this instance of the RTA controller, the verified controller is analyzed using formal methods to prove that it never violates the two safety properties. Similarly, the decision module is analyzed using formal methods to prove that the constraints on the unverified controller are sufficient to guarantee that decision module output never violates the controller safety properties. The safety properties are analyzed against a model developed from the formalized requirements in SpeAR, and against an architecture implementation developed in AADL by AGREE. In SpeAR and AGREE, the proof evidence from each of the RTA controller subcomponents is combined compositionally to produce proof evidence that the RTA controller subsystem will not violate the safety properties. Figure 27 summarizes which outputs of the RTA controller and its subcomponents are formally verified in this approach with a green check mark.

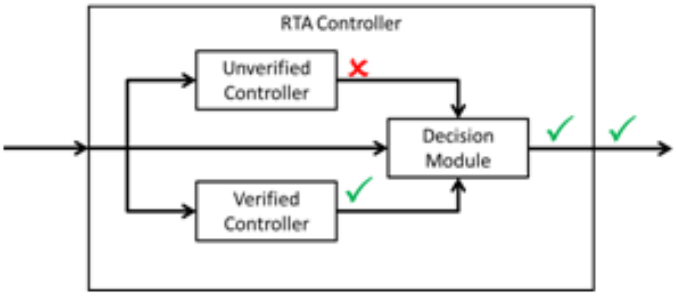

Figure 27. Expected Formal Verification Analysis Results for the RTA Controller and Subcomponents

Limitations in SLDV prevent analysis at the RTA subsystem level, however SLDV

may be used to prove that the safety properties are held by the model of the verified controller and decision module.

4.3.6 Assumptions

The nature of model checking requires assumptions to be made which constrain its searchable space. Without constraints, the model checker may assign any value of the appropriate data type to the signals entering the component of interest, without regard to whether meaning of these numbers is physically possible. For example, the model checker may assign an altitude of 30,000 feet and then instantaneously assign an altitude of 100,000 feet in the next time step, without an assumption constraining the maximum climb rate of the aerospace model. In addition to constraining the search space to physically possible solutions and reducing computational demands, formal assumptions provide precise documentation of design assumptions that may be evaluated by SMEs. Assumptions should be used with caution, as they can over constrain the search space and ignore critical areas where the system under analysis could enter. In this work, three assumptions are made about the decision module input in order to constrain the model checker. First, it is assumed that the previous reaction wheel speeds entering the decision module do not exceed the maximum reaction wheel speeds. Mode logic to deal with the eventuality that the reaction wheels are spinning faster than allowable at system initialization is left for future work. Second and third, it is assumed that the verified controller output does not violate the two safety properties governing maximum velocity or acceleration. The mathematical expression of these assumptions is presented in Figure 28. This assumption is valid because the rate-limited PID controller is formally proven to meet these properties as a part of this analysis. In general, assuming that inputs to the decision module meet properties should only occur if proof of that claim exists, otherwise the assumption

would unnecessarily constrain the model checker.

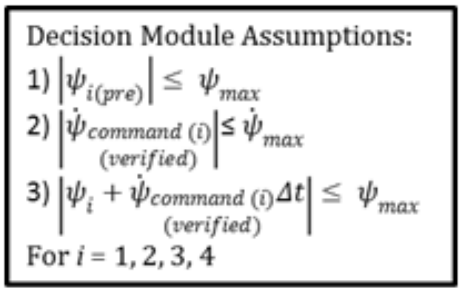

Figure 28. Assumptions used by the Decision Module to Constrain the Formal Methods Analysis Space

4.4 Model Development and Verification

The verified architecture provides the framework for a Simulink model and simulation of the control system; however the modeling stage is the first time that the nonlinear equations that govern the controller and state equations are used. In Simulink, like in SpeAR and AGREE, the model was broken down from the system level into sublevels of the Controller and Environment. The file structure in Simulink was designed to mimic the architectural breakdown. Simulink model reference blocks were used to reference each of the subsystems and compose the entire system. Three files are included in each of the subsystem folders: the Simulink model of the subsystem, a model of the properties of the subsystem, and a verification model that references the subsystem and properties models with model reference blocks in order to conduct the formal analysis. The file structure is shown in Figure 29. The benefit of using model reference and this file structure is the ability to quickly to substitute an alternative implementation for a subsystem to compare performance both in simulation and verification analysis.

In Figure 29, a MATLAB m-file of constants of the simulation, a controller model, an environment model, and a simulation model are present at the system level. De-

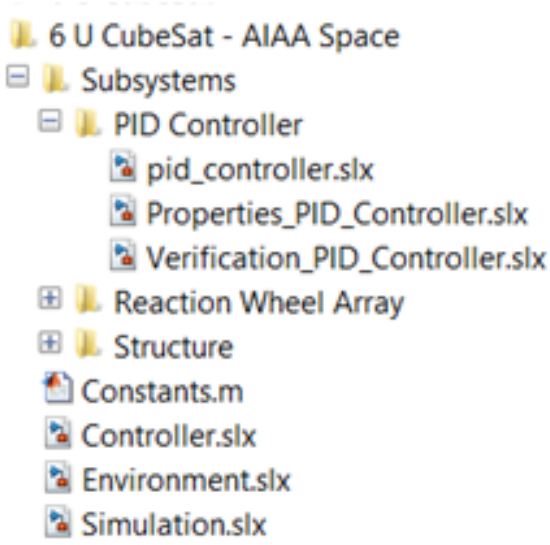

Figure 29. Basic Simulink Model File Structure

pending on the size of the model, it may be desirable to place the controller and environment in their own subsystem folders. Within each component folder is a component model that describes the behavior of that component, a properties model that contains the assumptions and requirements to be proven about that component, and a verification model that references the component model and properties model to conduct formal methods analysis of whether the component implementation meets the requirements. The remainder of this section will explore the verification and design of the models.

4.4.1 Component Verification with Simulink Design Verifier (SLDV)

Each requirement must be assigned a verification proof objective in SLDV. This was done by writing the all the requirements like that shown in Figure 8 in a user-defined function block, labeled the properties model, for each component that is to be analyzed. The inputs of this properties model are the inputs and outputs of the component of interest and the outputs of the properties model are the assumptions on the inputs of the component being analyzed and a proof objective associated with

each of the requirements. SLDV has special assumption and proof objective blocks that may be placed on the output of the verification block for the analysis. The outputs of the properties model may also be plotted so that the designer has a quick time history of any property violations. A verification model combines the model of the component, labeled a functional model, with the properties model for analysis. An example of a verification model is presented in Figure 30.

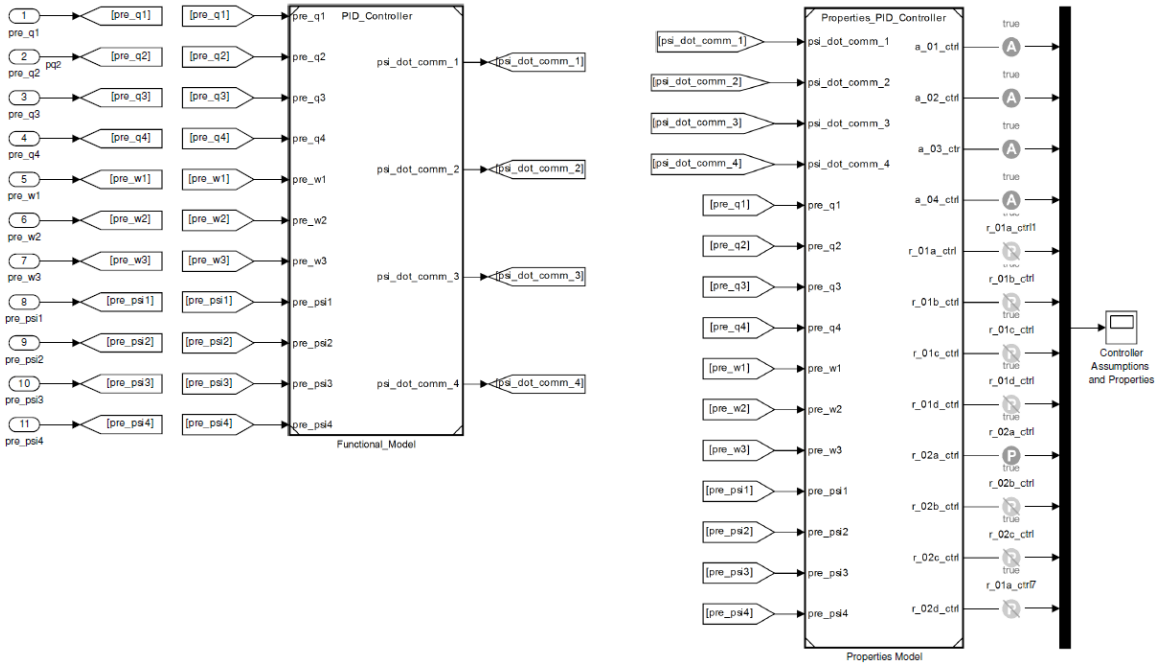

Figure 30. PID Controller Verification Model

In Figure 30 the inputs to the verification model are on the far left side. These inputs are the elements of the state vector for the system. Each of these inputs feed into the functional model block, which is a Simulink model reference block that references the model of the PID controller, and the properties model block, which is a Simulink model reference block back to the user defined function that contains all of the assumptions and requirements that are to be proven for the component in the analysis. The outputs of the functional model block are also fed into the properties model. At each of the outputs of the properties model is either a SLDV assumption

block or proof objective block, represented as a gray circle with an “A” or “P” at the center, respectively. Proof objective blocks that are dark gray are enabled, while proof objective blocks that are light gray with a diagonal line through them are disabled. Depending on the number of properties (requirements) that are to be proven about the model, and the size and complexity of the model, it may be desirable to only prove a few properties at a time. In Matlab 2015a and earlier versions, a bug is present that could prevent all the properties being proven at once without error. In addition, proving less properties could result in a faster analysis time. Finally, in Figure 30 all the outputs are plotted in a scope block. Formal methods verification in SLDV is completed on a discretized model and controller with a fixed timestep.

4.4.2 System Level Simulation

At the system level, a simulation model uses model reference blocks to connect to the controller and environment and simulate their interaction. The simulation model is shown in Figure 31. Starting on the left side of Figure 31, the state of the system

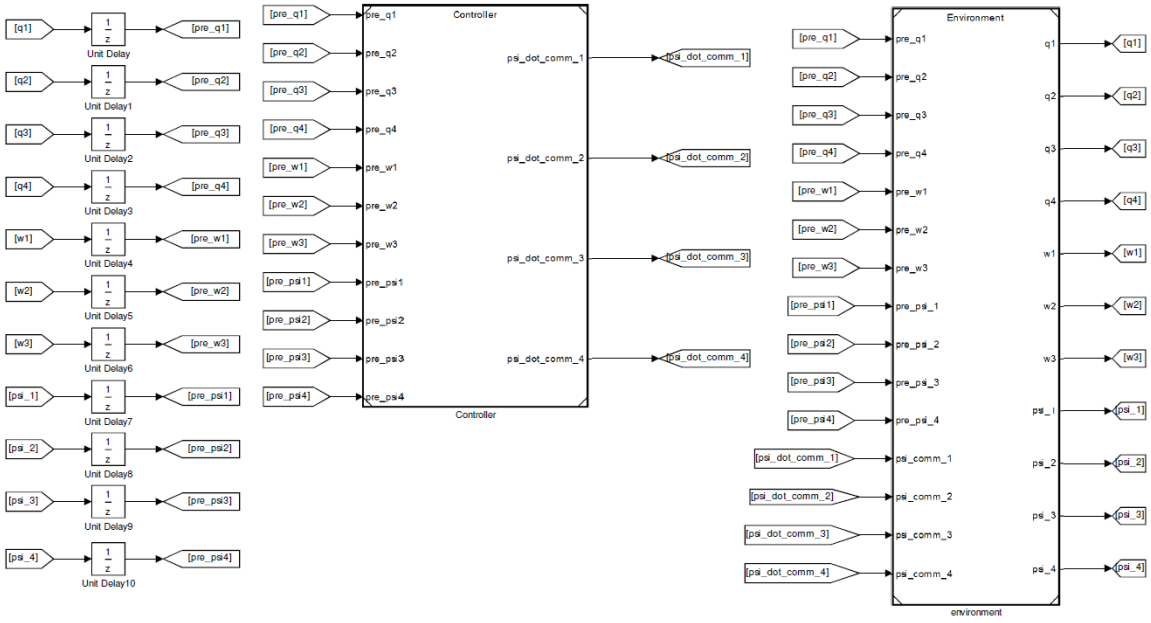

Figure 31. Simulation Model of the 6U CubeSat Attitude Control Performance

is fed into unit delay blocks to prevent a cyclic relationship between the environment and component that results in unexpected behaviors and errors. The delay could be placed anywhere between the controller and environment, but is placed just prior to the controller in this model to represent the delay associated with sensing the system state. The output of each of the unit delay blocks is a previous state component. The previous state is then input into the controller which calculates and outputs a commanded angular acceleration for each of the four reaction wheel actuators. The commanded angular accelerations along with the previous state are then input into the environment which updates and outputs the state of the system.

4.4.3 Model Assumptions and Limitations

Where possible, properties of the AFIT 6U CubeSat Bus and ACS, such as moments of inertias and wheel speeds are used. It is assumed that no external torques, such as gravity gradient or aero torques, are acting on the CubeSat. It is also assumed that the CubeSat rotational motion can be described by rigid body dynamics. In addition, the numerical simulation is discretized and conducted at fixed time step. The 3×3 moment of inertia matrix of the 6U CubeSat is taken from the AFIT SolidWorks model. The moment of inertia about the reaction wheel's spin axis was measured using the Model XR250 Measurement Instrument[66] and the motor properties were taken from the Maxon motor specification sheet.[72] Characterization of the hardware may be found in Chapter VII.

4.4.4 PID Controller Model

The PID controller model accepts the system state and calculates and outputs a commanded angular acceleration for each of the reaction wheels and is implemented as a user-defined function block, with the code presented in Figure 32. The code

```
function [psi_dot_comm_1, psi_dot_comm_2, psi_dot_comm_3,psi_dot_comm_4] = fcn(pre_q1,↙
pre_q2,pre_q3,pre_q4,pre_w1,pre_w2,pre_w3,pre_psi1,pre_psi2,pre_psi3,pre_psi4,D,M_tilde,↙
Kp,I,Kd,psi_max,psi_dot_max,delta_t,S_pinv,S)
% 14 Aug 2015
% States at the end of the last time step
    q       = [pre_q1,pre_q2,pre_q3,pre_q4];    %s/c quaternion
    omega   = [pre_w1,pre_w2,pre_w3];           %s/c angular rates
    psi     = [pre_psi1,pre_psi2,pre_psi3,pre_psi4];     %RWA angle rates

    % Compute reaction wheel array angular momentum
    h_rwa = S*psi';

    % Error between current quaternion and desired final quaternion
    q_error = M_tilde*q';

    % Computed control
    h_dot_rwa = Kp*I*q_error(1:3,1)+Kd*I*omega'-skew(omega')*(I*omega'+h_rwa);
    psi_dot = S_pinv*h_dot_rwa;
    psi_dot_comm_1 = psi_dot(1);
    psi_dot_comm_2 = psi_dot(2);
    psi_dot_comm_3 = psi_dot(3);
    psi_dot_comm_4 = psi_dot(4);
```

Figure 32. PID Controller MATLAB Code

first calculates the reaction wheel array angular momentum, then calculates the error between the current and desired quaternion. Using this information, a required change of angular momentum is calculated and then used to calculate a commanded angular acceleration for each of the reaction wheels. A rate limiting scheme based on R01 and R02 is included below the code shown in Figure 32. This rate limiting code is shown in Figure 33.

```
% Check that commanded psi_dot does not exceed max_psi_dot limitation
% of the RW, and if it does, set to the max
if abs(psi_dot_comm_1) >= psi_dot_max
    psi_dot_comm_1 = sign(psi_dot_comm_1)*psi_dot_max;
end
if abs(psi_dot_comm_2) >= psi_dot_max
    psi_dot_comm_2 = sign(psi_dot_comm_2)*psi_dot_max;
end
if abs(psi_dot_comm_3) >= psi_dot_max
    psi_dot_comm_3 = sign(psi_dot_comm_3)*psi_dot_max;
end
if abs(psi_dot_comm_4) >= psi_dot_max
    psi_dot_comm_4 = sign(psi_dot_comm_4)*psi_dot_max;
end
% Check if the commanded psi_dot would exceed max psi limitation, if so
% set the commanded psi_dot to 0
if abs(psi(1)+ psi_dot_comm_1*delta_t) >= psi_max
    psi_dot_comm_1 = 0;
end
if abs(psi(2)+ psi_dot_comm_2*delta_t) >= psi_max
    psi_dot_comm_2 = 0;
end
if abs(psi(3)+ psi_dot_comm_3*delta_t) >= psi_max
    psi_dot_comm_3 = 0;
end
if abs(psi(4)+ psi_dot_comm_4*delta_t) >= psi_max
    psi_dot_comm_4 = 0;
end
```

Figure 33. Controller Rate Limiting MATLAB Code

The rate limiting code functions by comparing the commanded acceleration to the max acceleration and limiting the command to the max if the commanded acceleration exceeds the maximum. It also checks to see whether the commanded acceleration will result in a reaction wheel angular velocity in excess of the maximum velocity, and if the maximum velocity will be exceeded, the commanded acceleration for that wheel is set to 0.

4.4.5 Environment Model

The environment is separated into the reaction wheel array actuator and the spacecraft structure. The Simulink model of the environment is shown in Figure 34. As stated previously, the input to the environment is the previous state and the reaction wheel angular acceleration commands from the controller. The previous

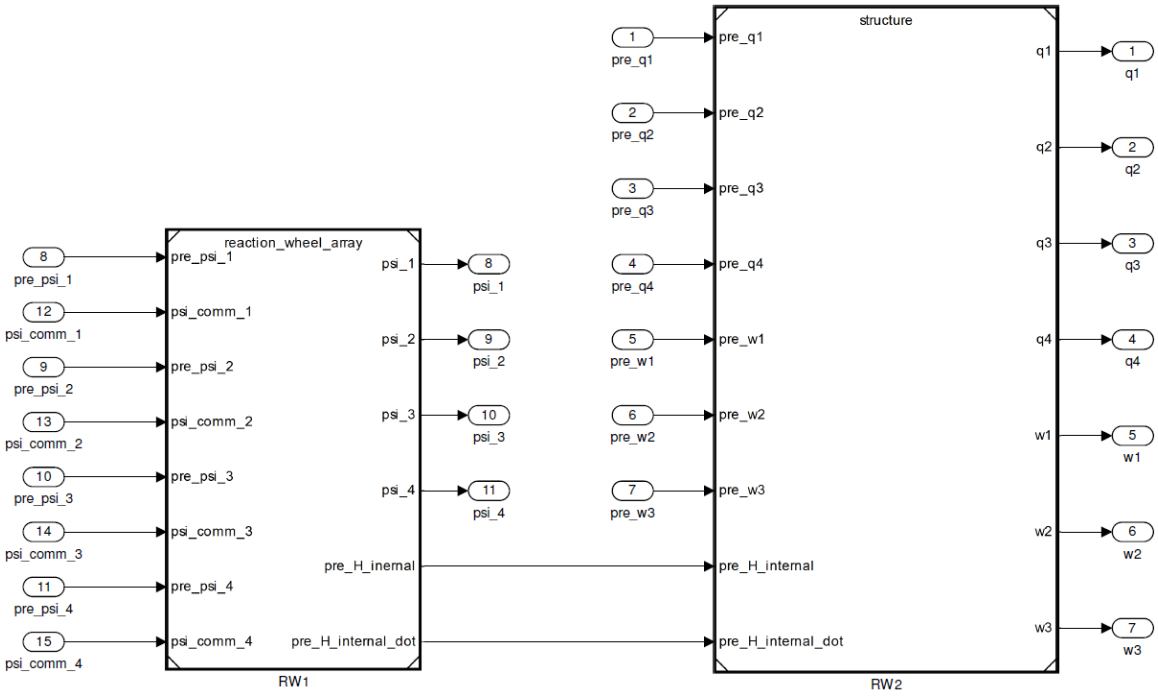

Figure 34. Environment Simulink Model

state and commanded wheel accelerations are input into the reaction wheel array component, which then updates and outputs the updated reaction wheel angular velocities, as well as the angular momentum and change in angular momentum of the RWA that acts on the spacecraft. The structure model receives the previous spacecraft orientation and angular velocity as well as the updated RWA angular momentum and change of angular momentum and updates and outputs the spacecraft orientation and angular velocity.

4.4.6 Reaction Wheel Array Model

As discussed in Chapter II, the function of the RWA is to impart a change of angular mometum on the spacecraft which must be met with an equal and opposite change in angular momentum from the spacecraft to conserve the total angular momentum of the system in the absence of external torques. The functionality of the RWA is broken down into instances of a reaction wheel component. The Simulink

model of the RWA is shown in Figure 35. The RWA contains model reference blocks

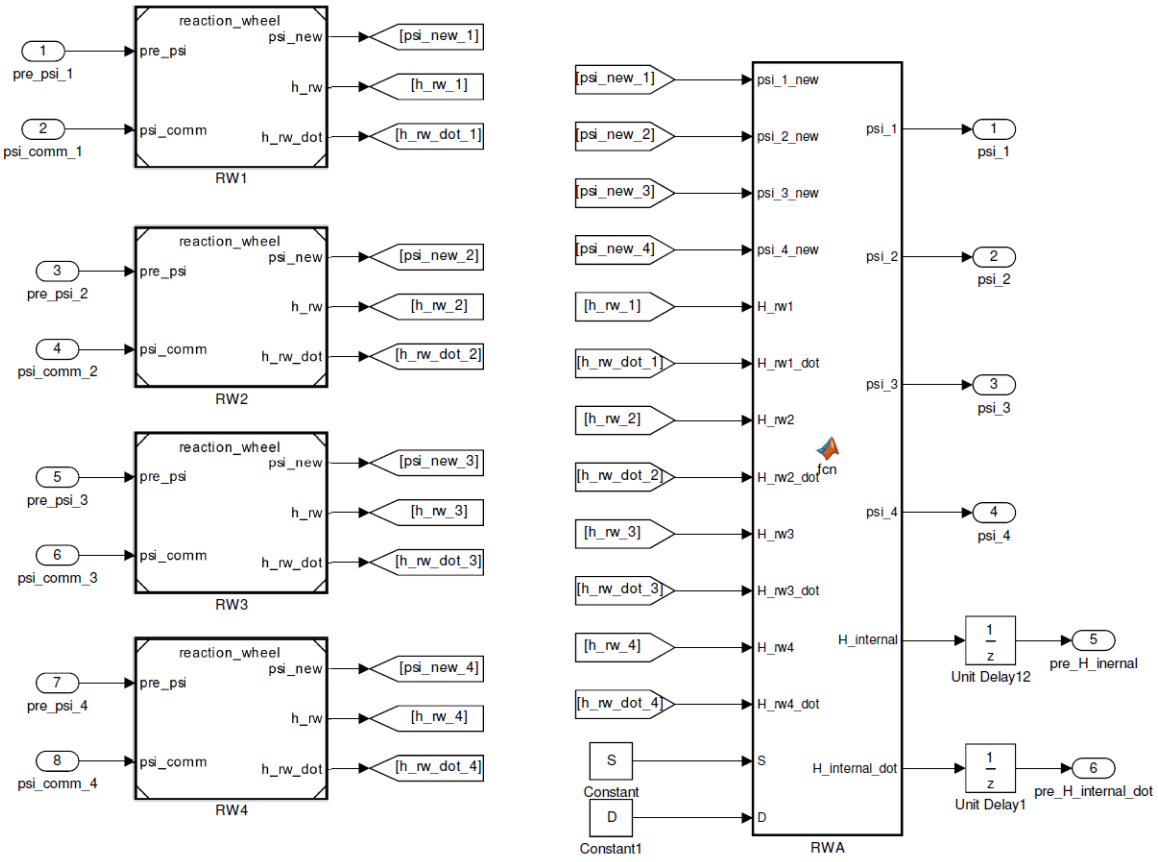

Figure 35. RWA Simulink Model

that import four instances of a single reaction wheel component model. Following composition practices from cyber-physical systems presented in Section 2.1.4, each reaction wheel instance has unique input and output names. The code for a reaction wheel component model is shown in Figure 36. Each reaction wheel instance receives its previous angular velocity and the commanded angular acceleration from the controller, and updates and outputs its angular velocity, angular momentum, and change in angular momentum about the access of spin of the reaction wheel. The change in reaction wheel angular momentum about the axis of spin is calculated as a product of the wheel's mass moment of inertia D and their angular acceleration. The updated angular velocity is a sum of the previous angular velocity with the product

```
function  [psi_new,H_rw,H_rw_dot]= fcn(pre_psi, psi_dot_comm, delta_t,D)
% set acceleration to commanded acceleration
psi_dot = psi_dot_comm;
% calculate change in angular momentum contribution
H_rw_dot = D*psi_dot;
% calculate new angular velocity
psi_new = pre_psi + psi_dot*delta_t;
% calculate change in angular momentum contribution
H_rw = D*psi_new;
```

Figure 36. Reaction Wheel Model Code

of the angular acceleration and the constant timestep of the simulation. The updated reaction wheel angular momentum is a product of the reaction wheel's mass moment of inertia D with the updated angular velocity of the wheel.

```
function [psi_1,psi_2,psi_3,psi_4,H_internal,H_internal_dot]= fcn(psi_1_new,psi_2_new,↙
psi_3_new,psi_4_new,H_rw1,H_rw1_dot,H_rw2,H_rw2_dot,H_rw3,H_rw3_dot,H_rw4,H_rw4_dot,S,D)
% Reaction Wheel Array Simulation
% Author: Kerianne Gross, AFRL/RQQA, kerianne.gross@us.af.mil
% Last Updated: 7 May 2015

% The sole purpose of the reaction wheel array block is to orient the
% angular momentum with the body frame

% "Hall sensor" output of angular rates
psi_1= psi_1_new;
psi_2= psi_2_new;
psi_3= psi_3_new;
psi_4= psi_4_new;

% Angular Momentum
H_internal = (1/D)*S*[H_rw1;H_rw2;H_rw3;H_rw4];
% Change in Angular Momentum
H_internal_dot = (1/D)*S*[H_rw1_dot;H_rw2_dot;H_rw3_dot;H_rw4_dot];
```

Figure 37. RWA Model Code

The RWA is modeled with a user defined function block that receives the updated angular velocities, angular momentums, and changes in angular momentum from each of the reaction wheels. The code contained within the RWA block is shown in Figure 37. With its inputs and the constant $\boldsymbol{S}$ matrix that orients each reaction wheel with the spacecraft body frame and the mass moment of inertia D of each individual wheel, the RWA block outputs the angular velocity of each wheel as well as the total angular momentum and change of angular momentum of the RWA.

4.4.7 Spacecraft Structure Model

The code contained within the structure block is shown in Figure 38, and based on the state equations presented in Chapter III but calculates each state component individually.

```
function  [q1_new,q2_new,q3_new,q4_new,pre_w1_new,pre_w2_new,pre_w3_new]= fcn(pre_q1,
pre_q2,pre_q3,pre_q4,pre_w1,pre_w2,pre_w3,pre_H_internal,pre_H_internal_dot,S_pinv,D,Ixx,
Iyy,Izz,delta_t,m,ca1,ca2,ca3,ca4,sa1,sa2,sa3,sa4,cb,sb)

% ca1,ca2,ca3,ca4,sa1,sa2,sa3,sa4,cb,sb

pre_psi = S_pinv*pre_H_internal;
psi_1 = pre_psi(1);
psi_2 = pre_psi(2);
psi_3 = pre_psi(3);
psi_4 = pre_psi(4);
psi_dot = S_pinv*pre_H_internal_dot;
psi_dot_1 = psi_dot(1);
psi_dot_2 = psi_dot(2);
psi_dot_3 = psi_dot(3);
psi_dot_4 = psi_dot(4);
m1 = m(1);
m3 = m(2);
m2 = m(3);
q1_dot = (pre_q2*pre_w3)/2 - (pre_q3*pre_w2)/2 + (pre_q4*pre_w1)/2;
q2_dot = (pre_q3*pre_w1)/2 - (pre_q1*pre_w3)/2 + (pre_q4*pre_w2)/2;
q3_dot = (pre_q1*pre_w2)/2 - (pre_q2*pre_w1)/2 + (pre_q4*pre_w3)/2;
q4_dot = - (pre_q1*pre_w1)/2 - (pre_q2*pre_w2)/2 - (pre_q3*pre_w3)/2;
pre_w1_dot = m1/Ixx - psi_2*((D*pre_w2*cb)/Ixx - (D*pre_w3*sa2*sb)/Ixx) - psi_3*
((D*pre_w2*cb)/Ixx - (D*pre_w3*sa3*sb)/Ixx) - psi_4*((D*pre_w2*cb)/Ixx -
(D*pre_w3*sa4*sb)/Ixx) - psi_1*((D*pre_w2*cb)/Ixx - (D*pre_w3*sa1*sb)/Ixx) +
(Iyy*pre_w2*pre_w3)/Ixx - (Izz*pre_w2*pre_w3)/Ixx + (D*psi_dot_1*ca1*sb)/Ixx +
(D*psi_dot_2*ca2*sb)/Ixx + (D*psi_dot_3*ca3*sb)/Ixx + (D*psi_dot_4*ca4*sb)/Ixx;
pre_w2_dot = psi_1*((D*pre_w1*cb)/Iyy + (D*pre_w3*ca1*sb)/Iyy) + psi_2*((D*pre_w1*cb)/Iyy
+ (D*pre_w3*ca2*sb)/Iyy) + psi_3*((D*pre_w1*cb)/Iyy + (D*pre_w3*ca3*sb)/Iyy) + psi_4*
((D*pre_w1*cb)/Iyy + (D*pre_w3*ca4*sb)/Iyy) + m2/Iyy - (Ixx*pre_w1*pre_w3)/Iyy +
(Izz*pre_w1*pre_w3)/Iyy - (D*psi_dot_1*sa1*sb)/Iyy - (D*psi_dot_2*sa2*sb)/Iyy -
(D*psi_dot_3*sa3*sb)/Iyy - (D*psi_dot_4*sa4*sb)/Iyy;
pre_w3_dot = m3/Izz - psi_1*((D*pre_w2*ca1*sb)/Izz + (D*pre_w1*sa1*sb)/Izz) - psi_2*
((D*pre_w2*ca2*sb)/Izz + (D*pre_w1*sa2*sb)/Izz) - psi_3*((D*pre_w2*ca3*sb)/Izz +
(D*pre_w1*sa3*sb)/Izz) - psi_4*((D*pre_w2*ca4*sb)/Izz + (D*pre_w1*sa4*sb)/Izz) +
(Ixx*pre_w1*pre_w2)/Izz - (Iyy*pre_w1*pre_w2)/Izz - (D*psi_dot_1*cb)/Izz -
(D*psi_dot_2*cb)/Izz - (D*psi_dot_3*cb)/Izz - (D*psi_dot_4*cb)/Izz;

% Update individual states
q1_new = pre_q1+q1_dot*delta_t;
q2_new = pre_q2+q2_dot*delta_t;
q3_new = pre_q3+q3_dot*delta_t;
q4_new = pre_q4+q4_dot*delta_t;
pre_w1_new = pre_w1+pre_w1_dot*delta_t;
pre_w2_new = pre_w2+pre_w2_dot*delta_t;
pre_w3_new = pre_w3+pre_w3_dot*delta_t;
```

Figure 38. Spacecraft Structure Block Code

The spacecraft structure component model accepts the previous spacecraft orientation, previous spacecraft angular velocity, and internal angular momentum and change in angular momentum of the RWA and computes and outputs the updated spacecraft orientation and spacecraft angular velocity. The previous reaction wheel angular velocities and accelerations are backed out of the internal angular momentum and change of angular momentum contributed by the RWA.

4.4.8 RTA Model

When the RTA controller is used, the simulation model references the RTA controller rather than the PID controller but otherwise remains unchanged. The environment component of the model remains completely the same. The controller component references the RTA controller and the PID controller is referenced by the verified controller component of the RTA controller design. The model of the RTA Controller is shown in Figure 39. As seen in Figure 39, the RTA Controller model features three main components: an unverified controller, a verified controller, and a decision module. The unverified controller component could be any high performance controller than cannot be verified, and is intended to be an intelligent controller; however, for the purposes of demonstration, the unverified controller component in Figure 39 is a model reference block that references a ramp function, which will violate requirement R02 that limits the maximum reaction wheel angular velocities. The verified controller is a model reference block that calls the rate-limited PID controller. The unverified and verified controller components both receive the previous state of the system and output a commanded angular acceleration to each of the four reaction wheels. The competing outputs of both the unverified and verified controllers are inputs to the monitor and decision module component which along with the previous state of the system is used to determine which controller output will be sent to the

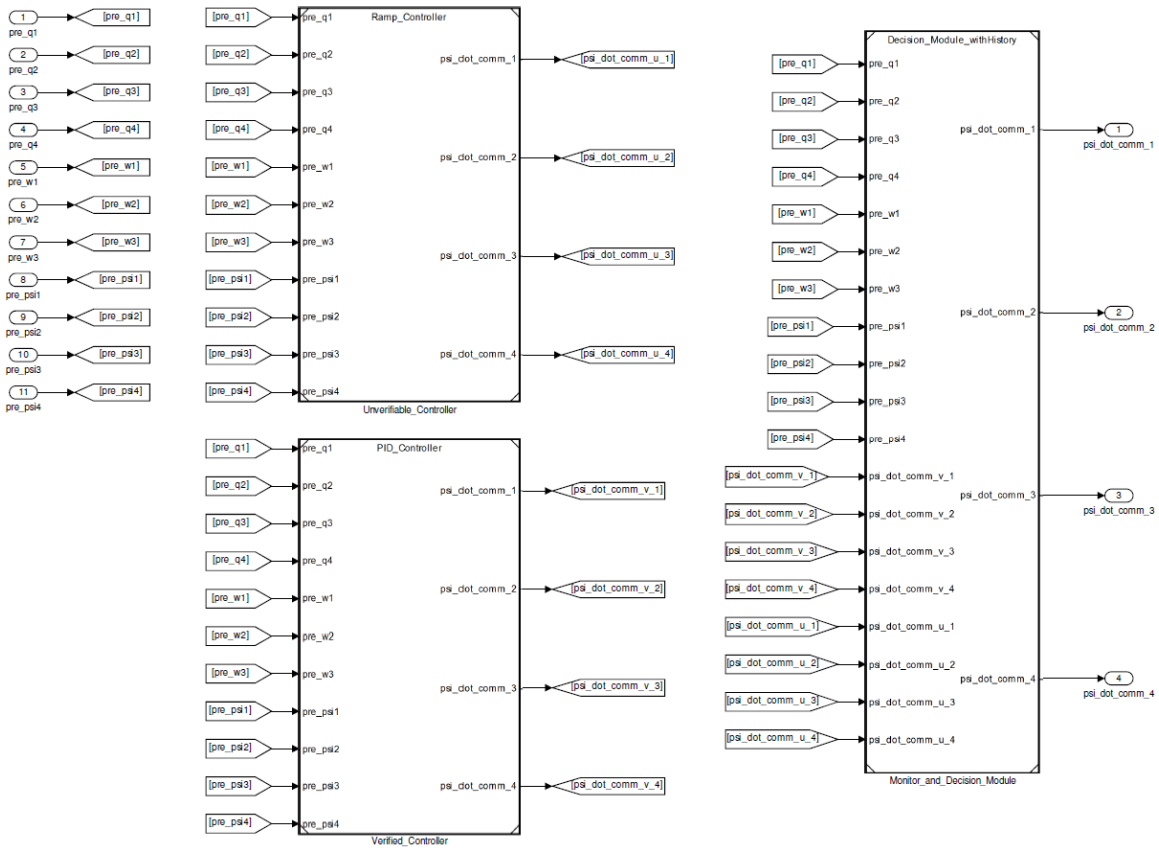

Figure 39. Run Time Assurance Controller Model

RWA. The decision module code is shown in Figure 40. As seen in Figure 40, a persistent local variable is introduced called "Violation." This local variable is used to provide a history of whether or not the unverified controller has previously violated the two safety properties of interest. If a previous violation has occurred, the output of the unverified controller is deemed to be untrustworthy, and the output of the verified controller will automatically be selected as the output from the decision module, and larger RTA controller subsystem. If a violation has not occurred previously, the decision module component checks whether the commanded angular velocity outputs of the unverified controller component will violate the maximum angular velocity and maximum angular acceleration of the reaction wheels.

```
function [psi_dot_comm_1, psi_dot_comm_2, psi_dot_comm_3,psi_dot_comm_4] = fcn(pre_q1,
pre_q2,pre_q3,pre_q4,pre_w1,pre_w2,pre_w3,pre_psi1,pre_psi2,pre_psi3,pre_psi4,D,M_tilde,
Kp,I,Kd,psi_max,psi_dot_max,delta_t,time,S_pinv,S,psi_dot_comm_v_1,psi_dot_comm_v_2,
psi_dot_comm_v_3,psi_dot_comm_v_4,psi_dot_comm_u_1,psi_dot_comm_u_2,psi_dot_comm_u_3,
psi_dot_comm_u_4)
% Decision Module for Reaction Wheel Control
% Author: Kerianne Gross, AFRL/RQQA, kerianne.gross@us.af.mil
% Last Updated:
% 20 November 2015
%
% States at the end fot he last time step
q       = [pre_q1,pre_q2,pre_q3,pre_q4];    %s/c quaternion
omega   = [pre_w1,pre_w2,pre_w3];           %s/c angular rates
psi     = [pre_psi1,pre_psi2,pre_psi3,pre_psi4];     %RWA angle rates

% Use persistent variable Violation to keep time history of the system
persistent Violation

if isempty(Violation)
    Violation = 0;
end

% CHECKS: 1) PREVIOUS VIOLATION
%         2) VIOLATION OF RW MAX ANGULAR ACCELERATION (PSI_DOT_MAX)
%         2) VIOLATION OF RW MAX ANGULAR VELOCITY (PSI_MAX)
if (Violation == 1 || ...
    abs(psi_dot_comm_u_1) > psi_dot_max || ...
    abs(psi_dot_comm_u_2) > psi_dot_max || ...
    abs(psi_dot_comm_u_3) > psi_dot_max || ...
    abs(psi_dot_comm_u_4) > psi_dot_max || ...
    abs(psi(1)+ psi_dot_comm_u_1*delta_t) > psi_max || ...
    abs(psi(2)+ psi_dot_comm_u_2*delta_t) > psi_max || ...
    abs(psi(3)+ psi_dot_comm_u_3*delta_t) > psi_max || ...
    abs(psi(4)+ psi_dot_comm_u_4*delta_t) > psi_max)
    Violation_flag = 1;
else
    Violation_flag = 0;
end
% CHECK: PREVIOUSLY VIOLATED
if (Violation_flag == 1)
    psi_dot_comm_1 = psi_dot_comm_v_1;
    psi_dot_comm_2 = psi_dot_comm_v_2;
    psi_dot_comm_3 = psi_dot_comm_v_3;
    psi_dot_comm_4 = psi_dot_comm_v_4;
else
    psi_dot_comm_1 = psi_dot_comm_u_1;
    psi_dot_comm_2 = psi_dot_comm_u_2;
    psi_dot_comm_3 = psi_dot_comm_u_3;
    psi_dot_comm_4 = psi_dot_comm_u_4;
end

% Keep record if Violation has occured
if Violation_flag == 1
    Violation = Violation_flag;
end
```

Figure 40. Decision Module Code

4.5 Feasability Analysis

As was discussed in Section 4.2.1, liveness requirements, such as a requirement that specifies a maximum slew rate for a maneuver, are satisfied in part by showing that it is mathematically feasible to achieve the desired requirement. Since the intent of a slew rate requirement is to provide guidance to proper sizing of an ACS actuator, FA may be sufficient to show satisfaction of the requirement. In order to show satisfaction of the slew rate requirement in this research, the maximum possible spacecraft angular velocity about each axis as a result of RWA actuation from initial spacecraft and reaction wheel velocities of zero was studied. The constants used to do the calculations in this section are summarized in Table 6.

Table 6. Constants Used in Feasability Analysis

Variable	Description	Value	Units
D	Reaction wheel mass moment of inertia in the axis of spin	4.12×10^{-5}	kg-m^2
ψ_{max}	Maximum reaction wheel angular velocity	576.0	rad/s
ψ_{max}	Maximum reaction wheel angular velocity	5500	rpm
I_{xx}	Spacecraft mass moment of inertia in the x-axis	0.021875	kg-m^2
I_{yy}	Spacecraft mass moment of inertia in the y-axis	0.0443657	kg-m^2
I_{zz}	Spacecraft mass moment of inertia in the z-axis	0.0559267	kg-m^2

First, the maximum angular velocity about the spacecraft z-axis $\vec{\omega}_{max_z}$ is achieved when all 4 wheels are spinning at their maximum velocity, since all four wheels contribute equally to the spacecraft z-axis, and may be calculated with

$$\vec{\omega}_{max_z} = -\boldsymbol{IS} \begin{bmatrix} \psi_{max} \\ \psi_{max} \\ \psi_{max} \\ \psi_{max} \end{bmatrix}. \tag{44}$$

The maximum velocity about the x-axis $\vec{\omega}_{max_x}$ is achieved by the use of only wheels 1 and 2 or 3 and 4, due to the orientation of the wheels in the spacecraft. If wheels 1 and 2 are accelerated to their maximum velocity from rest, a total spacecraft angular velocity may be calculated from

$$\vec{\omega}_{max_x} = -\boldsymbol{IS} \begin{bmatrix} \psi_{max} \\ \psi_{max} \\ 0 \\ 0 \end{bmatrix}. \tag{45}$$

The maximum velocity about the y-axis $\vec{\omega}_{max_y}$ is achieved by the use of only wheels 1 and 4 or 2 and 3. If wheels 1 and 4 are accelerated to their maximum velocity from rest, a total spacecraft angular velocity may be calculated from

$$\vec{\omega}_{max_y} = -\boldsymbol{IS} \begin{bmatrix} \psi_{max} \\ 0 \\ 0 \\ \psi_{max} \end{bmatrix}. \tag{46}$$

The results of this analysis are presented in Section 5.2.

4.6 Hypothesis Testing in Simulation

In this research, requirement satisfaction determination in simulation is based on hypothesis testing concepts. Hypothesis testing is a formal way of determining whether or not a system meets a specification or requirement. Traditionally, hypothesis testing is used in statistics to estimate a population parameter, such as a population mean, based on a population sample and a preconceived notion, or hypothesis, of the value of that parameter.[71] Each hypothesis tested consists of a null hypothesis H_0, which is traditionally the assumed status quo, and an alternative hypothesis H_1), which traditionally bears the burden of proof. One way to select the null hypothesis, is to assign it the requirement, specification, or property you are trying to prove, and to place all other cases in the alternative hypothesis. In the case of evaluating inequalities, the null hypothesis should always contain the equivalence term $(=, \leq, \geq)$.[71] The null and alternative hypothesis for each requirement are summarized in Table 7.

Table 7. Null and Alternative Hypotheses used to Evaluate the Requirements

Req't	H_0	H_1	Assumption
R01	$\lvert\dot{\psi}_{com}\rvert \leq \dot{\psi}_{max}$	$\lvert\dot{\psi}_{com}\rvert > \dot{\psi}_{max}$	H_0 is true
R02	$\psi_{pre} + \dot{\psi}_{com}\Delta t\rvert \leq \psi_{max}$	$\psi_{pre} + \dot{\psi}_{com}\Delta t\rvert > \psi_{max}$	H_0 is true
R03	$e_p \leq e_r$	$e_p > e_r$	H_0 is true
R07	eventually $\lvert\lvert\vec{\omega}_{max}\rvert\rvert \geq \omega_r$	never $\lvert\lvert\vec{\omega}_{max}\rvert\rvert \geq \omega_r$	H_1 is true
R08	$\lvert\lvert\vec{\omega}\rvert\rvert \leq \omega_{dr}$	$\lvert\lvert\vec{\omega}\rvert\rvert > \omega_{dr}$	H_0 is true
R09	$e_p \leq e_{tdr}$	$e_p > e_{tdr}$	H_0 is true
R10	$T_{s2\%} \leq T_{s2\%r}$	$T_{s2\%} > T_{s2\%r}$	H_1 is true
R11	$T_{s5\%} \leq T_{s5\%r}$	$T_{s5\%} > T_{s5\%r}$	H_1 is true
R12	$T_{\omega_{max}} \leq T_{rt}$	$T_{\omega_{max}} > T_{rt}$	H_1 is true
R13	$\%OS \leq \%OS_r$	$\%OS > \%OS_r$	H_1 is true

The requirements of the 6U CubeSat attitude control system are evaluated using hypothesis testing within each requirement's scope as stated previously. Requirements R01, R02, R03, R08, and R09 are requirements regarding variables that are

recalculated at every timestep and are considered violated if those variables exceed a specified value in any timestep. In the Cyber-Physical systems community, these requirements are considered invariants of the system, as described in Section 2.2.1. The approach used to hypothesis test these requirements is to assume that the null hypothesis is true and if at any point during the simulation the alternative hypothesis becomes true, then the requirement is falsified. For example it is assumed that the reaction wheel angular accelerations are always less than or equal to the maximum values, and if any one of the reaction wheels exceed the maximum acceleration at any point in the simulation, the requirement is falsified. This check is done inside the simulation loop, with a tracking variable that is permanently flagged as false for the current timestep and all future timesteps if the requirement is violated. This check is done with the following generalized psuedocode: `if RXX is true and RXX is within scope and variable_x > RXX_value, then RXX is false`.

For requirement R07, the slew rate is also recalculated at every timestep and the requirement is considered to be true if at any point in the simulation, the angular velocity of the spacecraft is greater than or equal to the slew rate requirement. In the cyber-physical systems community, requirement R07 is considered a liveness property, as described in Section 2.2.1. The approach used to hypothesis test this requirement is to assume that the null hypothesis is false, and if at any point during the simulation, the null hypothesis becomes true, the requirement is satisfied. This requirement check is also done inside the simulation loop, but with a variable that is permanently flagged as true for the current timestep and all future timesteps if the requirement is satisfied even once. This check is done with the following generalized pseudocode: `if RXX is false and variable_x` $\geq$ `RXX_value, then RXX is true`.

Requirements R10, R11, R12, and R13 all deal with a control systems response characteristic that is defined as a single value, and do not fall into a category of an

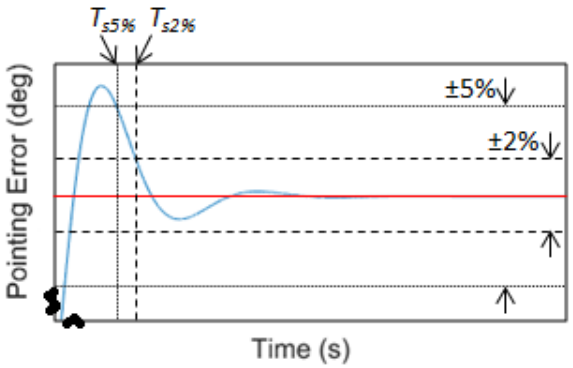

Figure 41. Generalized Example of 2 and 5 Percent Settling Time Determination

invariant or liveness property. The 2% and 5% settling time (R10 and R11) may be visualized in Figure 41, which represents the final few seconds of the maneuver. Figure 41 depicts the pointing error at the end of the maneuver with dotted horizontal lines representing a range of within 5% of the desired orientation, and dashed horizontal lines representing a range within 2% of the desired orientation. A dotted vertical line represents the 5% settling time, and the dashed vertical line represents the 2% settling time. When the pointing error enters and does not leave the 5% or 2% range for the remainder of the maneuver it is 5 or 2% settled. The 2% and 5% settling time (R10 and R11) of the simulation is set within the loop by identifying the first timestep when the orientation of the satellite is and remains less than or equal to 2% or 5% of the desired orientation. This assignment is done with two checks described by the following generalized pseudocode:

`if X_percent_settled is false and percent_of_final is` $\leq$
`X_percent, then RXX is true and time_X_percent_settled =`
`current_time. if RXX is true and percent_of_final is >`
`X_percent, then RXX is false.`

Anytime the system is within the desired range of the final orientation, these checks set the `X_percent_settled` flag to true and the `time_X_percent_settled` to the

current time of the simulation, but will continue to check if the system ever leaves the desired range, and if so, will set the `X_percent_settled` flag to false. Requirement satisfaction is determined outside the loop with the following generalized pseudocode:

`if X_percent_settled is true and time_X_percent_settled` $\leq$ `time_required, then RXX is true.`

The rise time and percent overshoot (R12 and R13) are also calculated in the loop with associated requirements checked inside the loop. The rise time T_r and percent overshoot $\%OS$ for an example slew maneuver are depicted in Figure 42. Figure 42 shows the overshoot of the pointing error $e_p(T_r)$ as a horizontal dotted

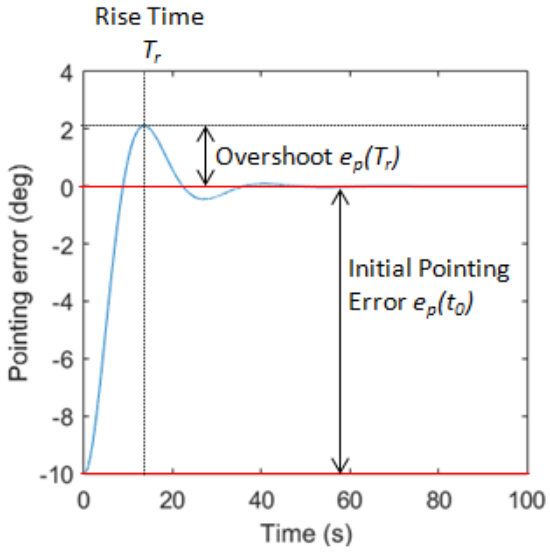

Figure 42. Example of Rise Time and Percent Overshoot Determination

line and the rise time T_r as a vertical dotted line. The percent overshoot is calculated as a ratio of the overshoot $e_p(T_r)$ and the initial pointing error $e_p(t_0)$. In simulation, the rise time and percent overshoot are identified by locating the first peak in the pointing error graph, and assigning the rise time as the simulation time where that peak occurs and percent overshoot as the pointing error as a percent of the original pointing error. Prior to the simulation, the rise time is initialized

as -1 seconds, and the requirement outside the loop is checked with the following pseudocode: `if rise_time > 0 and rise_time ≤ required_rise_time, then RXX is true`. The percent overshoot is perhaps the simplest check of them all with the following pseudocode: `if percent_overshoot ≤ required_percent_overshoot, then RXX is true`.

4.7 Simulation Analysis

In this section the set up for a SIM is described for a PID and RTA controller. For the SIM of the PID and RTA controllers, a slewing maneuver was simulated from an orientation of [0,0,0] to [90, 45, -45] degrees, as shown in Figure 43.

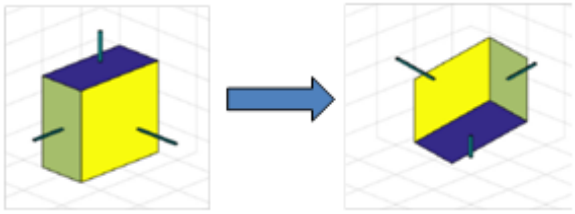

Figure 43. Initial and Final Orientation of the 6U CubeSat for the Simulated Slewing Maneuver

4.7.1 Simulation Assumptions and Limitations

Several assumptions that limit the fidelity of the analysis and simulation are as follows:

1. A simplifying assumption is made that the sensors provide perfect truth data and the sensors themselves are abstracted out of the simulation.

2. It is assumed that the reaction wheel speeds input into the controller will not exceed the maximum angular velocity, and that only the controller can cause this condition to occur. In other words, the angular velocity of the reaction

wheels is assumed to not exceed the maximum allowable before the controller can react.

3. It is assumed that the system does not experience any external torques such as gravity gradients or aerodynamic drag.

4. It is assumed that the system does not experience faults.

5. It is assumed that the CubeSat motion can be completely described using rigid body dynamics.

6. The simulation was conducted with a fixed time step. Initially the simulation was run at 10 Hz; however error propagation during the ramp phase of the simulation was exceptionally high at approximately 30%. In the simulation of the RTA Controller, the rate was increased to 1000 Hz, which reduced the calculation error to 0.3% over the 200 second simulation. In the SFEDS, each simulation was run at 500 Hz and the orientation quaternion was renormalized at every timestep.

Properties of the reaction wheel array and 6U CubeSat testbed were measured or estimated where possible; however some properties were assigned based on comparative performance for similar systems. The 6U CubeSat testbed 3×3 inertia matrix was taken from the AFIT SolidWorks model estimate. The individual reaction wheel and motor 3×3 inertia matrix was measured about the spin axis using the Model XR250 Measurement Instrument. The maximum reaction wheel speed used was the maximum speed observed in the laboratory as measured by a laser tachometer.

4.7.2 Simulation Constants

Several constants are used in the simulations. Simulation properties are summarized in Table 8, properties of the CubeSat are summarized in Table 9, properties of

the RWA are summarized in Table 10, initial and final conditions for the SIM are summarized in Table 11 and Table 12, and values used to create the gains of the PID controller are summarized in Table 13.

Table 8. Simulation Properties for PID Control Simulation

Variable	Description	Value	Units
Δt	Time step of the simulation	0.001	s
t_0	Initial time of the simulation	0	s
t_f	Final time of the simulation	100	s
n	Number of simulation timesteps	$\frac{t_f}{\Delta t}$	-

Table 9. CubeSat Properties

Variable	Description	Value	Units
I_{xx}	Spacecraft mass moment of inertia in the x-axis	0.021875	kg-m^2
I_{yy}	Spacecraft mass moment of inertia in the y-axis	0.0443657	kg-m^2
I_{zz}	Spacecraft mass moment of inertia in the z-axis	0.0559267	kg-m^2
$\boldsymbol{I}$	Spacecraft mass moment of inertia matrix	(diagonal matrix)	kg-m^2

Table 10. Reaction Wheel Array Properties

Variable	Description	Value	Units
D	Reaction wheel mass moment of inertia in the axis of spin	4.12E^{-5}	kg-m^2
ψ_{max}	Maximum reaction wheel angular velocity	576.0	rad/s
ψ_{max}	Maximum reaction wheel angular velocity	5500	rpm
τ_{max}	Maximum motor torque	0.0559267	kg-m^2
$\dot{\psi}_{max}$	Maximum reaction wheel angular acceleration	181.3	rad/s^2
α_1	Reaction wheel 1 angle from x-axis	315	deg
α_2	Reaction wheel 2 angle from x-axis	45	deg
α_3	Reaction wheel 3 angle from x-axis	135	deg
α_4	Reaction wheel 4 angle from x-axis	225	deg
β	Reaction wheels angle from z-axis	45	deg

Table 11. Initial Conditions

Variable	Description	Value	Units
Θ_{1_0}	Initial rotation about the spacecraft x-axis	0	deg
Θ_{2_0}	Initial rotation about the spacecraft y-axis	0	deg
Θ_{3_0}	Initial rotation about the spacecraft z-axis	0	deg
q_{1_0}	Initial quaternion element 1	0	-
q_{2_0}	Initial quaternion element 2	0	-
q_{3_0}	Initial quaternion element 3	0	-
q_{4_0}	Initial quaternion element 4	1	-
Φ_0	Initial principal Euler Angle	0	-
a_{1_0}	Initial Euler Axis element 1	0	-
a_{2_0}	Initial Euler Axis element 2	0	-
a_{3_0}	Initial Euler Axis element 3	1	-
$\dot{\psi}_0$	Initial reaction wheel angular acceleration (all wheels)	0	rad/s^2
ψ_0	Initial reaction wheel angular velocity (all wheels)	0	rad/s
ω_0	Initial spacecraft angular velocity	0	rad/s
m_0	Initial external torque (all axes)	0	N-m

Table 12. Final Conditions

Variable	Description	Value	Units
Θ_{1_f}	Final rotation about the 1st spacecraft axis	90	deg
Θ_{2_f}	Final rotation about the 2nd spacecraft axis	45	deg
Θ_{3_f}	Final rotation about the 3rd spacecraft axis	-45	deg
q_{1_f}	Final quaternion element 1	0.5	-
q_{2_f}	Final quaternion element 2	0.5	-
q_{3_f}	Final quaternion element 3	0	-
q_{4_f}	Final quaternion element 4	0.7071	-
Φ_0	Final principal Euler Angle	1.5708	rad
a_{1_0}	Final Euler Axis element 1	0.7071	-
a_{2_0}	Final Euler Axis element 2	0.7071	-
a_{3_0}	Final Euler Axis element 3	0	-

Table 13. PID Control Values

Variable	Description	Value	Units
ω_n	Natural frequency	1	Hz
ζ	Damping ratio	5	-

4.8 Space Filling Experimental Design Simulation Analysis

To complete the SFEDS of the PID controller, test points may be generated using a nearly orthogonal hypercube design to vary the initial and final orientation of the spacecraft for a slewing maneuver simulation. The initial and final pointing angle of the satellite are described by a set of 3 rotations about the x, y, and z-axis, which are then converted to quaternions to avoid encountering singularities during the simulation. The initial and final rotation about the x, y, and z-axes could be any value between 0 and 360 degrees. The x, y, and z components of the initial and final pointing angles are the six factors, or independent variables of the design. In order to test every possible combination of initial and final spacecraft orientations, 360^6 simulations would need to be completed, which at 11 seconds a simulation, would take over 69 million years of computing time. Fortunately much more efficient simulation techniques have been developed to simulate a representative sample of all possible test points. For this research, a NOLH design is used to feed inputs to the SFEDS. 257 design points were generated for the 6 factors using the spreadsheet developed by Professor Susan M. Sanchez,[79], which employs the algorithm described by Cioppas 2002 Doctoral Dissertation.[29] The 257 design points, one for each unique input and output in the NOLH design, are read by a MATLAB m-file that simulates 257 runs.

4.9 Reachability Analysis

As introduced in Section 4.2.1.3, requirements R04, R05, and R06 all deal with the concept of reachability. Each of these requirements state that the spacecraft should be capable of pointing in any direction from 0 to 360 degrees about a particular axis: x-axis for R04, y-axis for R05, and z-axis for R06. While reachability could be formally proven for a linear system by proving that the system is completely controllable using a controllability matrix, analysis of a nonlinear system's reachability is not trivial. In

this research a MATLAB batch simulation is used to provide some insight into the reachable space of the ACS, by simulating slews from 0 degrees to each whole number degree from 1 to 360 for each axis individually. This results in 360 test points per axis for a total of 1080 test points overall. If the pointing error at the end of the 100 second simulated test case is less than 0.001 degrees, then the final angle in the simulation is considered to be reachable.

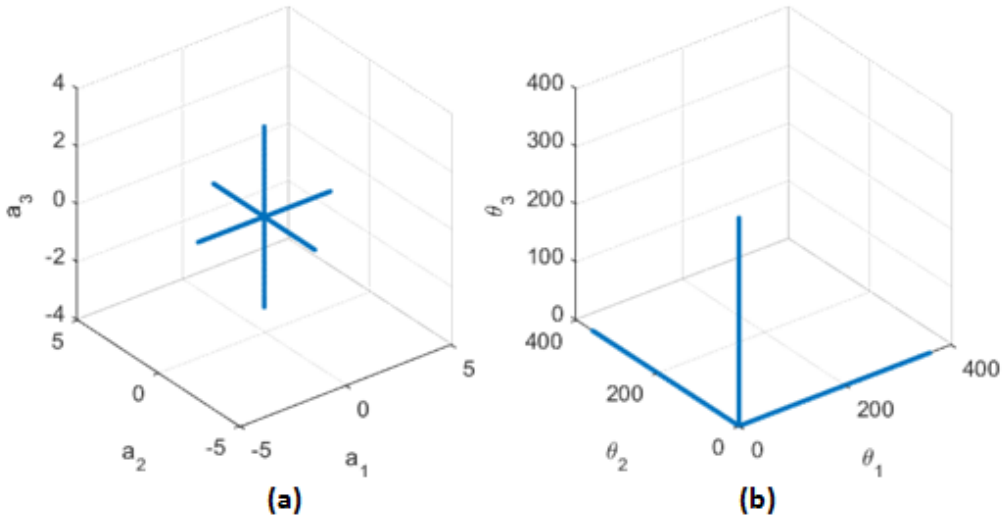

Figure 44. Reachability Simulation Test Points Representation

To visualize where these test points are located, two graphs were created as shown in Figure 44. Figure 44 (a) shows the test points as a product of their Euler axis with their Euler angle, so that they cover the x, y, and z-axes with values from -π to π. Figure 44 (b) shows the test points in degrees along the x, y, and z-axes, with points at every degree from 1 to 360. The results of this analysis are presented in Section 5.5. The use of alternative reachability analysis techniques such as a Flow* [27] simulation of reachable system states, or that presented by Lewis [65] or Bradley et al. [21] are left for future work.

4.10 Summary

Chapter IV described the requirements to be verified and the architecture and models developed for the research before describing how the FMA, SIM, SFEDS, and FA are conducted. The results of these analyses will be presented in Chapter V.

V. Results

In Chapter V, results from the analysis techniques described in the Chapter IV are presented. Results from FMA and SIM of the PID and RTA controller designs, FA of the slew rate requirement, SFEDS of the PID controller design, and batch simulation for the reachability of the ACS are presented as verification evidence that the requirements are satisfied by the controller implementation.

5.1 Formal Methods Analysis Results

Results of the FMA of the RTA controller are shown in this section. The PID controller is used as the verified controller component in the RTA controller design, so FMA is conducted on it as part of the compositional verification of the RTA controller. The RTA controller design was successfully implemented in SpeAR, AADL, and Simulink. Using SpeAR's reason tool, AGREE, and SLDV, the implementation of the controller in each of these tools was analyzed against the eight safety properties presented in Section 4.3.4. In SpeAR and AGREE, the unverified controller, verified controller, and decision module were all analyzed separately before they were combined compositionally into the RTA controller and analyzed at the controller subsystem level. All eight of the properties were proven valid for the verified controller, decision module, and larger RTA controller subsystem; however, the properties could not be proven for the unverified controller. In SLDV, analysis could only be completed on the verified controller and decision module, but could not be completed at the RTA controller subsystem level, due to limitations in SLDV analysis capabilities.

5.1.1 SpeAR Requirements Analysis Results

Using the methods described in Sections 2.4.1 and 4.3.2, the requirements of the system were analyzed to determine if they property constrain the design to prevent violation of system safety properties. The results of the SpeAR analysis presented in Figure 45 show that all eight safety properties presented in Section 4.3.4 are valid at the RTA controller level and for all of the RTA controller components except the unverified controller component. In Figure 45, the results of the RTA controller

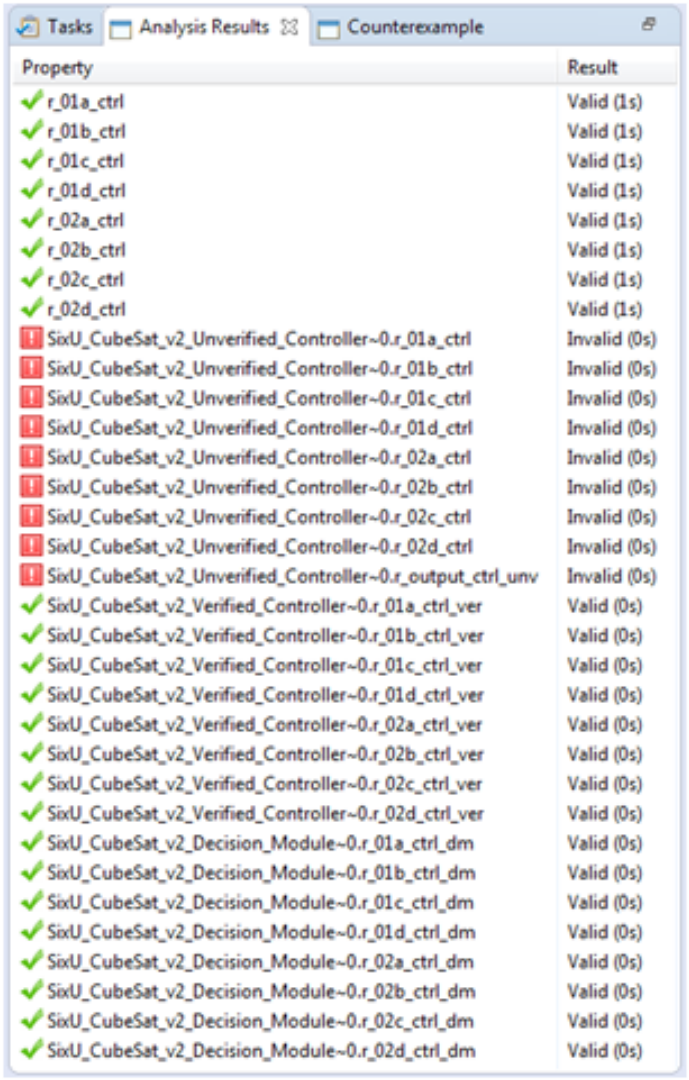

Figure 45. Results of SpeAR Analysis on RTA Controller Design

level analysis are listed at the top, the results of the unverified controller analysis are listed second, the results of the verified controller analysis are listed third, and the results of the decision module analysis are listed last. The green checkmarks and "Valid" result indicate where the safety properties are held, while the red exclamation point and "Invalid" indicate which safety properties are not held. These results are as expected. In other words, based on the assumptions on the input to each of the subsystems, and the derived requirements of the subsystems, it was proven that no case existed in which the output of the verified controller, decision module, or RTA controller as a whole would violate the safety properties of the system in abstracted requirements and architecture levels of the system description. SpeAR is being developed by members of the same organization responsible for developing AGREE and uses the same underlying solvers as AGREE, JKind and Z3. SpeAR and AGREE are both designed to enable compositional verification, a capability that allows SpeAR and AGREE to pull information from the subcomponent level analysis up to the subsystem level analysis. For this research, it means that SpeAR and AGREE can pull information from the verified controller, unverified controller, and decision module up to complete analysis of the RTA controller as a whole. If desired, it could pull information from the controller and environment levels up to complete a system level verification as well. In SpeAR, compositional verification of the RTA controller took less than a second to complete. This short analysis timeframe makes SpeAR analysis a valuable tool that can be run quickly to provide feedback during requirements development each time that an assumption, requirement, or derived requirement is added, deleted, or changed.

5.1.2 AGREE Architecture Analysis Results

Using the methods described in Sections 2.4.2 and 4.3.2, an architecture implementation of the ACS in AADL was analyzed using AGREE to determine if the implementation violates the safety properties presented in Section 4.3.4. The results of the AGREE analysis for the RTA controller are shown in Figure 46. While the SpeAR analysis took less than 1 second to complete, AGREE analysis took approximately 16 seconds, which also makes it a useful tool to quickly provide feedback to designers when architecture information is added, deleted or modified during architecture development. The results in Figure 46 are expanded at the RTA level to reveal the velocity and acceleration properties for each wheel. The decision module (DM_sub) and verified controller (VER_sub) properties are all also proven; however the properties are invalid for the unverified controller (UNV_sub), as indicated by the red box with containing the exclamation mark. In order to completely prove the RTA controller, one simply needs to remove the safety property guarantees from the unverified controller; however the safety properties are included in the SpeAR and AGREE analysis shown results here for demonstration purposes.

Property	Result
Verification for RTA_Controller.Impl	8 Invalid, 45 Valid
Contract Guarantees	8 Valid
r_01a_ctrl_rta: RW1 commanded angular acceleration shall not exceed maximum allowable angular acceleration	Valid (2s)
r_01b_ctrl_rta: RW2 commanded angular acceleration shall not exceed maximum allowable angular acceleration	Valid (2s)
r_01c_ctrl_rta: RW3 commanded angular acceleration shall not exceed maximum allowable angular acceleration	Valid (2s)
r_01d_ctrl_rta: RW4 commanded angular acceleration shall not exceed maximum allowable angular acceleration	Valid (2s)
r_02a_ctrl_rta: The predicted reaction wheel angular velocity shall not exceed maximum allowable angular velocity	Valid (2s)
r_02b_ctrl_rta: The predicted reaction wheel angular velocity shall not exceed maximum allowable angular velocity	Valid (2s)
r_02c_ctrl_rta: The predicted reaction wheel angular velocity shall not exceed maximum allowable angular velocity	Valid (2s)
r_02d_ctrl_rta: The predicted reaction wheel angular velocity shall not exceed maximum allowable angular velocity	Valid (2s)
Contract Assumptions	3 Valid
Contract Consistency	5 Valid
Verification for DM_sub	15 Valid
Verification for UNV_sub	8 Invalid, 3 Valid
Verification for VER_sub	11 Valid

Figure 46. Results of AGREE Analysis on RTA Controller Design

5.1.3 SLDV Model Analysis Results

Using the methods described in Sections 2.4.3 and 4.3.2, the Simulink model of the ACS could be analyzed using SLDV at the component level but the size of the model prevented RTA level analysis from being completed. Theoretically, analysis at the RTA level should be possible using model reference; however the Simulink model contains a much more detailed implementation of the system than the SpeAR and AADL model, resulting in a more computationally complex analysis task. For example, SLDV analysis of the decision module was able to prove all 8 of the safety properties as valid; however it took SLDV 23 seconds to complete the analysis for the decision module, while it took SpeAR less than 1 second and AGREE about 16 seconds to prove all the properties of the RTA controller, including the RTA controller's decision module and verified controller subcomponents. SLDV results of the rate-limited PID controller analysis were similar. When trying to prove RTA controller level analysis, the results were undecided due to nonlinearities. This is likely the result of a bug in SLDV which is expected to be fixed in future releases and may enable RTA controller level analysis in the near future. The result of the SLDV analysis for the decision module from the analysis report, which includes the 23 second analysis time, is shown in Figure 47. The result of the SLDV analysis for the PID controller in the SLDV log window, which displays during and after analysis, is shown in Figure 48.

Analysis Information

Model:	Verification_Decision_Module
Replacement Model:	C:\Users\Kerianne\Documents\MATLAB\sldv_output\Verification_Decision_Module\Verification_Decision_Module_replacement16.slx
Mode:	Property proving
Status:	Completed normally
Analysis Time:	23s

Objectives Status

Number of Objectives: 8
Objectives Proven Valid: 8

Figure 47. Results of Simulink Design Verifier Analysis on the Decision Module

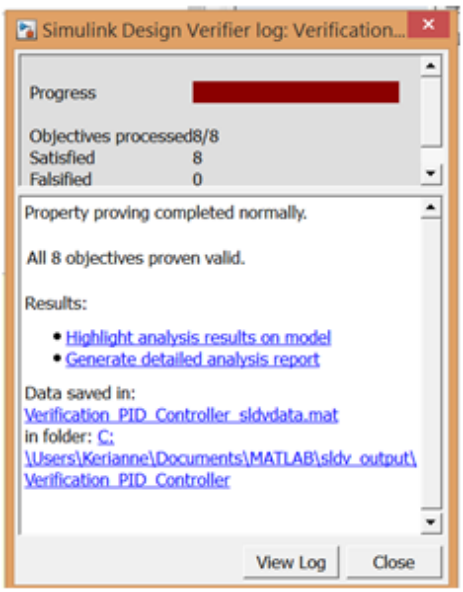

Figure 48. Results of Simulink Design Verifier Analysis on the PID Controller used as the Verified Controller

5.2 Feasibility Analysis Results

Using the methods prescribed in Section 4.5, analysis was conducted to determine if it is feasible for the RWA selected to produce the desired minimum slew rate specified by requirement R07 in Section 4.1.1. First, a maximum spacecraft angular velocity about the z-axis of the spacecraft $\vec{\omega}_{max_z}$ was found to be 68.7 degrees per second or 1.20 radians per second, with no angular velocity about the spacecraft x and y-axes, since the orientation of the wheels cancels out contributions to those axes. The maximum angular velocity about the spacecraft x-axis $\vec{\omega}_{max_x}$ was found to be 62.2 degrees per second or 1.08 radians per second, with 34.4 degrees per second or 0.60 radians per second generated about the z-axis due to the contribution of the wheels' angular momentum to the z-axis in addition to the x-axis. The maximum velocity about the spacecraft y-axis $\vec{\omega}_{max_y}$ was found to be 30.6 degrees per second or 0.53 radians per second, with 34.4 degrees per second or 0.60 radians per second generated about the z-axis. The results of the feasibility analysis are summarized

in Table 14. The maximum possible velocity about each spacecraft axis when the

Table 14. Maximum Possible Spacecraft Angular Velocity About Each Axis

Variable	Axis	Value	Units
$\vec{\omega}_{max_x}$	x	62.2	deg/s
$\vec{\omega}_{max_y}$	y	30.6	deg/s
$\vec{\omega}_{max_z}$	z	68.7	deg/s

spacecraft and reaction wheels start at rest and the reaction wheels are accelerated to their maximum velocity, as described in Section 4.5, far exceed the threshold and objective slew rates of 3 and 7 degrees per second, respectively. Based on these results, not only is it feasible for the spacecraft to reach the desired slew rate, the RWA is likely over designed, and the requirement may be met by a lower mass RWA design.

5.3 Simulation Analysis Results

In this section, results are presented for the SIM of the PID controller only. Then SIM results for the RTA controller are presented for a case where the unverified controller is a velocity ramp function, which is guaranteed to violate the safety properties for the purpose of simulating the transition from the unverified to the verified controller within the RTA controller architecture.

5.3.1 Simulation of PID Controller

Using the methods, assumptions, and constants from Sections 4.6, 4.7.1, and 4.7.2, a simulation was conducted to show the system requirements are met by the PID controller design. A summary of the results of the requirement analysis is shown in Table 15.

Table 15. Results of the Requirements Analysis on the PID Control Design for a Single Simulation Case

Req't	Description	Threshold	Objective
R01	$\lvert\dot{\psi}_{com}\rvert \leq \dot{\psi}_{max}$	satisfied	satisfied
R02	$\psi_{pre} + \dot{\psi}_{com}\Delta t\rvert \leq \psi_{max}$	satisfied	satisfied
R03	$e_p \leq e_r$	satisfied	satisfied
R07	eventually $\lvert\lvert\vec{\omega}_{max}\rvert\rvert \geq \omega_r$	satisfied	satisfied
R08	$\lvert\lvert\vec{\omega}\rvert\rvert \leq \omega_{dr}$	satisfied	satisfied
R09	$e_p \leq e_{tdr}$	satisfied	satisfied
R10	$T_{s2\%} \leq T_{s2\%r}$	satisfied	satisfied
R11	$T_{s5\%} \leq T_{s5\%r}$	satisfied	satisfied
R12	$T_{\omega_{max}} \leq T_{rt}$	satisfied	satisfied
R13	$\%OS \leq \%OS_r$	satisfied	satisfied

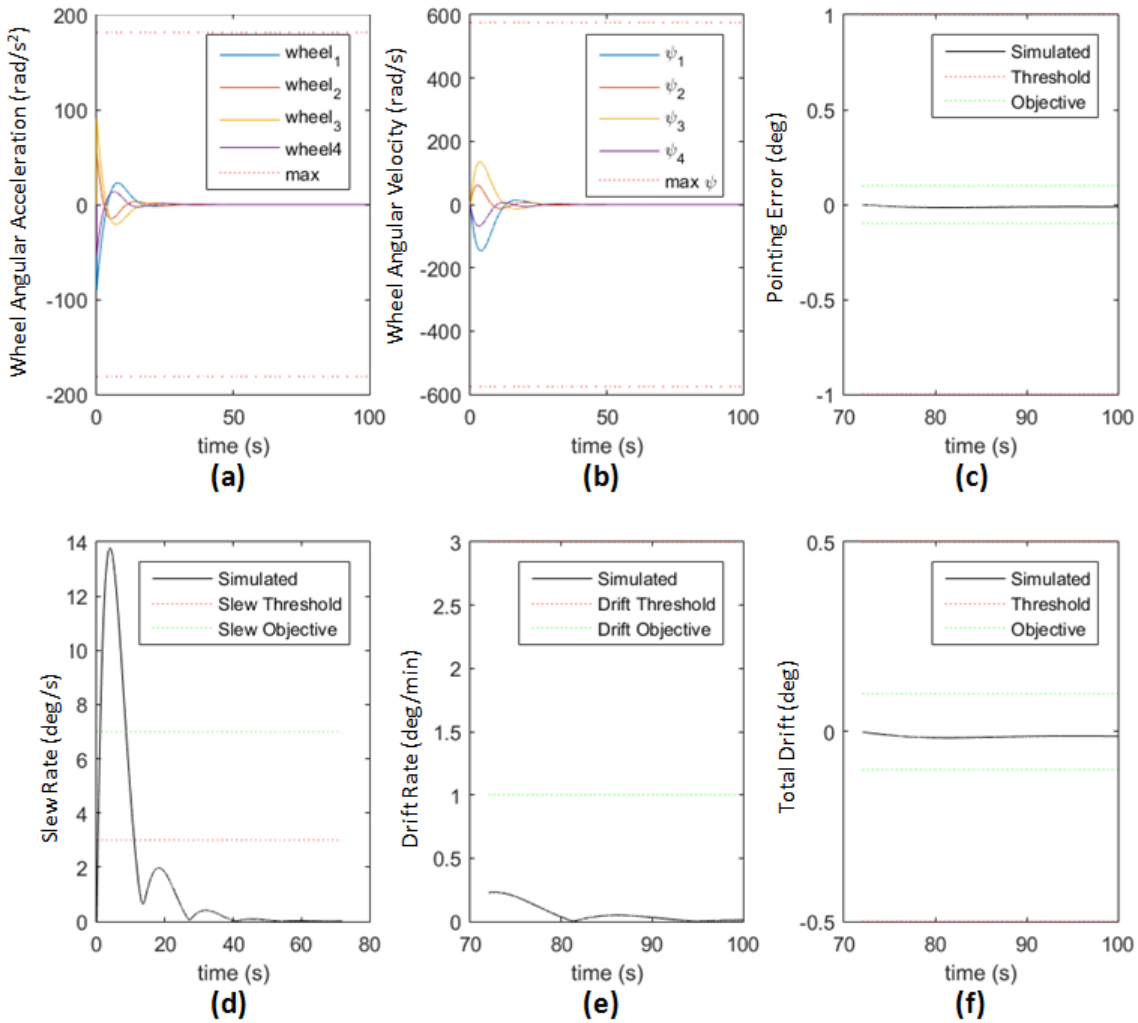

Figure 49. Performance Requirements for PID Controller Simulation

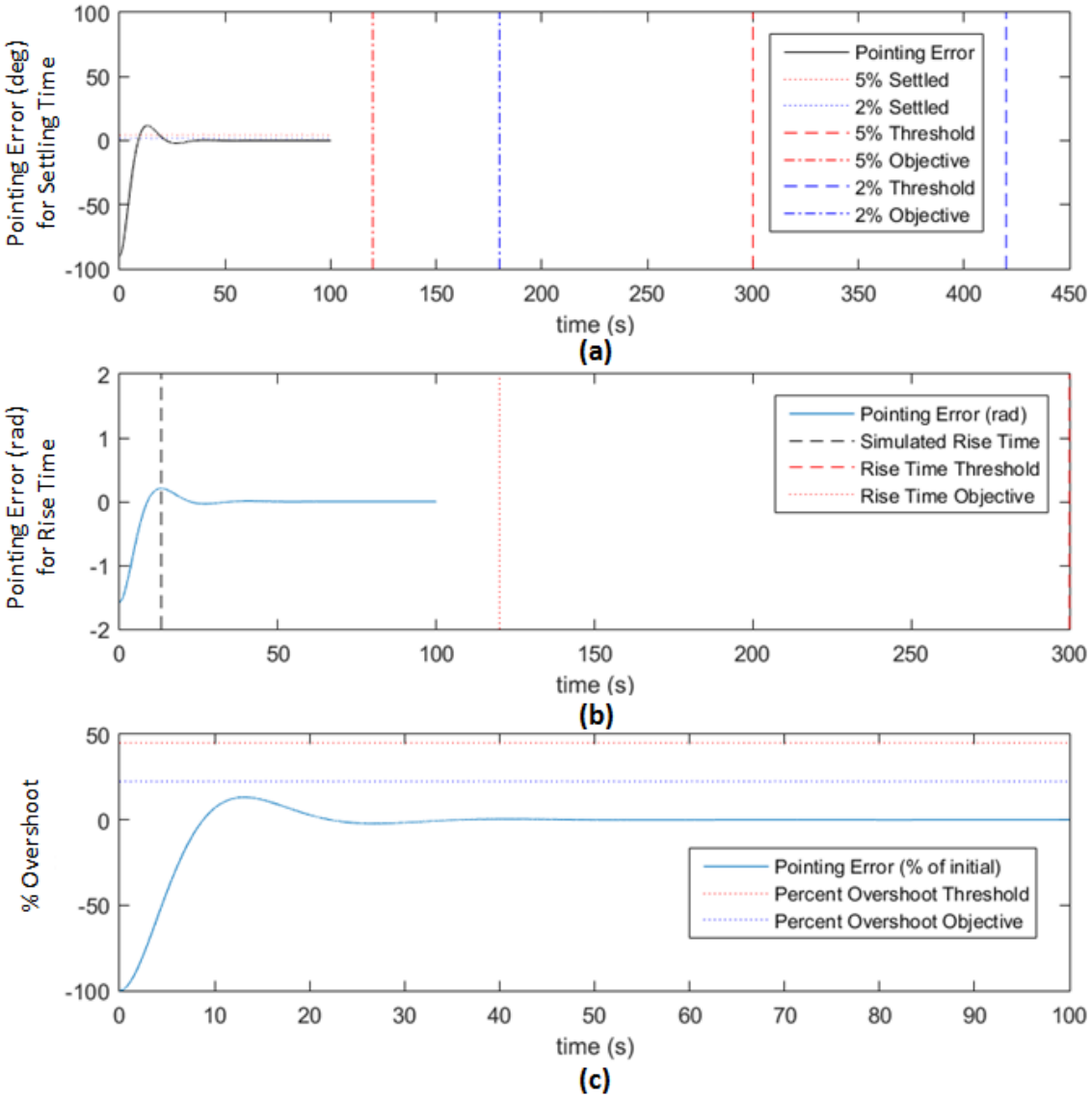

Figure 50. Control Systems Analysis Requirements for Simutaion with Reaction Wheel Array Actuation Only

To visualize the requirements, several plots were created as shown in Figure 49 and Figure 50. As seen in Figure 49 (a) and (b), the wheel angular acceleration and wheel angular velocity plots for all 4 of the wheels are well within the maximum limits, showing satisfaction of R01 and R02, respectively. The pointing error in Figure 49 (c), drift rate in Figure 49 (e), and total drift in Figure 49 (f) are all within the threshold

and objective requirements, showing satisfaction of requirements R03, R08, and R09 respectively. The slew rate in Figure 49 (d) exceeds the threshold and requirement, showing satisfaction of requirement R07.

As seen in Figure 50 (a), the 2 and 5 percent settled times are far less than the required settled time objectives. In Figure 50 (b) it may be seen that the rise time satisfies the rise time threshold and objective requirements, depicted as vertical lines on the plots. In fact, the rise time of 13.1 seconds is far less than the 120 second objective. As seen in Figure 50 (c) the percent overshoot requirement is also well below the objective and threshold for the requirement. A summary of the settling times, rise time, and percent overshoot is displayed in Table 16.

Table 16. Control Systems Analysis Requirements Summary

Req't	Variable	Value	units
R10	$T_{s2\%}$	18.8	s
R11	$T_{s5\%}$	20.6	s
R12	$T_{\omega_{max}}$	13.1	s
R13	$\%OS$	13.2	%

5.3.2 Simulation of Run Time Assured Controller

For the purposes of demonstrating the switching action of the RTA controller, a second simulation was conducted using the setup described in Sections 4.6, 4.7.1, and 4.7.2 for the model described in Section 4.4.8. A ramp function with a constant acceleration command to each of the reaction wheels was used in the place of the unverified controller because it was sure to violate system safety properties. The results of the slewing maneuever simulation are shown in Figure 51.

The spacecraft orientation is expressed as a quaternion and eventually completes

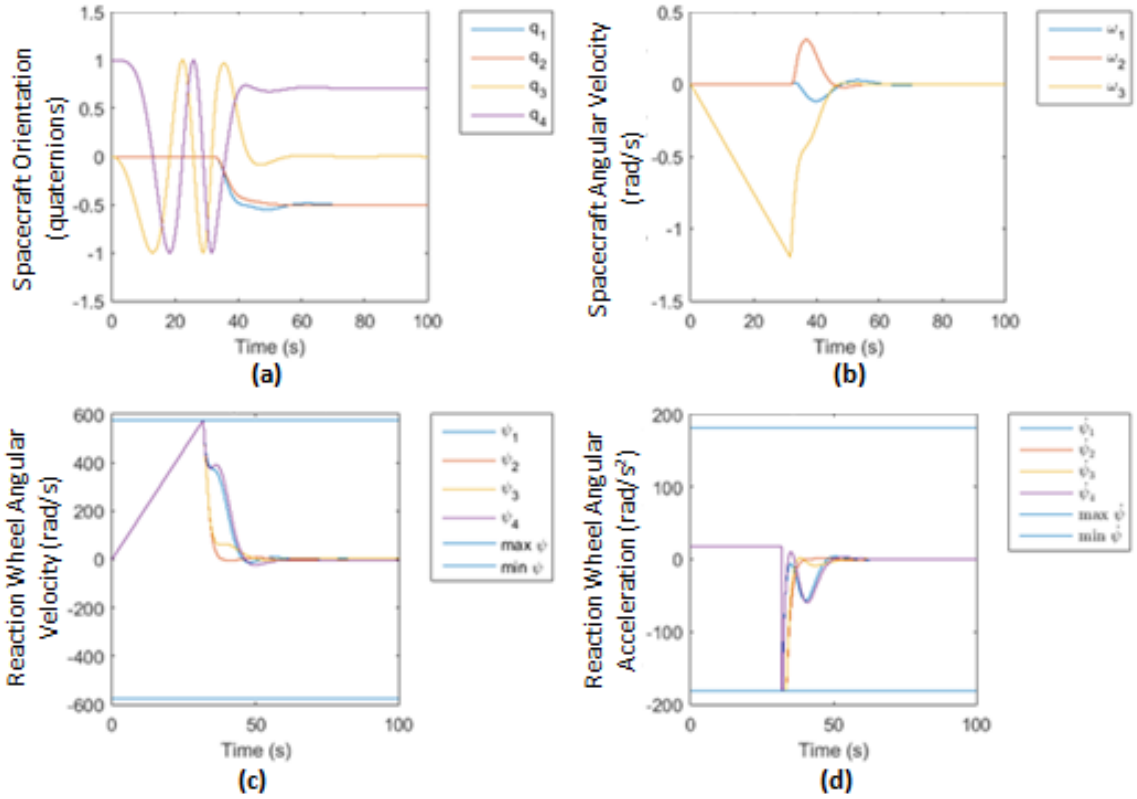

Figure 51. RTA Simulation with a Constant RWA Acceleration Ramp as Unverified Controller

the desired maneuver as seen in Figure 51 (a). As seen in Figure 51 (d), the unverified controller (ramp function) maintains control by issuing the constant acceleration command until 31 seconds into the simulation when the reaction wheel velocity shown in Figure 51 (c) reaches the maximum allowable. The spacecraft reaches a maximum angular velocity of about 1.2 radians per second about the z-axis under the control of the ramp function as seen in Figure 51 (b). When the reaction wheels reach their maximum velocity, the decision module switches command to the verified controller (rate-limited PID controller) for the remainder of the slewing maneuver. As seen in Figure 51 (d), the rate-limited PID controller reaches the maximum negative reaction wheel acceleration possible immediately after the switch in an attempt to dramatically decrease wheel speeds. After a few seconds, the acceleration commands leave

the maximum acceleration as the PID controller begins to settle out and eventually reach the desired spacecraft orientation.

5.4 Space Filling Experimental Design Simulation of the PID Controller

In this section, results are presented for the SFEDS of the PID controller. Using the SFEDS setup described in Section 4.8, 257 test points in a nearly orthogonal hypercube design were used to vary the initial and final orientation of the spacecraft for a slewing maneuver SFEDS. The total number of violations for each requirement when 257 test points were simulated with the rate-limited PID control design are summarized in Table 17.

Table 17. Number of Requirement Violations for PID Control With Rate Limiting

Req't	Description	Violations
R01	$\lvert\dot{\psi}_{com}\rvert \leq \dot{\psi}_{max}$	0
R02	$\lvert\psi_{pre} + \dot{\psi}_{com}\Delta t\rvert \leq \psi_{max}$	0
$R03_t$	$e_p \leq e_{r_t}$	0
$R03_o$	$e_p \leq e_{r_o}$	0
$R07_t$	eventually $\lvert\lvert\vec{\omega}_{max}\rvert\rvert \geq \omega_{r_t}$	2
$R07_o$	eventually $\lvert\lvert\vec{\omega}_{max}\rvert\rvert \geq \omega_{r_o}$	6
$R08_t$	$\lvert\lvert\vec{\omega}\rvert\rvert \leq \omega_{dr_t}$	0
$R08_o$	$\lvert\lvert\vec{\omega}\rvert\rvert \leq \omega_{dr_o}$	0
$R09_t$	$e_p \leq e_{tdr_t}$	0
$R09_o$	$e_p \leq e_{tdr_o}$	0
$R10_t$	$T_{s2\%} \leq T_{s2\%r_t}$	0
$R10_o$	$T_{s2\%} \leq T_{s2\%r_o}$	0
$R11_t$	$T_{s5\%} \leq T_{s5\%r_t}$	0
$R11_o$	$T_{s5\%} \leq T_{s5\%r_o}$	0
$R12_t$	$T_{\omega_{max}} \leq T_{rt_t}$	0
$R12_o$	$T_{\omega_{max}} \leq T_{rt_o}$	0
$R13_t$	$\%OS \leq \%OS_{r_t}$	257
$R13_o$	$\%OS \leq \%OS_{r_o}$	257

As seen in Table 17, only requirements R07 and R13 were violated. These violations will be explored in more detail in the next two subsections.

5.4.1 Slew Rate Requirement Violations

All violations of R07, which specifies a minimum value for the maximum slewing rate achieved in the maneuver, occur below 25 degrees of net angle, defined as the pointing error $e_p(t_0)$ between the initial and final orientation. These violations likely occur because the spacecraft angular velocity does not need to be very large for a slewing maneuver across a small angle versus a large angle. For small angles, achieving the desired slew rate during the maneuver would likely result in very large overshoots of the desired orientation. The pointing error and slew rate of the cases where R07 were violated are plotted in Figure 52.

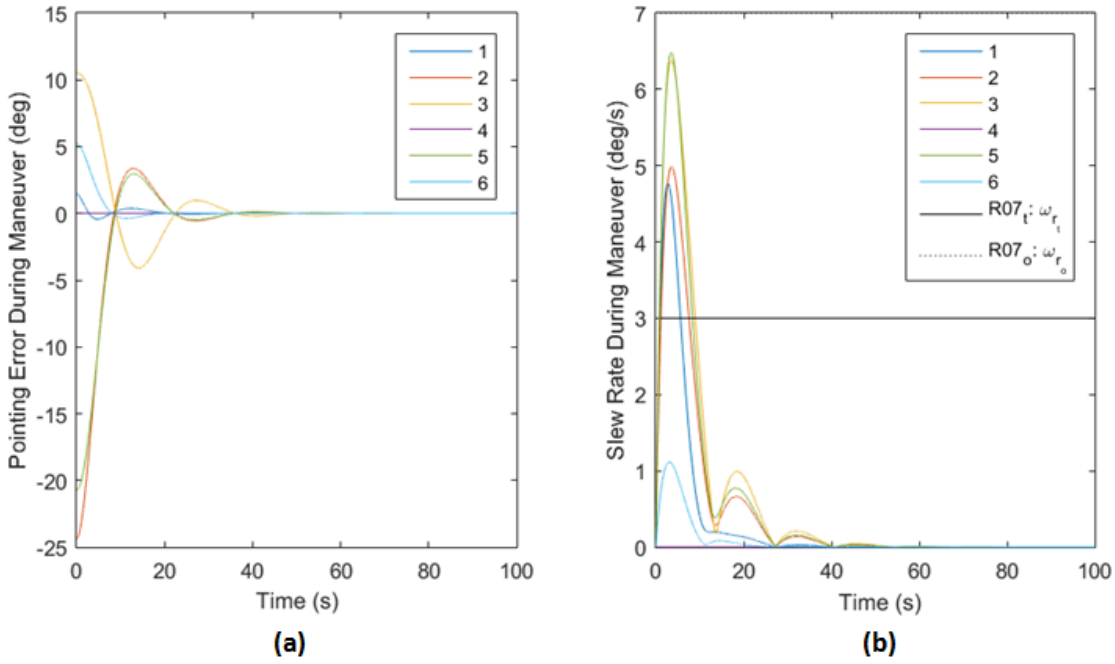

Figure 52. Pointing Error and Slew Rate for Test Cases Violating R07

From Figure 52 (a), it can be seen that the violating cases all reach their desired final orientation in approximately 60 seconds or less without excessive slew rates, which are shown in Figure 52 (b). The initial and final orientations are exactly the same for one of the test points, so no slew rate is required at all. The slew rate violations encountered in the SFEDS likely do not violate the intent of the

requirement, which is that the system responds quickly to orientation commands. For this reason, SFEDS is probably not the best way to determine whether a slew rate requirement has been violated; however, SFEDS analysis does give designers a good idea of how fast the spacecraft is slewing across a range of inputs. In this case, 255 of 257 requirements featured a maximum slew rate in excess of 3 degrees per second and 251 of 257 requirements featured a maximum slew rate in excess 7 degrees per second during the maneuver.

5.4.2 Percent Overshoot Requirement Violations

The percent overshoot requirement is a particularly problematic requirement because it is essentially a moving target that changes based on the maneuver commanded. Based on the original definition of percent overshoot that was calculated as a ratio of the first inflection point and the initial error $e_p(t_0)$, all cases violated the percent overshoot requirement. A plot of the percent overshoot versus net angle, or initial error is shown in Figure 53. Many violations likely occur because the

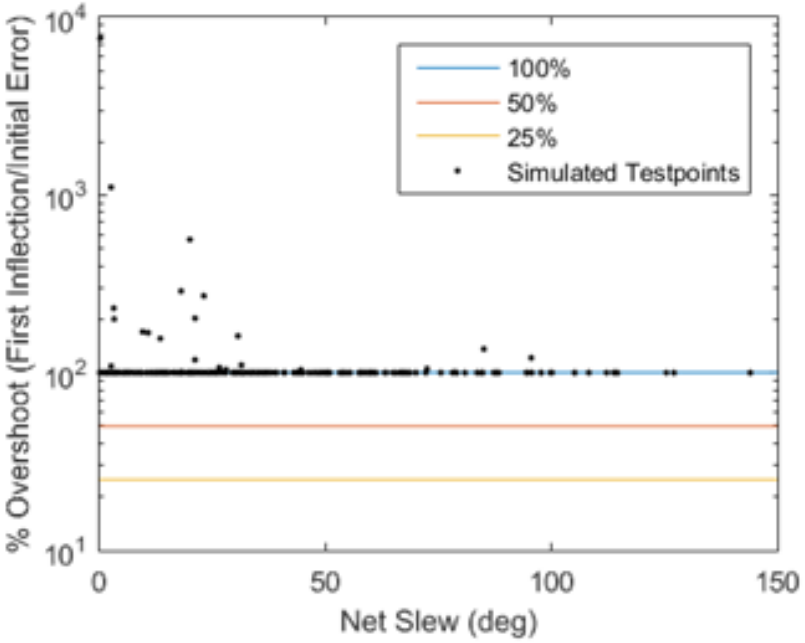

Figure 53. Percent Overshoot Versus Net Slew Angle

percent overshoot is calculated as a ratio of the net angle, defined as the pointing error $e_p(t_0)$ between the initial and final orientation. When the net angle is small, even a small overshoot angle is a much larger percentage of that net maneuver. This may be observed in Test Point 41, shown in Figure 54. In many cases an inflection

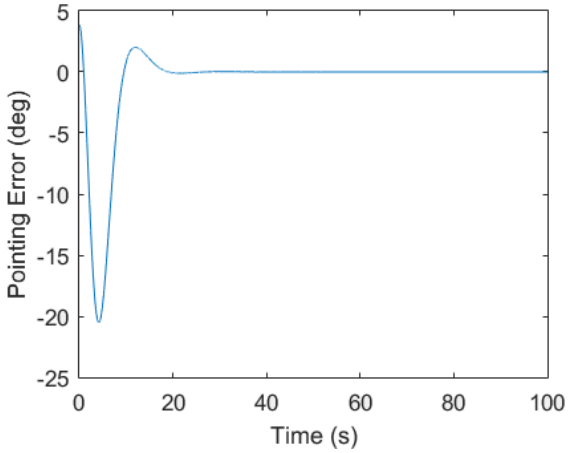

Figure 54. Pointing Error from Test Case 41 with a Large Percent Overshoot for a Small Angle Maneuver

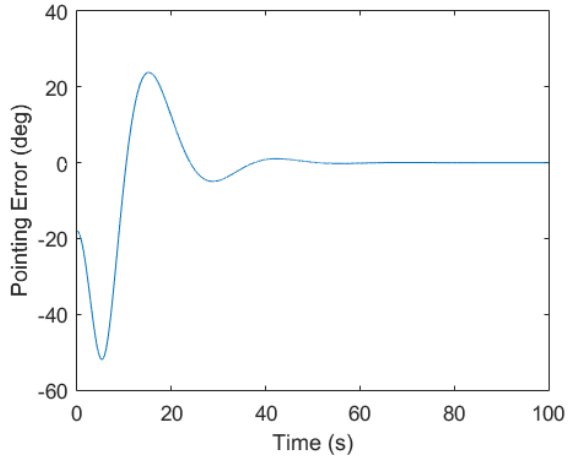

Figure 55. Pointing Error from Test Case 163 with Initial Divergence from the Desired Angle

point was identified in the simulation with in the first few time steps of the simulation. In some cases, this was an extremely small increase in the pointing error after the initial time step before the error decreased. In other cases, the percent error

increased significantly before approaching the desired orientation, as observed in test point 163 shown in Figure 55. Test cases such as 163 prompted a re-evaluation of the definition of the percent overshoot. Two new definitions were generated: Max Overshoot and Max Deviation. Max Overshoot is the maximum pointing error encountered after the pointing error has crossed the zero line. Max Deviation is the maximum pointing error encountered over the course of the simulated test point. These two characteristics are labeled in Figure 56. In the first alternative percent

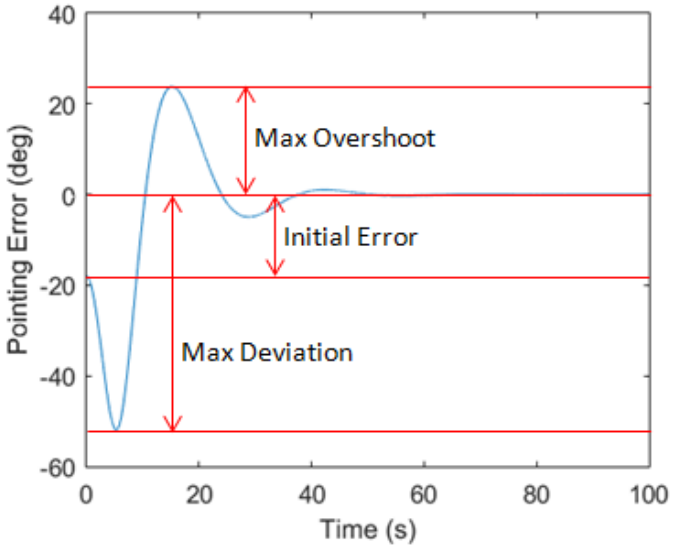

Figure 56. Max Deviation, Max Overshoot, and Initial Error $e_p(t_0)$ Definitions

overshoot definition, the percent overshoot is a ratio of the max overshoot and the initial error $e_p(t_0)$. This alternative percent overshoot is plotted against the net angle in Figure 57. In the second alternative percent overshoot definition, the percent overshoot is a ratio of the max overshoot and the max deviation. The second alternative percent overshoot is plotted against the net angle for each test point in Figure 58. The number of violations of requirement R13 based on the two alternative definitions of percent overshoot are summarized in Table 18. While percent overshoot may be used as a performance requirement for control systems, the sensitivity of requirement satisfaction to the magnitude of the maneuver makes this requirement particularly

vulnerable to violations. Instead of using a percent overshoot, a max deviation or raw max overshoot value may be a more appropriate requirement.

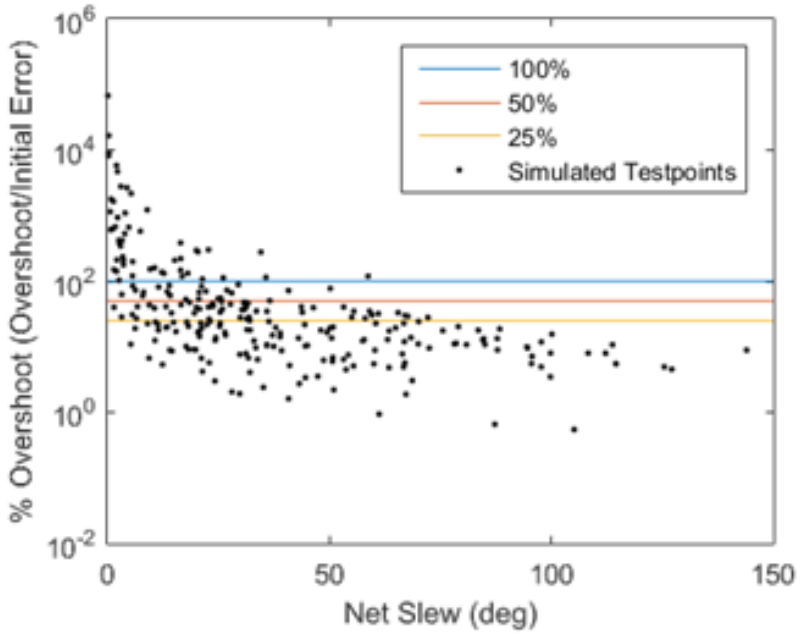

Figure 57. First Alternative Percent Overshoot versus Initial Error

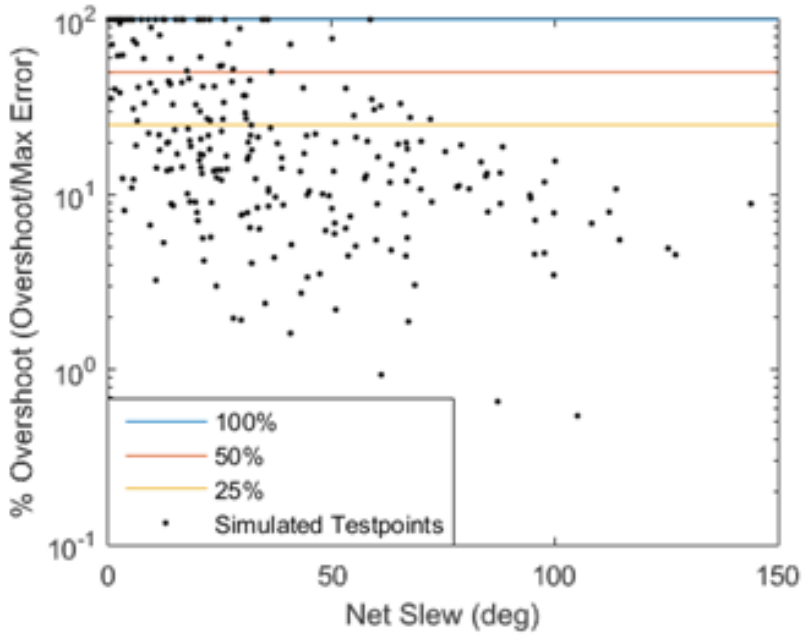

Figure 58. Second Alternative Percent Overshoot versus Initial Error

Table 18. Requirement Violations for Alternative Percent Overshoot Definitions

Requirement	Violations
$R13_t$ (alternative 1)	76
$R13_o$ (alternative 1)	126
$R13_t$ (alternative 2)	63
$R13_o$ (alternative 2)	102

5.4.3 Using JMP to Identify a Statistical Response Model of Percent Overshoot

JMP was used to identify if any models could be generated based on the raw data used to determine requirement violations. No statistical models of any meaning could be developed. In other words, all models had an R^2 value near 0, indicating that there was a very poor correlation between the model and the data. This lack of a statistical model for the response of the requirements may be a product of the nonlinear nature of the simulated controller and satellite dynamics. Variations of the requirements were analyzed as a result of this data. The only variation of data used to generate requirements that resulted in a relatively high R^2 value was a difference between the raw values of the max overshoot and max deviation as a function of the net angle, which was found to have a statistical model with an R^2 value of 0.712, which is relatively strong correlation when the maximum possible R^2 value is 1. The statistical model generated by JMP for this data is shown in Eq.(47).

$$\frac{180}{\pi}\Bigg(0.0346737090350608 + 0.920855002187844(e_p(t_0))+$$

$$\Big((e_p(t_0) - 0.627294612945526)(e_p(t_0) - 0.627294612945526)\Big)(-0.00899644953544666)\Bigg). \tag{47}$$

The difference in max deviation and max overshoot for each of the test points is plotted against the net angle as black dots, and the statistical model for the data is plotted as a blue line in Figure. 59.

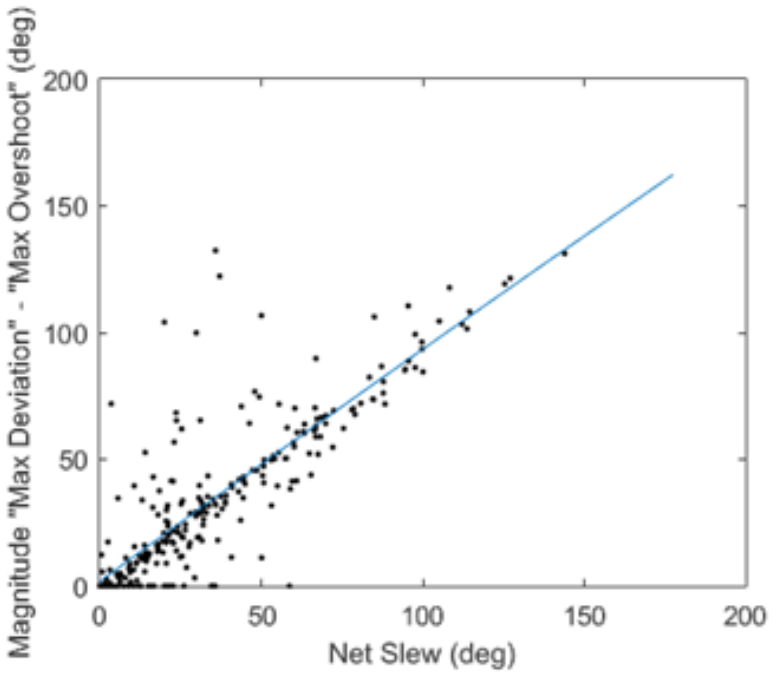

Figure 59. Max Deviation - Max Overshoot versus Initial Error and Model

5.4.4 Impact of Rate Limits on Requirement R01 and R02 Violations

A second SFEDS was conducted using the same 257 test points to determine the impact of the rate limits on the PID controller by observing how often R01 and R02

would be violated in the absence of rate limiting. The results of this simulation found that the PID controller without rate limiting violated the maximum acceleration requirement R01 in 63 cases, all within the first few seconds of the simulation. No violations of R02 occurred in the 257 simulated cases. When rate limits are placed on the PID controller, as derived requirements of R01 and R02, then R01 and R02 are not violated. The violations for each of the remaining requirements are the same as the violation results for the PID control with rate limiting. In other words, rate limiting only prevents violations of requirement R01.

5.4.5 Impact of Error Propagation on Requirement Violations

A major challenge encountered in the SFEDS analysis was balancing error propagation with simulation runtime for a discretized simulation. In order to run the discretized simulation at a reasonable rate for 257 test points, a timestep of 0.005 seconds was used. However, even with this small timestep, error propagation throughout the simulation led to a number of requirements violations that were not observed when a correction was added to the orientation calculation in the simulation code. Requirements R03 and R09 were particularly impacted as these requirements deal directly with the error between the observed and desired pointing angle of the spacecraft. Without any corrections, R03 and R09 objectives were violated in the same 64 test cases and the threshold values of R03 and R09 were violated in 5 and 12 test cases respectively. In addition, the objective and threshold of the rise time requirement, R12, was violated in 3 cases. The requirement violations due to the propagation of calculation errors in the simulation are summarized in Table 19. To remedy this error, the quaternion vector was renormalized at each time step. Because the magnitude of the quaternion should always be 1, each of the quaternion components was divided by the magnitude of the quaternion at each time step to correct for orientation er-

Table 19. Requirement Violations Due to Error Propagation

Req't	Description	Violations
$R03_t$	$e_p \leq e_{r_t}$	5
$R03_o$	$e_p \leq e_{r_o}$	64
$R09_t$	$e_p \leq e_{tdr_t}$	12
$R09_o$	$e_p \leq e_{tdr_o}$	63
$R12_t$	$T_{\omega_{max}} \leq T_{rt_t}$	3
$R12_o$	$T_{\omega_{max}} \leq T_{rt_o}$	3

rors. This does not prevent errors from building up in the spacecraft velocity over the course of the simulation, so it is possible a small error persisted in the final SFEDS.

5.4.6 Space Filling Experimental Design Simulation Summary

The results of the SFEDS showed that requirement violations were only encountered in requirement R07, which is met if a minimum value of the maximum slew rate is observed in the maneuver, and R13, which deals with percent overshoot. Explanations were provided for these violations and alternative percent overshoot definitions were explored. It was found that in many cases, a small inflection point in pointing error e_p occurred immediately after simulation initialization or the pointing error initially diverged from the desired orientation as seen in Figure 55. A statistical model could not be generated for any of the requirements violations when compared with the net slewing angle; however a statistical model could be generated to relate the different between the max deviation of the pointing error from the desired angle and the max overshoot. Rate limiting of the PID controller was found to prevent violation of requirement R01, which limits the maximum acceleration of the reaction wheels, in 63 cases. A mitigation technique for calculation errors was presented. It was noted that the SFEDS was not appropriate for determining satisfaction of a slew rate requirement and that a requirement on overshoot limitations may better be expressed as a max deviation or max overshoot value in degrees or radians.

5.5 Reachability Analysis Results

A description of the reachability simulation setup is provided in Section 4.9. As was described, an angle is considered to be reachable if the pointing error at the end of the 100 second simulated test case is less than 0.001 degrees. All 1080 test points were determined to be reachable. Plots of the pointing error over time for each axis may be seen in Figure 60.

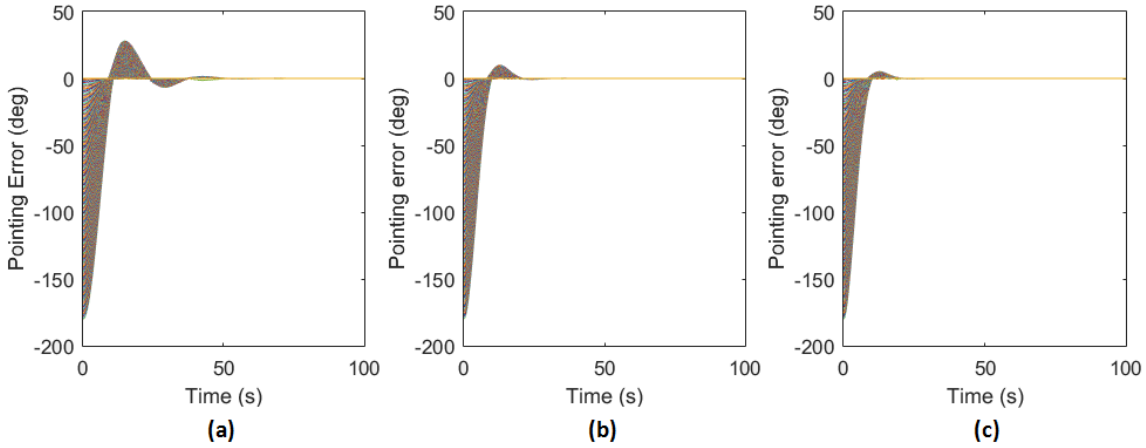

Figure 60. Reachability Pointing Error

Figure 60 (a) is a simulation of all the points on the x-axis, Figure 60 (b) is a simulation of all points on the y-axis, and Figure 60 (c) is a simulation of all points on the z-axis. Because the controller maneuvers across the shortest distance between two angles, the maximum error is 180 degrees. An unexpected insight gained from this set of test points is the difference in overshoot by axis. The x-axis simulated pointing error overshoots the desired angle by a maximum of 27.9 degrees, the y-axis simulated pointing error overshoots the desired angle by a maximum of approximately 9.9 degrees, and the z-axis simulated pointing error overshoots the desired angle by a maximum of 5.1 degrees. It makes sense that the z-axis would have the lowest overshoot because all four reaction wheels may have the same maximum contribution to the z-axis, giving the system the most control authority about the z-axis. The

difference between the x and y occurs because the spacecraft moment of inertia about the y-axis is more than twice the moment of inertia about the spacecraft x-axis.

5.6 Summary

In Chapter V, results from FMA, SIM, SFEDS, reachability SFEDS, and FA show satisfaction of all of the requirements of the PID controller with the exception of the percent overshoot requirement. It is recommended that a max overshoot value be used, rather than a percent overshoot requirement that is highly sensitive to the initial error. FMA and SIM results of the RTA controller were also presented showing that the two safety requirements that limit the maximum angular velocity and acceleration of the wheels are always satisfied by the RTA implementation.

VI. Conclusions and Recommendations

6.1 Research Summary

The scope of this research was to generate verification evidence that a controller implementation satisfied a set of requirements during the early design stages of a theoretical RTA controller design. The RTA controller's unverified controller component in this research was an abstracted "black box" controller that could represent any complex, learning, or intelligent controller design; however this controller could also be a "gray box" or "white box" controller that provides more knowledge of the unverified controller behavior. The RTA controller's verified controller was a rate-limited PID controller. A decision module was added that monitored the behavior of the unverified and switched to the verified controller to prevent safety property violations. The RTA controller's verified controller and decision module subcomponents and the composition of the RTA controller were analyzed with FMA to prove that safety requirements of the system are never violated. This result means that no matter what control design is inserted in the unverified controller block, as long as the inputs to the RTA controller are within a certain range, the output of the RTA controller is guaranteed to not violate safety properties. Properties of the spacecraft and RWA were analyzed using FA. The rate-limited nonlinear PID controller performance was analyzed with SIM and SFEDS to understand under what conditions performance requirements were satisfied or violated. Finally, a constant acceleration, velocity ramp function was inserted into the unverified controller and SIM analysis was performed to demonstrate the performance of the system when safety properties were threatened and the decision module switched control to the verified controller.

The approach presented in this research lays the groundwork for formally verified spacecraft attitude RTA control designs. Initial results on the application to a 6U

CubeSat ACS with RWA actuation indicate that the use of an RTA architecture is a viable way to assure that unverified control algorithms do not violate a predefined set of properties. However, much more work is required to build a controller design that is robust to faults and environmental conditions and interface that controller with the spacecraft hardware. In addition, the decision module presented in this work only monitored and switched on safety property violations, but does not consider performance properties.

This work produced the first application and evaluation of SpeAR, AGREE, and SLDV to provide traceable verification evidence for nonlinear system control in the requirements, architecture and modeling design phases. In order to provide verification evidence from the analysis methods used in this research, a new methodology was developed to formally describe different classes of controls requirements. In addition a new methodology was developed to combine the use of RTA, FA, FMA, SIM, and SFEDS to provide appropriate verification evidence for a spacecraft ACS design. Traceability of the requirements analysis in each design phase from requirements to architecture to models was accomplished with a consistent naming convention.

6.2 Conclusions

In this section, conclusions are presented about the selection of appropriate verification techniques, coverage of the design space by those techniques, the advantages and disadvantages of FMA, and the impact of simulation errors in this analysis.

6.2.1 Selection of Appropriate Verification Techniques

Appropriate combinations of verification techniques were identified based on the intent, scope, and complexity of each requirement as well as the coverage provided by the verification technique, which will be discussed in Section 6.2.2. Verification

of first two requirements in this research could be isolated to the controller software, and requirement satisfaction could be formally proven. This formal proof was possible because the rate limiting of the PID controller and the switching conditions of the RTA controllers decision module were implemented with linear equations. For these first two requirements, FMA was computationally inexpensive and provided complete coverage of the design space. However, additional insight could be gained on whether these requirements would be violated without rate limiting for the PID controller using SIM and SFEDS. Verification evidence for the remaining requirements, with the exception of those dealing with reachability, could also be generated using SIM and SFEDS, which allow visualization of requirement satisfaction, and identification of trends requirement violation conditions. A focused batch simulation for reachability analysis can provide some insight into reachability requirement satisfaction; however, coverage by such a technique is extremely limited. Finally, appropriate verification technique selection for liveness requirements, such as a achievement of a specific slew rate, is intent driven. Since the slew rate requirement intent is to provide guidance to proper sizing of control actuators, FA was conducted in this research to show that the actuator selected was capable of satisfying the requirement.

6.2.2 Verification Technique Design Space Coverage

The 6U ACS used in this research is designed to slew from one pointing orientation to another pointing orientation. Perhaps the most intuitive way to visualize the initial and final orientations is through an Euler Angle representation, presented in Section 3.2.1.2, where the orientation is represented as three successive rotations about a individual axes. The possible space of initial and final positions can then be visualized as a cube with axes labeled θ_1, θ_2, and θ_3, that each range from 0 to 360 degrees. The SIM analysis conducted in this research, which featured a single

simulation case, can then be visualized as Figure 61, where the initial position is represented by a $\circ$, the final position is represented by an $\times$, and a line is drawn showing the shortest distance between the two points.

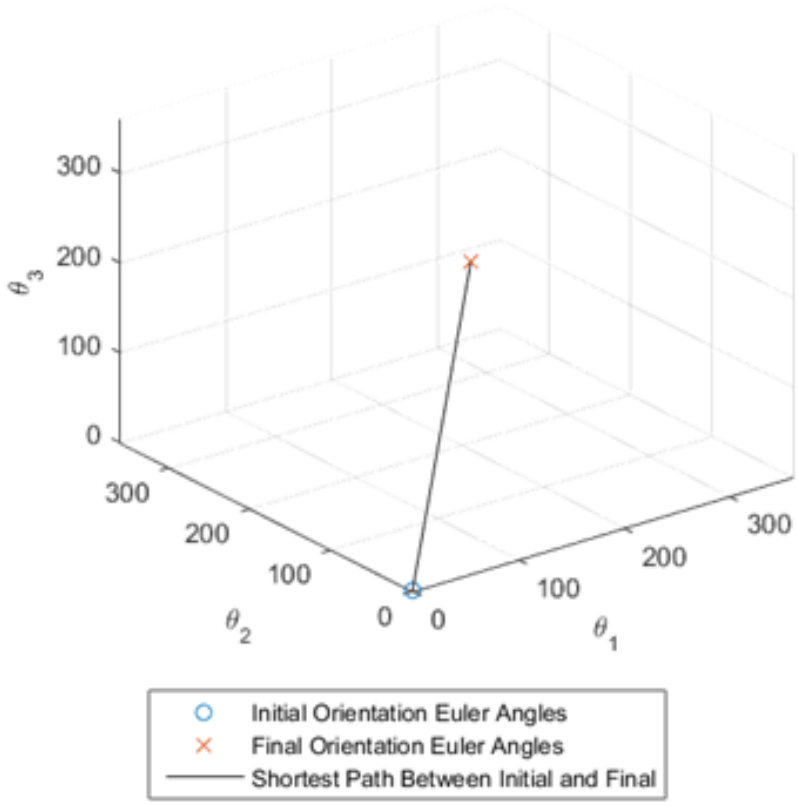

Figure 61. Experimental Space Coverage by the Single Simulation Case in this Research

Extending coverage from a single simulation case using a space filling experimental design, like that used in the SFEDS in this research, results in coverage represented by Figure 62.

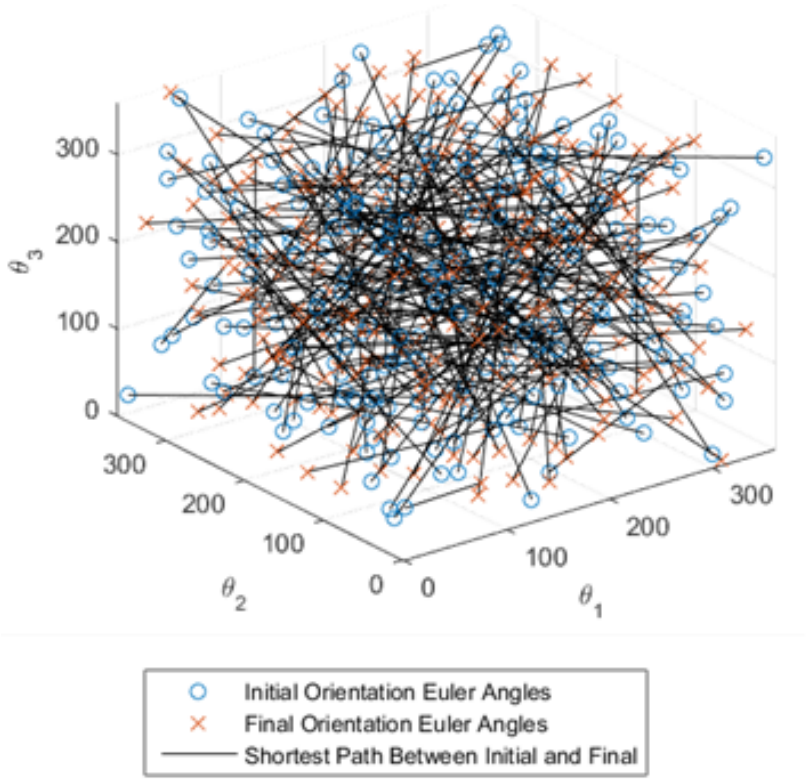

Figure 62. Experimental Space Coverage by the Space Filling Experimental Design Simulation in this Research

The version of the SFEDS used to do an initial reachability analysis along each axis resulted in design space coverage represented by Figure 63.

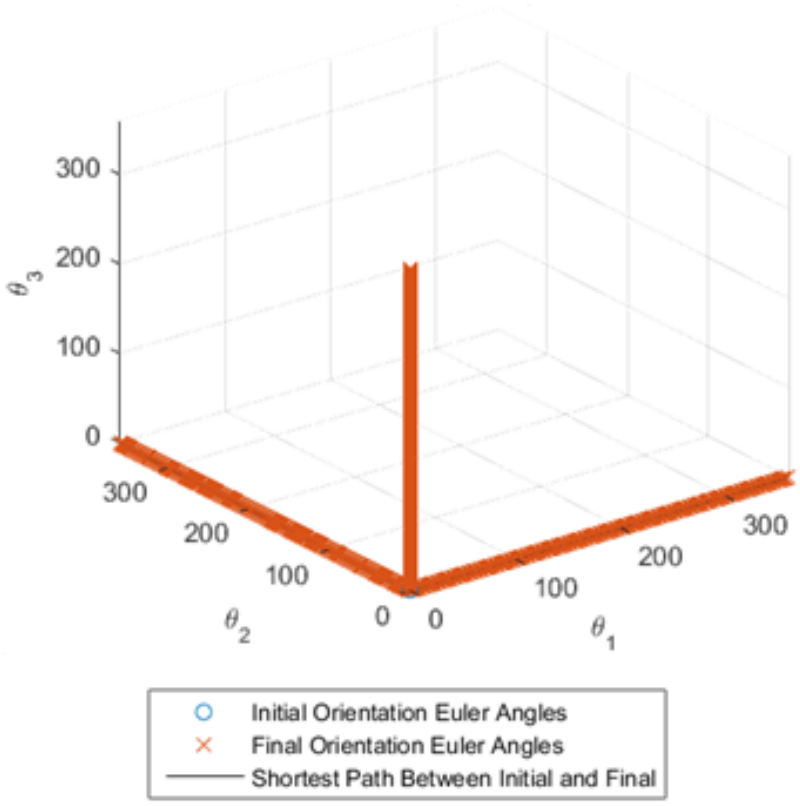

Figure 63. Experimental Space Coverage by the Reachability Simulation in this Research

The power of FMA becomes evident in the visualization of coverage of the entire design space as seen in Figure 64.

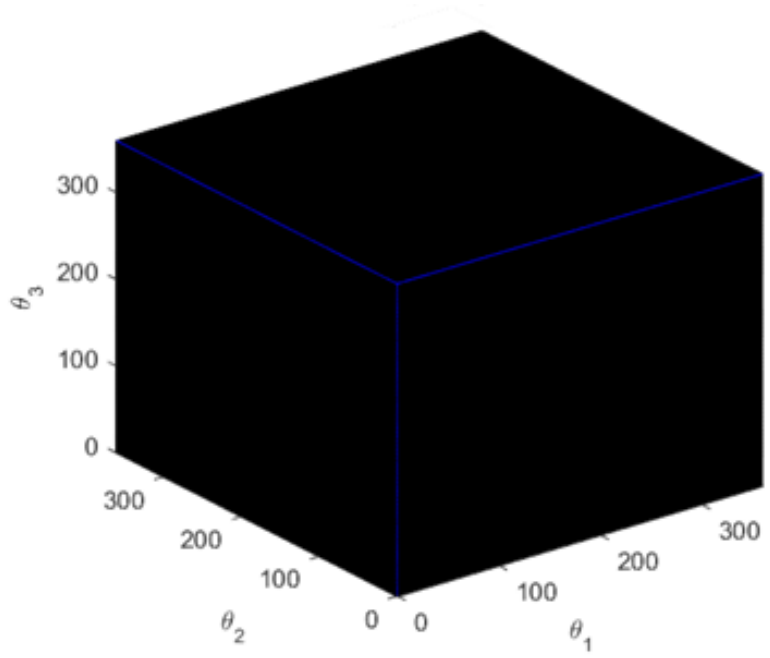

Figure 64. Experimental Space Coverage by the Formal Methods Analysis in this Research

While FMA provides complete coverage of the design space, current limitations in formal methods tools, only allowed it to be used to verify 2 of the 13 requirements. Conducting a SIM gives a designer initial insight into the performance of the system with relatively little computational burden and allows for visualization of requirements satisfaction for all 13 requirements. Conducting an SFEDS gives even more insight and confidence in the design and helps to identify problem areas in the experimental space where requirement violations are more likely to occur; however SFEDS can be computationally expensive. To conduct an initial reachability analysis to gain insight into requirements satisfaction, a simulation can be conducted in a targeted way such as that presented in this research; however even this leaves considerable open experimental design space.

6.2.3 Formal Methods Analysis Tools

The power of formal methods lies in its ability to prove that for a range of input conditions, the output of a component will never violate safety properties or requirements within some number of steps. However, proof is limited to invariant requirements that require something always holds, or bounded liveness requirements that require something occur within a fixed timeframe. Controls performance requirements such as settling time, rise time, and percent overshoot which do not have a temporal or conditional scope cannot be verified in the SpeAR, AGREE, or SLDV.

The most useful capabilities of SpeAR and AGREE are their compositional verification and their ability to do abstract analysis. SpeAR and AGREE can analyze property or requirement satisfaction at the lowest levels of the system and compose those results at increasingly higher levels up to the system level, which is an extremely powerful approach to dealing with the state-space explosion problem. In addition, analysis can be done on abstract variables. As long as the type of the variable is provided, an expression or value does not have to be assigned to it in order for it to be analyzed formally. This allowed analysis of the control systems outputs by only specifying the rate limiting strategy for the PID controller, without specifying the underlying PID control equations and equations of motion.

The power of using Simulink and SLDV is the ability to express and simulate nonlinear equations and use model reference to compose a simulation. However, values or expressions have to be assigned to each variable in SLDV and formal verification can only be completed at the component level because compositional verification is not currently supported. However, the rapid development cycle and improvements in SLDV make this a possible addition in a future release.

6.2.4 Simulation Errors

Like many cyber-physical systems in which the controller is discretized, the simulations in this researched were subject to calculation errors. When simulations were run at 10 Hz, some calculations were 30% off at the end of a 200 second simulation. At 1000 Hz, these errors were reduced to 0.3% over a 200 second simulation. In order to reduce errors in the SFEDS which was run at 500 Hz, the quaternion vector was renormalized at each time step. This does not prevent errors from building up in the spacecraft velocity over the course of the simulation, so it is possible a small error persisted in the final SFEDS.

6.3 Recommendations

In this section, recommendations are made for ACS requirement specification and future development of the formal methods tools used in this research.

6.3.1 Requirement Specification

All of the formalized requirements initially described in Section 4.2.1 worked well for analysis in this research with the exception of percent overshoot. Because percent overshoot is a ratio of the initial error, and a small inflection point often occurred right after the maneuver initialed or the orientation moved away from the desired position before it moved toward it in some cases, the definition provided in Section 4.2.1 did not work well for the analysis. Two alternative definitions were defined in Section 5.4. However, the biggest recommendation was that the overshoot requirement be a raw overshoot requirement such as 30 or 40 degrees rather than a percentage of the initial pointing error.

6.3.2 Formal Methods Toolset Development Recommendations

The formal methods tools evaluated in this research are becoming more capable and useable, but still have several gaps. The tools used to analyze the requirements, architecture and model of the 6U CubeSat reaction wheel array attitude control subsystem are sufficient to express system behavior and requirements that can be defined with simple math and inequalities; however, a large gap still remains in their nonlinear system analysis capabilities making them suitable for some applications, but not sufficient for complete analysis of the complex and nonlinear spacecraft attitude control example presented in this thesis.

Throughout this research, several gaps were identified in the SpeAR, AGREE, and SLDV tools that could greatly improve their usability and applicability to complex systems such as the 6U CubeSat ACS. While most recommendations are specific to individual tools, all of the tools could improve with nonlinear math support. The largest barrier for the 6U CubeSat requirements analysis is the lack of support for nonlinear math in SpeAR, AGREE, and SLDV, making exhaustive proof of the 6U CubeSat requirements infeasible without first linearizing the system.

SpeAR and AGREE could both be improved with support for matrix multiplication and a built-in notion of time. In the case of the 6U CubeSat, the PID controller and equations of motions for the satellite are most succinctly expressed and easiest to debug in matrix form. Without matrix support, these equations become very long, and it is more difficult to identify errors. As many of the requirements for this system and other control systems deal with system performance over time, adding a notion of time to SpeAR and AGREE would also be exceptionally useful. One workaround for this gap is to include time as an additional state that increments by the time step defined by the system rate. A built-in notion of time would simplify the expression of time-dependent properties.

SpeAR could also be improved with changes to the interface and unit expression capabilities. The current version of SpeAR allows the user to define the interface of the subsystem with named and typed inputs and outputs. However, while it currently allows inputs to be expressed separately, outputs are defined as a unique type that contains all the outputs of the system. Future versions of SpeAR will allow the user to express each output of a system separately. One of the promising features of SpeAR is its ability to perform unit checking on basic units. However, when more complex units were utilized in this example, SpeAR was not able to perform correct unit checking. This issue will also be addressed in the next version of SpeAR.

Though SLDV was the most mature of any of the formal methods tools, there were still several challenges that needed to be worked through during development. Initially analysis of any of the properties in SLDV produced errors, so an updated, 2015a version was used. This allowed some properties to be proven, some to be violated, and some that produced an "undecided due to nonlinearities" error. It didnt make sense for there to be such a variety of analysis results because all four reaction wheel commands are bounded using the exact same equations in the controller design. After this perplexing development, software developers at MathWorks were contacted who recommended only trying to prove one property at a time. By implementing this workaround, the four properties regarding reaction wheel acceleration where proven with no changes to the model, however the velocity properties were violated. This time though, counterexamples (an example of how a property can be violated) were presented for the wheel that showed the property could be violated if the angular velocity of the wheels exceeded the maximum allowable angular velocity before the control system had an opportunity to initiate control. To remedy this, an assumption was added that the previous value of the angular velocity was not greater than the maximum allowable angular velocity. This assumption constrains the formal analysis

tools to only search spaces in which the angular velocity doesnt start larger than the maximum before the control initiates. The inability to analyze all the properties at once was identified as a bug and the controller, properties, and verification models generated to do this analysis were sent to MathWorks. MathWorks identified what was causing the unintended behavior, provided workarounds in the model configuration, and will fix it in a future release.[62] Another major limitation of SLDV is its inability to conduct compositional verification of a system based on its components. For example, in this research, analysis of the RTA controller in SLDV yielded "undecided due to nonlinearies" errors. It is possible that this capability will be added in future releases.

6.4 Recommendations for Future Work

This section presents and discusses several gaps and opportunities for expansion of the research presented in this thesis.

6.4.1 Robustness and Resiliency Analysis

The models used in this research are deterministic; however, the initial system state and variations in the environment, aging of components, and other factors make the behavior of this controller design on an actual system non deterministic, and could be modeled in future work to study the robustness of the system and analysis techniques.

6.4.2 Hypothesis Testing Formal Methods Analysis Approach

Future research could apply the hypothesis testing approach presented in this research to the FMA approach. Analyzing both the null and alternative hypothesis would give extra confidence in the FMA verification evidence, by showing that

the FMA was not providing spurious results. Future research could investigate appropriate methodologies and evaluate the strengths and weaknesses of the combined hypothesis testing and FMA approach.

6.4.3 Hierarchical Requirement Development

All of the requirements presented in this research were presented in a flat, one level structure. However, research could be done to develop hierarchical requirements assigned to levels of the system or controller design where high level controller performance may be the highest level and derived requirements could be assigned at lower levels. Developing a methodology to hierarchically structure formal requirements and determining how that might impact how FMA is conducted could be an area of future study.

6.4.4 Formal Methods Analysis of Autocoded Software

This research could be expanded by linearizing the 6U CubeSat model, comparing the linear model performance to the nonlinear model, and conducting the formal methods analyses described in this paper. In addition, the abrupt saturation limits of the reaction wheels will need to be addressed in the formal methods analysis. Next, C code from the Simulink model could be analyzed using Mathworks Polyspace Bug Finder and Polyspace Code Prover.[6] The C code could be implemented on the AFIT 6U CubeSat Testbed attitude control subsystem and tests could be conducted to characterize the actual performance of the system. These test results could be combined with the simulation and formal methods analysis results to create an assurance case.

6.4.5 Alternative Reachability Analysis Techniques

In this research, some insight was gained into the system reachability through batch simulations; however, the use of alternative reachability analysis techniques such as a Flow* [27] simulation of reachable system states, or that presented by Lewis [65] or Bradley et al. [21] were left for future work. At the time of the publication of this thesis, the author is currently developing a hybrid system reachability simulation and anticipates publishing the results at a later date.

6.4.6 Additional Run Time Assurance Work

This work could be expanded in the future with the addition of mode logic, the use of two or more actuators, the use of three or more controllers within the RTA controller architecture, and the introduction of an intelligent controller. Mode logic could be added to deal with faults or situations outside of the systems current operating range. For example, the current design is limited by an assumption that the reaction wheels are not spinning above the maximum rate before the controller can react; however, this situation is possible if a fault occurs. Incorporating mode logic to react to this situation will allow the removal of this assumption.

Using two or more actuators will also make the control system more resilient. For example, the current control system design is intended to slew a satellite from rest in one position to rest in another position; however CubeSats can experience large angular velocities after being ejected from a CubeSat launcher and attempting to slow the spacecraft angular velocity with only a reaction wheel array actuator can cause the reaction wheels to saturate, a condition that occurs when a reaction wheel reaches a velocity limit that prevents it from exerting additional influence on the spacecraft attitude. Incorporating mode logic and an external torque actuator such as a magnetic torque coil, which interacts with the Earths magnetic field to control

the spacecraft attitude, could make the spacecraft more resilient. Information on magnetic torque coils and initial development of an architecture and simulation with a dual-actuated ACS is presented in Appendix II in Chapter VIII.

The RTA controller design presented in this paper only features two controllers: verified and unverified. Future iterations could include three or more controllers, where the decision module functions more like a negotiator that prioritizes control tasks based on the current spacecraft state as well as the output of the control tasks it is monitoring. A third controller within the RTA control architecture could be used to transition from one controller to another, or provide functionality to deal with a specific fault. Finally, the unverified controller block could play host to an artificially intelligent or learning controller that provides a more optimal control approach.

6.4.7 Floating Point Math Error Estimation

One of the gaps in the FMA process presented in this paper is quantifying the robustness of the analysis to calculation errors. Future work could also be conducted in making guarantees about the robustness of the system to errors in the floating point math used to estimate the equations of motion. Using a tool such as ASTREE [37], to quantify the worst case floating point error for each calculation, the designer could use the model checker to explore the tolerance of the guarantees to that error.

6.4.8 Reducing Abstraction

Effects such as integrator windup, timing delays, and sampling delays and their impact on the analysis results are abstracted out of this research but could be explored in future work. In this work, the sensors were abstracted and perfect truth data was used. Sensors and the impact of noise could be modeled and analyzed in future work to understand its impact on the verification evidence.

6.4.9 Simulation Error Reduction

In this research, the discrete simulations were conducted using a fixed-step Euler approximation solver for the change in system state at each timestep. Future work could investigate the processing time versus simulation error for different discrete fixed step solvers such as a second-order Heun method, a third-order Bogacki-Shampine method, or a fourth-order Runge-Kutta (RK4) method.[5]

6.5 Formal Methods Analysis Availability

The SpeAR, AADL/AGREE, and Simulink/SLDV files used to conduct the FMA in this research have been publically released with case number 88ABW-2015-6168 and are available on GitHub at `https://github.com/AFRL-VVCAS/6UCubeSat`. SpeAR and AADL/AGREE are open source products, and Simulink and SLDV are available through MathWorks.

VII. Appendix I: Hardware Characterization

To verify models used to simulate the ACS, several measurements were taken and behavior was compared to expectations. First, the mass moment of inertia (MOI) of the wheel and Maxon motor[72] (the rotating components of the reaction wheel array) was measured about the spin axis of the reaction wheel using the Model XR250 Measurement Instrument.[66] To reduce error, the average of four measurements was used, and the electrical cabling was tucked as seen Figure 65.

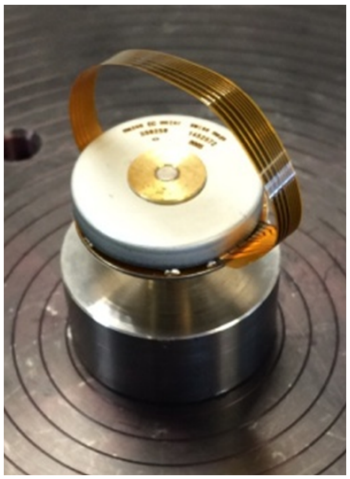

Figure 65. MOI Test Setup for Single Reaction Wheel and Motor

The measured MOI was an order of magnitude larger than the MOI predicted in a CAD model of the reaction wheel. Second, the MOI of the ADCS testbed was measured about the z-axis using the XR250. A tare was taken with a hollow plastic box on top (to be used to hold the spherical air bearing interface at the base of the testbed). Two reduce error, three measurements were taken, and the testbed was centered on the instrument and adjusted until it was measured level across the top with two perpendicular measurements. The test setup is shown in Figure 66.

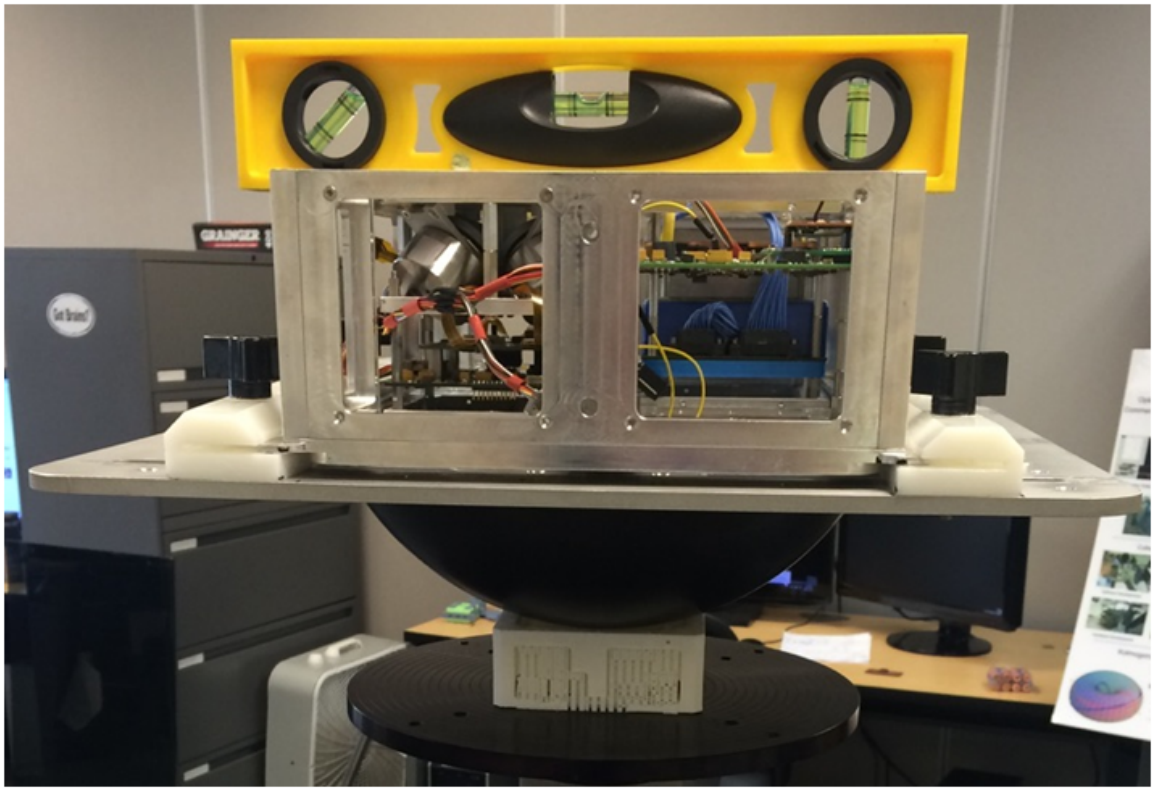

Figure 66. MOI Test Setup for Testbed Z-axis

The measured MOI of the ADCS testbed was 0.475 kg-m^2. Third, a modified version of the attitude control C code developed by Tibbs [92] was set to command a large acceleration until a large velocity was attained, and the steady state angular velocity of the reaction wheels was measured using a laser tachometer. The electronics board used to command the reaction wheels only facilitates three commands, so only three wheels were spinning. There was no variation in the measurement so one measurement was recorded for each wheel. The measured angular velocities of the reaction wheels are shown in Table 20.

Table 20. Measured Reaction Wheel Rotation Rates

Wheel	rad/s	rpm
1	549.5	5247
2	576.2	5500
3	534.1	5100

Fourth, the testbed was placed on the air bearing at rest in an off state and then turned on. Once the reaction wheels had reached their steady state speed, the period of rotation of the testbed about the z-axis was measured. The rotation rate and period of the testbed about the z-axis was compared to a predicted rotation rate and period calculated from the measured wheel and motor MOI, the measured z-axis testbed MOI, and the rotation rate of the wheels. The comparison is summarized in Table 20.

Table 21. Measured and Predicted ADCS Testbed Angular Velocity and Period

Source	Velocity	Period
Predicted	0.10 rad/s	62 s
Observed	0.22 rad/s	29 s
% Difference	112 %	53%

The predicted period was twice the length of the observed period and after examining several environmental factors such as air drag, imperfections in the air bearing, and wobble in the test bed which should all decrease speed, not increase it as seen, it was determined that a measurement error was to blame. The least reliable measurement was that of the reaction wheel MOI as the instrument used is intended for much larger objects up to 250 lbs, although the MOI measurement is technically larger than

the MOI measurement error of 2.92639e-07 kg-m^2 . Smaller MOI measuring devices exist for more precise measurements. Based on observed period, measured wheel speeds, and measured MOI of the testbed, a new predicted reaction wheel of 8.8e-05 kg-m^2 was calculated. The different reaction wheel MOI values are summarized in Table 22.

Table 22. Reaction Wheel MOI Measurement and Predictions

Source	Value
Measured MOI	4.12×10^{-5}kg-m^2
CAD-Predicted MOI	5.85×10^{-6}kg-m^2
Experiment-Predicted MOI	8.80×10^{-5}kg-m^2

The measured MOI was used in the simulations for this research. The source of the error in the CAD-predicted MOI was determined to be the selection of the incorrect material for the reaction wheel model, which has since been corrected.

VIII. Appendix II: Magnetic Torque Coils

Appendix II includes information on magnetic torque coil actuation and simulation as an avenue for future work in RTA, in which multiple verified controllers may be used to deal with different spacecraft modes or ACS actuator faults.

8.1 Magnetic Torque Coil Simulation Assumptions

It is assumed that no external torques beyond that applied by the ACS actuators, such as gravity gradient or aero torques, are acting on the CubeSat. The magnetic field of the Earth is modeled using the 11th generation of the International Geomagnetic Reference Field (IGRF). For the purposes of simulating magnetic torque, the spacecraft is assumed to be in a circular, equatorial orbit at an altitude of 400 kilometers on January 1, 2000.

8.2 AFIT Magnetic Torque Coils

The AFIT magnetic torque coils are comprised of 30 gage copper wire wrapped in 400 turns around a plastic frame and mount with an area of 0.0036 m^2, and are provided 500 mA, resulting in a maximum magnetic dipole moment of 0.72 A-m^2.[40] This coil design is pictured in Figure 67.

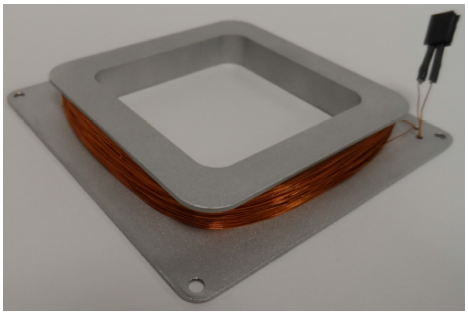

Figure 67. AFIT's Magnetic Torque Coil[40]

In the AFIT 6U CubeSat ADCS Testbed, there are 3 magnetic torque coils, each aligned with the body frame axes of the spacecraft. While the magnetic torque coils may be placed anywhere in the chassis as long as they are in an orthogonal orientation for 3-axis control, they are designed to be mounted inside the faces of the chassis as demonstrated in Figure 68.

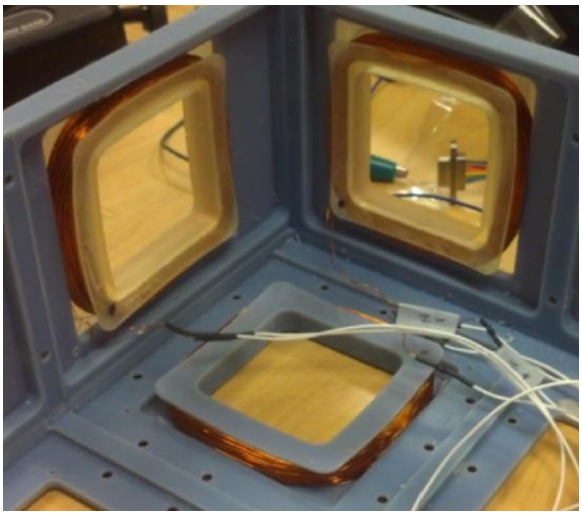

Figure 68. Three Magnetic Torque Coils Installed in a Plastic Chassis [12]

8.3 Magnetic Torque Coil Actuation

Magnetic torque coil arrays generate torque perpendicular to the Earth's magnetic field by applying electric current through a coiled wire. Unlike reaction wheel arrays, which generate an internal torque that does not change the total angular momentum of the spacecraft, magnetic torque coils produce an external torque which can increase or reduce the spacecraft's total angular momentum and are commonly used to desaturate momentum exchange devices. The external torque is also very useful for detumbling the spacecraft and placing it in a stable attitude relative to Earth's magnetic field. While magnetic torque coils have several advantages, their ability to provide attitude control is limited by the spacecraft's orientation and orbital position

relative to the direction of Earth's magnetic field. Magnetic torque coils produce less precise than many other actuators with a typical precision no better than 1 degree, and can interfere with onboard magnetometers; however, magnetic torque rods do not require any fuel.[96][22][51] The magnetic dipole moment $\vec{\mu}$ generated by a magnetic torque coil with an air core is

$$\mu = nIA \tag{48}$$

where n is the number of turns in the wire, I is the current passing through the wire and A is the cross sectional area of the space enclosed by the coil.[96]

The magnetic torque $\vec{\tau}$ produced by a single torque coil is calculated from

$$\vec{\tau} = \vec{\mu} \times \vec{B} \tag{49}$$

where $\vec{B}$ is the magnetic field vector of the Earth in the spacecraft body frame.[96] Arranging 3 magnetic torque coils orthogonal to one another in the spacecraft enables 2-axis attitude control. The torque vector ($\vec{\tau}$) generated by aligning identical magnetic torque coils with each of the body frame axes of the spacecraft is found from

$$\vec{\tau} = -\vec{B}^{\times} \boldsymbol{P}_t \vec{\mu} \tag{50}$$

where $\vec{B}^{\times}$ is a skew-symmetric matrix of components of $\vec{B}$ and $\boldsymbol{P}_t$ is a 3×3 identity matrix corresponding to the alignment of the magnetic torque coils with the body frame axes of the spacecraft.

8.4 Magnetic Torque Coil Desaturation of the Reaction Wheel Array

Due to the redundancy of a 4-wheel RWA, a unique control solution does not exist and a simple cross product law cannot be used to determine the magnetic torque

required to desaturate the reaction wheels. A method to desaturate a redundant reaction wheel array using a feedback term $\bar{u}^*$ presented by Hogan and Schaub [60] is shown here as

$$\bar{u}^* = -kD\vec{\psi} \tag{51}$$

where k is a constant gain, D is the reaction wheel moment of inertia about the axis of spin, and $\vec{\psi}$ is the vector of reaction wheel angular velocities. The resulting magnetic dipole moment $\bar{\mu}^*$ generated by a magnetic torque coil is calculated from

$$\bar{\mu}^* = -(\vec{B}^{\times}\mathbf{P}_t)^+ \left[\frac{1}{D}\mathbf{S}\right]\bar{u}^* \tag{52}$$

where the superscript + represents the pseudoinverse of the matrix. The resulting torque $\vec{\tau}^*$ produced by the magnetic torque coil's interaction with Earth's magnetic field may be calculated from

$$\vec{\tau}^* = -\vec{B}^{\times}\mathbf{P}_t\bar{\mu}^*. \tag{53}$$

While the torque generated by the torque coils would ideally be equal and opposite the torque generated by the reaction wheels ($\vec{\tau}^* + \vec{h}_{rwa} = 0$), this rarely happens, so Hogan and Schaub recommend the addition of another term $\Delta\bar{u}$ to account for the difference, as

$$\Delta\bar{u} = \left[\frac{1}{D}\mathbf{S}\right]^+(\vec{\tau}^* - \left[\frac{1}{D}\mathbf{S}\right]\bar{u}^*). \tag{54}$$

When added to the current reaction wheel angular momentum $\vec{h}_{rwa}$ and the feedback term $\bar{u}^*$, the new commanded change in reaction wheel angular momentum becomes

$$\vec{h}_{rwa_c} = \vec{h}_{rwa} + \bar{u}^* + \Delta\bar{u} = \left[\frac{1}{D}\mathbf{S}\right]^{\times}(\vec{\tau}^* - \left[\frac{1}{D}\mathbf{S}\right]\bar{u}^*). \tag{55}$$

8.5 Equations of Motion with Torque Coil and Reaction Wheel Arrays

In this section, the state derivative equations of motion for a system with RWA and magnetic torque control are presented.

The state vector $\bar{x}$ is defined here as:

$$\bar{x} = \begin{bmatrix} \bar{q} & \vec{\omega} & \vec{\psi} \end{bmatrix}^T = \begin{bmatrix} q_1, q_2, q_3, q_4, \omega_1, \omega_2, \omega_3, \psi_1, \psi_2, \psi_3, \psi_4 \end{bmatrix}^T \tag{56}$$

where the q terms are the elements of the quaternion describing the orientation, the ω terms are the spacecraft angular velocity about each principal axis, and the ψ terms are the angular velocity of each of the reaction wheels. The time rate of change of the state can then be expressed as: [88][60]

$$\dot{\bar{x}} = \begin{bmatrix} \mathbf{0}_{4\times4} & 0.5\boldsymbol{Q}_{4\times3} & \mathbf{0}_{4\times4} \\ \mathbf{0}_{3\times4} & -\boldsymbol{I}^{-1}\omega^{\times}\boldsymbol{I} & -\boldsymbol{I}^{-1}\omega^{\times}\boldsymbol{S} \\ \mathbf{0}_{4\times4} & \mathbf{0}_{4\times3} & \boldsymbol{L}_{4\times4} \end{bmatrix} \bar{x} - \begin{bmatrix} \mathbf{0}_{4\times4} \\ \boldsymbol{I}^{-1} \\ \mathbf{0}_{4\times3} \end{bmatrix} \boldsymbol{S}\dot{\vec{\psi}} + \begin{bmatrix} \mathbf{0}_{4\times4} \\ \boldsymbol{I}^{-1} \\ \mathbf{0}_{4\times3} \end{bmatrix} \vec{M} + \begin{bmatrix} \mathbf{0}_{4\times4} \\ \mathbf{0}_{3\times4} \\ \boldsymbol{U}_{4\times4} \end{bmatrix} \dot{\vec{\psi}} \tag{57}$$

where $\vec{M}$ is a vector of external torques acting on the spacecraft body axes, $\boldsymbol{U}_{4\times4}$ is a 4×4 identity matrix, $\boldsymbol{Q}_{4\times3}$ is

$$\boldsymbol{Q}_{4\times3} = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \tag{58}$$

and $\boldsymbol{L}_{4\times4}$ is the evolution of the wheel speeds as a function of the magnetic field, calculated from

$$\boldsymbol{L}_{4\times4} = -k * \left(\boldsymbol{U}_{4\times4} + \left[\frac{1}{D}\boldsymbol{S}\right]^{+} \left(\boldsymbol{B}^{\times}\boldsymbol{P}_t(\boldsymbol{B}^{\times}\boldsymbol{P}_t)^{+} - \boldsymbol{U}_{3\times3} \right) \left[\frac{1}{D}\boldsymbol{S}\right] \right). \tag{59}$$

8.6 Compositionaal Architecture with Magnetic Torque Coils

In Section 4.3.1, the compositional architecture of the system was described. In the model with magnetic torque coils, the coils are also broken down individually in the architecture. The flow of signals between the components for the model that includes magnetic torque coils is depicted in Figure 69. When the magnetic torque coils are included, a commanded magnetic dipole moment is sent to each coil. The resulting torque generated by the magnetic torque coil array is added to the total spacecraft structure angular momentum in the structure.

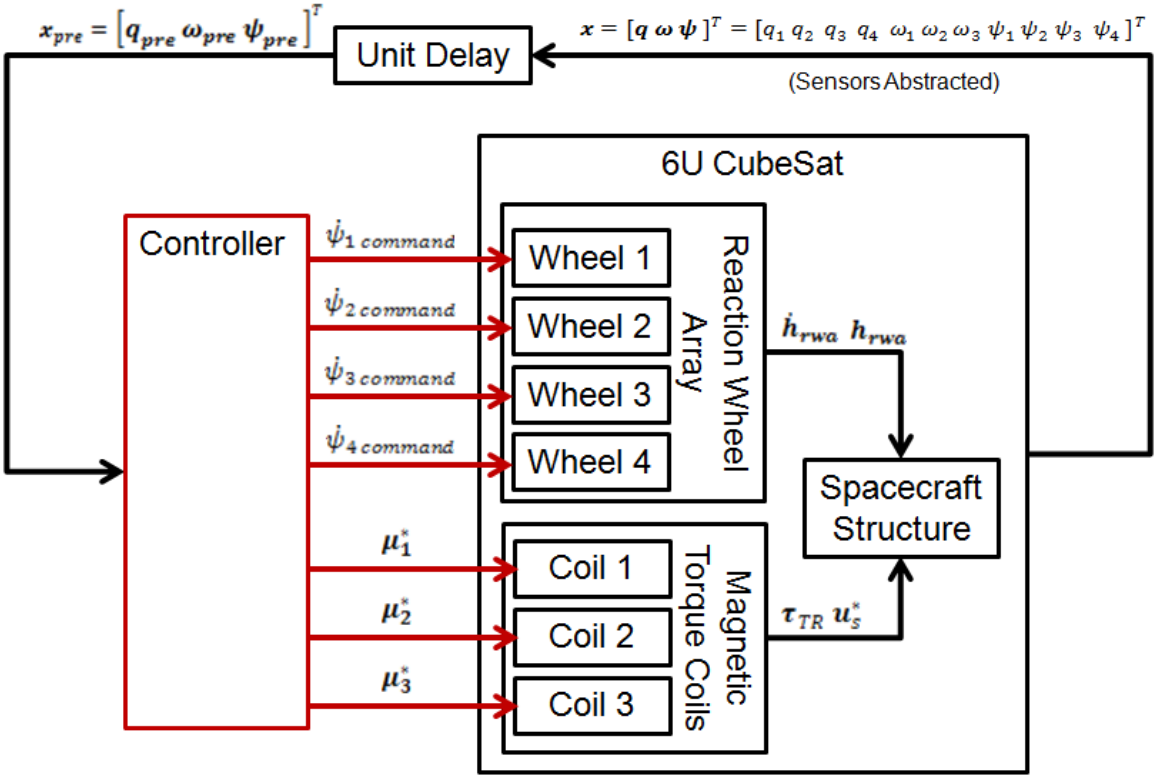

Figure 69. 6U CubeSat Singal Flow for Model with Reaction Wheel Array and Magnetic Torque Coil Actuation

8.7 Simulation of PID Control with Reaction Wheel and Magnetic Torque Coil Actuation

Using the methods, assumptions, and constants from Sections 4.6, 4.7.1, and 4.7.2, a simulation was conducted to show the system requirements are met by the PID controller design that uses magnetic torque coils to dump momentum from the reaction wheels. Just as seen in the reaction wheel only case, all of the testable requirements are met in this simulation. A summary of the settling times, rise time, and percent overshoot is displayed in Table 23.

Table 23. Control Systems Analysis Requirements Summary for Simulation with Magnetic Torque Coils and Reaction Wheel Array

Req't	Variable	Value	units
R10	$T_{s2\%}$	11.4	s
R11	$T_{s5\%}$	12.7	s
R12	$T_{\omega_{max}}$	7.8	s
R13	$\%OS$	13.16	%

Just as seen in Section 5.3.1, plots were created as shown in Figure 70 and Figure 71, to show satisfaction of these requirements.

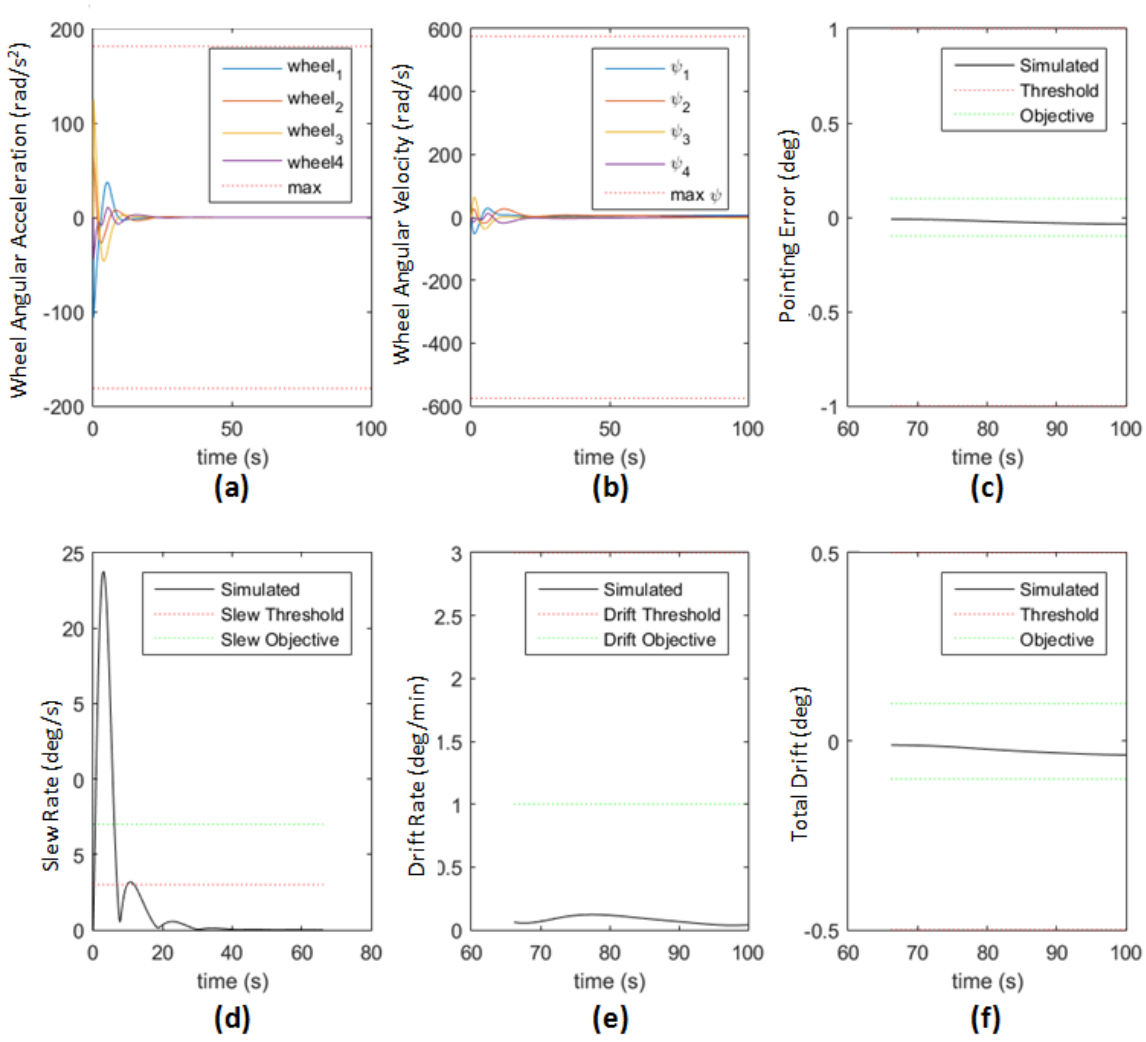

Figure 70. Performance Requirements for Simulation with Magnetic Torque Coils and Reaction Wheel Array

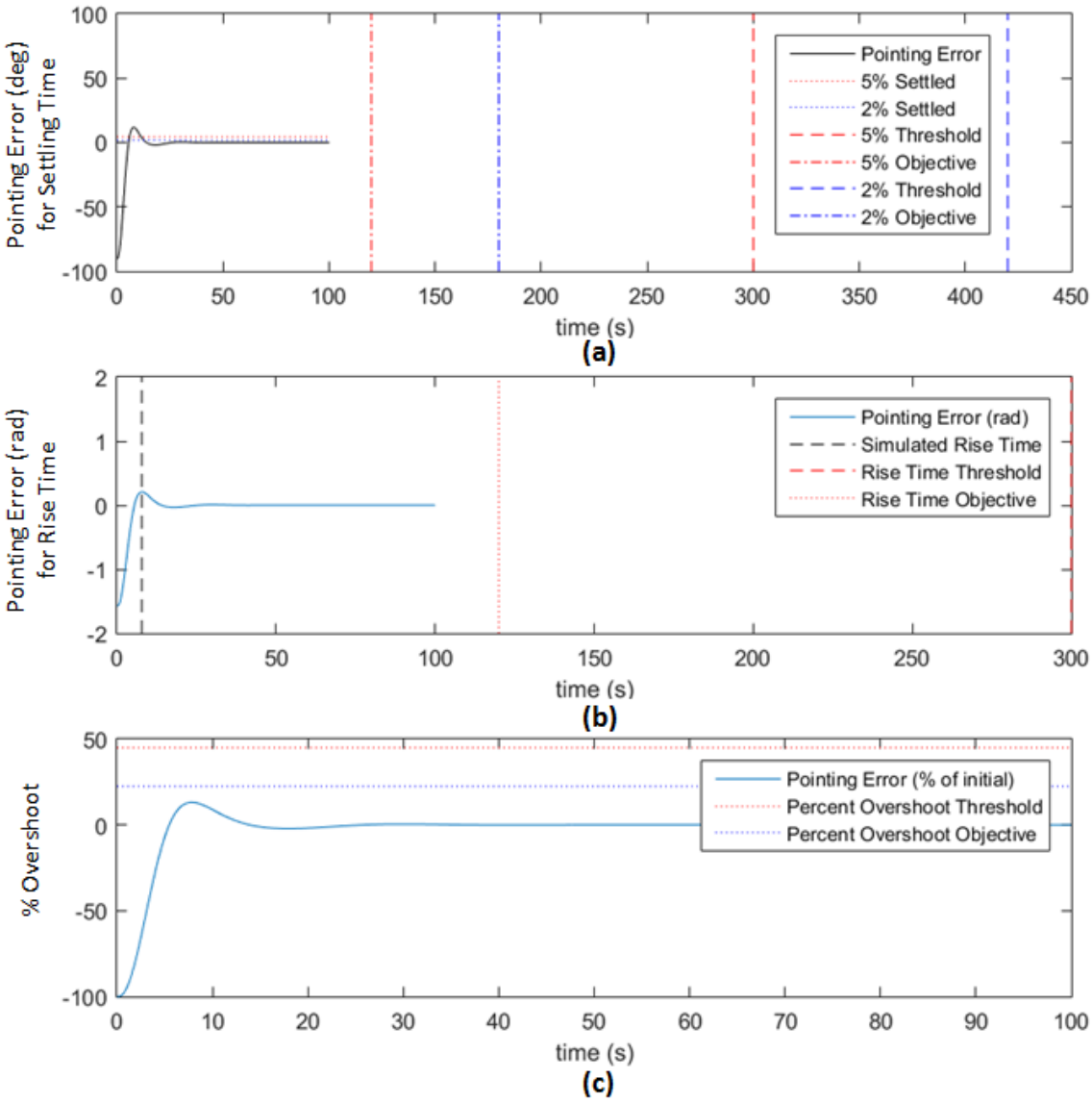

Figure 71. Control Systems Analysis Requirements for Simulation with Magnetic Torque Coils and Reaction Wheel Array

Bibliography

[1] "Ariane 5 Explosion". Public Domain Image. Available at *http://personal.victoria.ac.nz/stephen_marshall/SE/Failures/media/Ariane5_Explosion.jpg*.

[2] "Ariane 5 Launch". Public Domain Image credited to SpacePlex. Available at *http://theavion.com/wp-content/uploads/2016/02/a5_launch_14august2008_lr.jpg*.

[3] "An artist's concept portrays a NASA Mars Exploration Rover on the surface of Mars". Public Domain Image credited to NASA/JPL/Cornell University. Available at *http://photojournal.jpl.nasa.gov/catalog/PIA04413* .

[4] "Artists rendering of the Mars Climate Orbiter". Public Domain Image. Available at *http://www.vitalstatistics.info/uploads/mars%20climate%20orbiter.jpg*.

[5] "Choose a Solver". MATLAB R2015b Documentation, 2015. Available at *http://www.mathworks.com/help/simulink/ug/types-of-solvers.html*.

[6] "Comprehensive Static Analysis Products Using Polyspace Products: A Solution to Today's Embedded Software Verification Challenges". World Wide Web Page, 2013. Available at *http://www.mathworks.com/tagteam/77340_91517v02_30211_Polyspace-WhitePaper_final.pdf*.

[7] "Mission Specialists grapple Intelsat during". NASA. Available at *http://dayton.hq.nasa.gov/IMAGES/LARGE/GPN-2000-001035.jpg*.

[8] *An Assessment of Space Shuttle Flight Software Development Processes*. Technical report, National Academy Press, Washington D.C., 1993.

[9] "Mars Climate Orbiter: Mishap Investigation Board", 1999. Available at *ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf*.

[10] *The Economic Impacts of Inadequate Infrastructure for Software Testing*. Technical Report Planning Report 02-3, 2002.

[11] *DoD Autonomy Test and Evaluation, Verification and Validation (TEVV) Investment Strategy*. Technical report, Assistant Secretary of Defense / Research and Engineering (ASD/R&E), 2015.

[12] "2015 AFIT 6U EDU CubeSat Bus Interface Control Document (ICD)", Jul 2015. Air Force Institute of Technology, Version 1.0.

[13] Akers, S. B. "Binary Decision Diagrams". *IEEE Transactions on Computers*, C-27(6):509–516, 1978.

[14] Alur, R. *Principles of Cyber-Physical Systems*. The MIT Press, Cambridge, MA, 2015.

[15] Augustine, N. R. *Augustine's Laws*. American Institute of Aeronautics and Astronautics, Inc., Reston, VA, 1997.

[16] Baier, C. Katoen J.P. *Principles of Model Checking*. MIT Press, Cambridge, MA, 2008.

[17] Bak, S. Chivukula D.K. Adekunle O. Sun-M. Caccamo M. and L. Sha. "The System-level Simplex Architecture for Improved Real-time Embedded System Safety". *Proceedings of the 2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, 91–99. IEEE Computer Society, San Francisco, CA, Apr 2009.

[18] Bak, S. Manamcheri K. Mitra S. and M. Caccamo. "Sandboxing Controllers for Cyber-Physical Systems". *Proceedings of the 2011 IEEE/ACM International Conference on Cyber-Physical Systems*, 3–12. IEEE Computer Society, Chicago, IL, Apr 2011.

[19] Barrett, C. Conway C. Deters M. Hadarean-L. Jovanovic D. King T. Reynolds A. and C. Tinelli. "CVC4". *Proceedings of the 23rd International Conference on Computer Aided Verification*, 171–177. Springer, Snowbird, UT, Jul 2011.

[20] Box, G. E. P. and K. B. Wilson. "On the Experimental Attainment of Optimum Conditions". *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(1):1–45, 1951.

[21] Bradley, S. P. Hax A. C. Magnanti-T. L. "Applied Mathematical Programming", 1977. Chapter 13: Nonlinear Programming, pp. 410-464, Available at *http://web.mit.edu/15.053/www/AMP-Chapter-13.pdf*.

[22] Brown, C.D. *Elements of Spacecraft Design*. Second Edition, American Institute of Aeronautics and Astronautics, Inc., Reston, VA, 2002.

[23] Brown, L. P. "Agent based simulation as an exploratory tool in the study of the human dimension of combats", 2000. Masters Thesis, Naval Postgraduate School, Monterey, CA.

[24] Bruttomesso, R. Cimatti A. Franzn A. Griggio-A. and R. Sebastiani. "The Mathsat4 SMT solver". *20th International Conference on Computer Aided Verification*, 337340. Springer, Princeton, NJ, Jul 2008.

[25] Butler, R. "What Is Formal Methods?" NASA Langley Research Center Formal Methods Program, Available at *http://shemesh.larc.nasa.gov/fm/fm-what.html*.

[26] Buyske, S. and R. Trout. "Advanced Design of Experiments", 2001. Statistics 591 Lecture Series, Rutgers University.

[27] Chen, X. Abraham E. Sankaranarayanan S. "FLOW*: An Analyzer for Non-linear Hybrid Systems". *Computer Aided Verification (CAV), LNCS*, 8044:258–263.

[28] Christ, J. Hoenicke J. and A. Nutz. "SMTInterpol: An Interpolating SMT Solver". *19th International Workshop on Model Checking Software, SPIN 2012*, 248–245. Springer, Oxford, UK, Jul 2012.

[29] Cioppa, T. M. "Efficient nearly orthogonal and space-filling experimental designs for high-dimensional complex models", 2002. Doctoral Dissertation, Naval Postgraduate School, Monterey, CA.

[30] Clark, M. et al. "A study on Run Time Assurance for Complex Cyber Physical Systems". Air Force Research Laboratory, Aerospace Systems Directorate, Wright-Patterson AFB OH, Available at *http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA585474*, year = 2013.

[31] Clark, M. Fifarek A. Hamil J. and B. Hulbert. "ATEA TO9 Final Report: Autonomous and Advanced Control Ssytem Development, Integration, and Validation", Mar 2015. Technical Report, WPAFB.

[32] Clark, M. Kearns K. Overholt J. Gross-K. Barthelemy B. and C. Reed. "Air Force Research Laboratory Test and Evaluation, Verification and Validation of Autonomous Systems Challenge Exploration", 2014. Air Force Research Laboratory, Wright-Patterson AFB, OH.

[33] Clarke, E. M. and E. A. Emerson. "Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic". *Logic of Programs Workshop*, 52–71. Springer-Verlag, London, UK, 1981.

[34] Clarke, E. M. Klieber W. Novacek M.-Zuliani P. "Model Checking and the State Space Explosion Problem". *Tools for Practial Software Verification*, 1–31. Springer Berlin Heidelberg, 2010.

[35] Cohen, B. A. Hayne P. O. Paige-D. A. Greenhagen B. T. "Thoughts on Fault Tolerance in CubeSats and Small Sats", 2014. Annual Meeting of the Lunar Exploration Analysis Group.

[36] Corporation, Palisade. "Monte Carlo Simulation", 2016. Available at *http://www.palisade.com/risk/monte_carlo_simulation.asp*.

[37] Cousot, P. Cousot R. Feret J. Min-A. Rival X. "The Astre Static Analyzer", 2006. Available at *http://www.astree.ens.fr/* .

[38] Crenshaw, T.L. Gunter E. Robinson C.L. Shaw-L. Kumar P.R. "The Simplex Reference Model: Limiting Fault-propagation due to Unreliable Components in Cyber-physical System Architectures". *Proceedings of the 28th IEEE Real-Time Systems Symposium*, 400–412. IEEE Computer Society, Tucson, AZ, Dec 2007.

[39] Dahm, W. J. A. *Technology Horizons a Vision for Air Force Science & Technology During 2010-2030*. United States Air Force HQ, 2010.

[40] Dannemeyer, E. *Design and Analysis of an Attitude Determination and Control Subsystem (ADCS) for AFITs 6U Standard Bus*. Technical report, Air Force Institute of Technology, 2014.

[41] De Moura, L. and N. Bjorner. "Z3: An Efficient SMT Solver". *14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 337340. Springer, Budapest, Hungary, Apr 2008.

[42] De Moura, L. Ruess H. Sorea M. "Bounded Model Checking and Induction: From Refutation to Verification". *Proceedings of the 15th International Conference on Computer Aided Verification*, 14–26. Springer, Boulder, CO, Jul 2003.

[43] Denney, E. "Formal Methods for the Certification of Auto-generated Flight Code". NASA Ames Research Center.

[44] Dutertre, B. "Yices 2.2". *26th International Conference on Computer Aided Verification*. Vienna, Austria, Jul 2014.

[45] Dwyer, M. B. Avrunin G. S. Corbett J. C. "Property Specification Patterns for Finite-state Verification". *Proceedings of the Second Workshop on Formal Methods in Software Practice*, 7–15. ACM, Clearwater Beach, FL, Mar 1998.

[46] Een, N. Mishchenko A. Brayton R. "Efficient Implementation of Property Directed Reachability". *Proceedings of the 2011 Formal Methods in Computer-Aided Design (FMCAD)*, 125–134. IEEE, Austin, TX, Oct 2011.

[47] Feiler, P. H. "Supporting the ARP4761 Safety Assessment Process with AADL", 2014. Software Engineering Institute Carnegie Mellon University.

[48] Floyd, R.W. "Nondeterministic Algorithms". *Journal of the Association for Computing Macinery*, 14(4):636–644, 1967.

[49] Gacek, J. "Agacek/jkind", 2015. Available at *https://github.com/agacek/jkind*.

[50] Ghosh, S. Shankar N. Lincoln P. Elenius D. Li W. Steiener W. "Automatic Requirements Specification Extraction from Natural Language (ARSENAL)", Oct 2014. Final Report, SRI International, Menlo Park, CA.

[51] Griffin, M. D. and J. R. French. *Space Vehicle Design*. Second Edition, American Institute of Aeronautics and Astronautics, Inc., Reston, VA, 2004.

[52] Gross, K. H. "Application of Formal Methods to a 6U CubeSat Attitude Control System". *Proceedings of the 2015 Safe and Secure Software and Systems Symposium (S5)*. Air Force Research Laboratory, Dayton, OH, Jun 2015.

[53] Gross, K. H. Clark M. A. Hoffman J. A. Fifarek A.W. Rattan K.S. Swenson E. D. Whalen M. W. Wagner L. "Formally Verified Run Time Assurance Architecture of a 6U CubeSat Attitude Control Subsystem". *AIAA Infotech@ Aerospace, (AIAA 2016-0174)*, 222. American Institute of Aeronautics and Astronautics, San Diego, CA, Jan 2016.

[54] Gross, K. H. Hoffman J. A. Fifarek A.W. "Incremental Formal Methods Based Design Approach Demonstrated on a Coupled Tanks Control System". *IEEE High Assurance Systems Engineering*. Institute of Electrical and Electronics Engineers, Orlando, FL, Jan 2016.

[55] Gross, K. H. Hoffman J. Clark M. Swenson E. D. Cobb R. G. Whalen M. W. Wagner L. "Evaluation of Formal Methods Tools Applied to a 6U CubeSat Attitude Control System". *AIAA SPACE 2015 Conference and Exposition, SPACE Conferences and Exposition, (AIAA 2015-4529)*. American Institute of Aeronautics and Astronautics, Pasadena, CA, Sep 2015.

[56] Halbwachs, N. Caspi P. Raymond P. and D. Pilaud. "The Synchronous Data Flow Programming Language LUSTRE". *Proceedings of the IEEE*, 79(9):1305–1320, 1991.

[57] Hall, C. D. "Spacecraft Attitude Dynamics and Control", 2011. Virginia Polytechnic Instittue and State University.

[58] Hernandez, A. S. "Breaking barriers to design dimensions in nearly orthogonal latin hypercubes", 2008. Doctoral Dissertation. Naval Postgraduate School, Monterey, CA.

[59] Hernandez, A. S. Lucas T. W. and M. Carlyle. "Constructing Nearly Orthogonal Latin Hypercubes for Any Nonsaturated Run-Variable Combination". *ACM Transactions on Modeling and Computer Simulation*, 22(4), 2012.

[60] Hogan, E.A. and H. Schaub. "Three-axis attitude control using redundant reaction wheels with continuous momentum dumping". *AAS/AIAA Spaceflight Mechanics Meeting*. American Astronautical Society and American Institute of Aeronautics and Astronautics, Kauai, HI, 2013.

[61] Johnson, C.W. "The Natural History of Bugs: Using Formal Methods to Analyse Software Related Failures in Space Missions". *International Symposium of Formal Methods Europe*, 9–25. Springer, Newcastle, UK, Jul 2005.

[62] Kalinowski, S. "Update", August 27, 2015. Email.

[63] Lacan, P. Monfort J. N. Ribal L.V.Q. Deutsch A. Gonthier G. "Ariane 5: The software reliability verification process: the Ariane 5 Example", 1998. Proceedings for the DASIA 98 Conference on Data Systems in Aerospace, Athens, Greece.

[64] Leve, F. A. Hamilton B. J. and M. A. Peck. *Spacecraft Momentum Control Systems*. Springer International Publishing, Switzerland, 2015.

[65] Lewis, A. "A Brief on Controllability of Nonlinear Systems", 2002. Available at *http://www.mast.queensu.ca/ andrew/notes/pdf/2001a.pdf*.

[66] LLC., Space Electronics. "XR Series Moment of Inertia Instruments", 2015. Available at *http://www.space-electronics.com/Literature/XRSeriesImperial.pdf*.

[67] Masiello, T. "Air Force Research Lab Game Changers", 16 September 2014. Air Force Associante Air & Space Conference and Technology Exposition.

[68] McAllister, B. "NASA Future Challenges in Formal Methods". *Sixth NASA Formal Methods Symposium*. NASA Formal Methods, Houston, TX, Apr 2014.

[69] McKay, M. D. Beckman R. J. and W. J. Conover. "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code". *Technometrics*, 21(2):239–245, 1979.

[70] Meyers, R. H. Montgomery D. C. Anderson-Cook C. M. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. 3rd Edition, Wiley, Hoboken, NJ, 2009.

[71] Milton, J.S. and J. C. Arnold. *Introduction to Probability and Statistics*. The McGraw-Hill Companies, Inc., New York, NY, 2003.

[72] Motor., Maxon. "EC 32 flat 32 mm, brushless, 6 Watt, with Hall sensors", 2015. Available at *http://www.maxonmotorusa.com/maxon/view/product/motor/ecmotor/ecflat/ecflat32/339259*.

[73] Murphy, R. and J. Shields. "Defense Science Board Task Force Report: The Role of Autonomy in DoD Systems", July 2012. Technical Report CD 1172, Office of the Under Secretary of Defense for Acquisition, Technology and Logistics, Washington, DC.

[74] Paunicka, J. and D. Stuart. "Boeing CerTA FCS CPI Briefing". *Proceedings of the 2009 Safe and Secure Software and Systems Symposium (S5)*. Air Force Research Laboratory, Dayton, OH, Jun 2009.

[75] Queille, J. P. and J. Sifakis. "Specification and Verification of Concurrent Systems in CESAR". *Proceedings of the 5th Colloquium and International Symposium on Programming*, 337–351. Springer-Verlag, London, UK, 1982.

[76] Reis, A. Barth A. and C. Pizano. "Browser Security: Lessons Learned from Google Chrome". *Communications of the ACM*, 52(8):45–49, 2009.

[77] Rudd, L. and H. Hecht. "Certification Techniques for Advanced Flight Critical Systems", 2008. WPAFB Technical Report.

[78] Samson, J. R. "Thoughts on Fault Tolerance in CubeSats and Small Sats". *The Eighth Workshop on Fault-Tolerant Spaceborne Computing Employing New Technologies*. Albuquerque, NM, May 2015.

[79] Sanchez, S. M. "Nearly Orthogonal Latin Hypercube Design Excel Spreadsheet", 2005. Naval Post Graduate School.

[80] Seto, D. Ferriera E. and T. Marz. "Case Study: Development of a Baseline Controller for Automatic Landing of an F-16 Aircraft Using Linear Matrix Inequalities (LMIs)". Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Technical Report CMU/SEI-99-TR-020, Available at *http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13489* , year = 2000.

[81] Sha, L. "Using simplicity to control complexity". *IEEE Software*, 18(4):20–28, 2001.

[82] Software Engineering Institute, Carnegie Mellon University. "Architecture Analysis and Design Language", 2015. Available at *http://www.sei.cmu.edu/architecture/research/model-based-engineering/aadl.cfm*.

[83] Storm, W. Aiello T. Whalen M. "Highlights of the Certa FCS CPI Program". *Proceedings of the 2009 Safe and Secure Software and Systems Symposium (S5)*. Air Force Research Laboratory, Dayton, OH, Jun 2009.

[84] Swenson, E. "Euler Angles", 2014. MECH 632 Lecture Slides.

[85] Swenson, E. "Introduction", 2014. MECH 632 Lecture Slides.

[86] Swenson, E. "Notation, Reference Frames, and Rotation Matrices", 2014. MECH 632 Lecture Slides.

[87] Swenson, E. D. "6U CubeSat Design, Build, Test", 2014. Air Force Institute of Technology.

[88] Swenson, E. D. "REBEL RWA Analysis", 2015. Air Force Institute of, Wright-Patterson AFB, OH.

[89] Swihart, D. E. Barfield A. F. Griffin-E. M. Lehmann R. C. Whitcomb S. C. Skoog M. A. Flynn B. and K. E. Prosser. "Design, Integration and Flight Test of an Automatic Ground Collision Avoidance System". *Gyroscopy and Navigation*, 2(12):84–91, 2011.

[90] Thalheim, B. Jaakkola H. and Y. Kijoki. *Information Modelling and Knowledge Bases*. IOS Press, Inc., Fairfax, VA, 2014.

[91] The Mathworks, Inc. "Simulink Design Verifier: Identify and isolate design errors and generate tests", 2015. Available at *http://www.mathworks.com/products/sldesignverifier/index.html*.

[92] Tibbs, M. *Design and Test of an Attitude Determination and Control System for a 6U CubeSat using AFIT's CubeSat Testbed*. Technical report, Air Force Institute of Technology, 2015.

[93] Univeristy, Defense Acquisition. "Derived Requirements". World Wide Web Page. Glossary of Defense Acquisition Acronyms and Terms, Fifteenth Edition, December 2012Available at *https://dap.dau.mil/glossary/Pages/1781.aspx*.

[94] Univeristy, Defense Acquisition. "Verification". World Wide Web Page. Glossary of Defense Acquisition Acronyms and Terms, Fifteenth Edition, December 2012Available at *https://dap.dau.mil/glossary/pages/2860.aspx*.

[95] Weisstein, E. W. "Monte Carlo Method", 2016. MathWorld – A Wolfram Web Resource. Available at *http://mathworld.wolfram.com/MonteCarloMethod.html*.

[96] Wertz, J. R. Everett D. F. Pushcell-J.J. *Space Mission Engineering: The New SMAD*. Microcosm Press, 2011.

[97] Whalen, M. W. Gacek A. Cofer D.-Murugesan A. Heimdahl M. Rayadurgam S. "Your "What" Is My "How": Iteration and Hierarchy in System Design". *IEEE Software*, 30(2):54–60, 2013.

[98] Wie, B. *Space Vehicle Dynamics and Control*. Second Edition, American Institute of Aeronautics and Astronautics, Inc., Reston, VA, 2008.

[99] Wiegers, K. E. "Writing Quality Requirements", May 1999. Process Impact, Oregon, Available at *http://www.processimpact.com/articles/qualreqs.html*.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24-03-2016	Master's Thesis	Jan 2015 — Mar 2016

4. TITLE AND SUBTITLE

Evaluation of Verification Approaches Applied to a Nonlinear Control System

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Gross, Kerianne H., USAF Civilian

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENY-MS-16-M-214

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Aerospace Systems Directorate
Air Force Research Laboratory
2210 8th Street, Bldg 146
WPAFB, OH 45433-7765
Matthew Clark (937) 713-7004, Matthew.Clark.20@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RQQA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A. Approved for Public Release;Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Exhaustive test of complex and autonomous systems is intractable and cost prohibitive; however, design analysis techniques such as formal methods and design methodologies such as Run Time Assurance (RTA) could provide supplementary certification evidence early in system design. In this research, a 6U CubeSat Attitude Control Subsystem (ACS) is used as a challenge problem to evaluate the application of non-traditional verification methodologies such as formal methods and run time assurance architectures in conjunction with more traditional verification techniques. Thirteen hypothetical requirements are presented and formally defined. Strengths and weaknesses of the verification techniques are exposed in order to recommend capability expansions for further development. In analyzing the application of different formal methods tools, a new approach to verification was created to provide evidence of requirement satisfaction that leverages the capabilities of formal methods in conjunction with traditional verification techniques such as simulation cases, space filling experimental design simulation, and mathematical feasibility analysis.

15. SUBJECT TERMS

Formal Methods Analysis, Run Time Assurance, CubeSat, Attitude Control System, Reaction Wheel Array, Verification, Simulation, Space Filling Experimental Design, Nonlinear Control, Autonomy

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Eric D Swenson, AFIT/ENY
U	U	U	UU	193	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x7479; eric.swenson@afit.edu