3-24-2016

Analysis of a Near Real-Time Optimal Attitude Control for Satellite Simulators

Ryan M. Patrick

ANALYSIS OF A NEAR REAL-TIME OPTIMAL ATTITUDE CONTROL FOR SATELLITE SIMULATORS

THESIS

Ryan M. Patrick, 1st Lt, USAF

AFIT-ENY-MS-16-M-232

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENY-MS-16-M-232

ANALYSIS OF A NEAR REAL-TIME OPTIMAL ATTITUDE CONTROL FOR SATELLITE SIMULATORS

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

Ryan M. Patrick, B.S.

1st Lt, USAF

March 8, 2016

AFIT-ENY-MS-16-M-232

ANALYSIS OF A NEAR REAL-TIME OPTIMAL ATTITUDE CONTROL FOR SATELLITE SIMULATORS

THESIS

Ryan M. Patrick, B.S.
1st Lt, USAF

AFIT-ENY-MS-16-M-232

Abstract

Dynamic optimization of spacecraft attitude reorientation maneuvers can result in significant savings in attitude determination and control system size, mass, and power. Optimal control theory is generally applied using an open loop trajectory which is vulnerable to disturbances. A closed loop implementation of optimal control has been difficult to achieve due to the computational requirements needed to quickly compute solutions to the optimal control problem. This research focuses on evaluating a near real-time optimal control (RTOC) system for large angle slew maneuvers on the Air Force Institute of Technology's spacecraft simulator called SimSat. A near RTOC algorithm computes optimal control solutions at a rate of 0.4 Hz using a pseudospectral-based solver. The solutions or trajectories are then resampled at a fixed time step of 100 Hz and fed forward to a closed loop on SimSat. This algorithm is developed and tested on the hardware and compared to simulated and hardware results of a proportional-integral-derivative (PID) controller and an open loop optimal control controller for 90 degree and 180 degree Z-axis rotations. The benefits of decreased time to complete the maneuver and increased accuracy at the end of the optimal maneuver are shown to be improvements over traditional over PID control and open loop optimal control.

Acknowledgements

First I would like to thank my wife for all of her support in this endeavor. I cannot express my gratitude for your understanding as I worked through all of the trials and tribulations of my thesis research. I would have burned out a long time before getting this research done had you not been there for me every night as I came home frustrated with some aspect of my project. You and our daughter are the reason I kept going and managed to finish my work.

Next, I would like to thank my thesis advisor, Dr. Eric Swenson, for all of his help on this project. He got me started before I even knew what optimal control was and helped me every step of the way. Dr. Swenson allowed me to make mistakes and learn from them without holding my hand through the process. I don't know where I would be without his sound advice and practical knowledge.

I'd also like to thank Major Cole Doupe and Captain Ryan Carr. Maj Doupe was finishing up his dissertation research on SimSat as I was getting started and was willing to let me sit in on his research and teach me how to operate SimSat. After I started my tests he was never more than an email away to help me troubleshoot my errors in my MATLAB code and get the testbed working again after I had messed something up. Capt Carr was my GPOPS guru; helping me bring my computational time down from hours to seconds. His expertise in optimal control helped me bring all of the previous work that had been done in simulation to life on the actual harwdare.

Finally I'd like to thank my friends and family for keeping me sane in my year-and-a-half at AFIT.

Ryan M. Patrick

Table of Contents

Page

List of Figures

Page

List of Tables

Analysis of a Near Real-Time Optimal Attitude Control for Satellite Simulators

I. Introduction

1.1 Motivation

Optimal control theory is an extension of the calculus of variations that applies the methods of static optimization to a dynamic control problem [7]. Optimal control theory can be applied to generate the "optimal" or "near optimal" control input for a given performance index to generate a trajectory that maximizes or minimizes the performance index. For instance, a spacecraft's trajectory could be optimized for the minimal amount of fuel for an orbit transfer maneuver or an aircraft's autopilot could optimize a route from one location to another that takes the least amount of time to complete. Optimal control theory is generally applicable to all engineering disciplines. Commuters normally use optimal control theory every day to determine the shortest route home from work during rush hour traffic which may not be the shortest route but may be the one that minimizes time.

The application of optimal control theory to the spacecraft reorientation problem may likely be of interest to all stakeholders in the satellite design process. Many remote sensing satellites have requirements for rapid, accurate slewing. A spacecraft using optimal control theory could be applied so that these slew maneuvers could be completed in less time or with less control effort than a spacecraft using traditional control techniques. This would allow an imaging satellite for instance to gather more images on a single pass or switch between modes in less time. However, optimal control theory is generally applied as open-loop with little to no feedback to correct for errors along the manuever. Any disturbance that is not anticipated, such as gravity-gradient torques or air drag, can cause the satellite to deviate from the optimal trajectory if the disturbances are not modeled. If left uncorrected, these effects may result in a pointing error at the end of a maneuver or in an absolute worst-case scenario the loss of the satellite. These disturbances must be accounted for or corrected in order for optimal control techniques to be applied to the spacecraft reorientation problem.

One method of counteracting these disturbances is near real-time optimal control (RTOC) where the controller computes a new optimal control solution on a frequent basis. The spacecraft can then follow the most recently computed optimal solution for a given performance index. While computationally intensive, near RTOC may provide a time-optimal reorientation solution in the presence of external disturbances.

1.2 Research Objectives

The goal of this research is to evaluate the application of optimal control theory to the spacecraft reorientation maneuver problem. This will be accomplished in order to reduce the time required to complete the maneuver through the application of near RTOC techniques. AFIT's spacecraft simulator, SimSat, shown in Fig. 1.1, is the test platform on which a near RTOC controller is compared with an open-loop optimal controller and a traditional proportional-integral-derivative (PID) controller. This research will apply near RTOC techniques developed first by Ross *et. al* [34] and modified for planned use on SimSat by McFarland [26] Parameters in the near RTOC algorithm affecting trajectory switching and optimal control solver settings will be modified and the effects of changing those parameters on the performance of the near RTOC controller will be analyzed.

1.3 Methodology

The near RTOC controller evaluated on a AFIT's SimSat to assess the feasibility of decreasing the time required to complete a reorientation maneuver. A PID controller, an open-loop optimal controller, and a near RTOC controller were designed and implemented on SimSat. Two different rest-to-rest reorientation maneuvers were performed over all three axes and state data was collected. Near RTOC parameters determining when the algorithm switched from one trajectory to another were varied in order to determine which values yielded the best performance. Metrics to compare optimal manuevers with non-optimal maneuvers were developed and the three methods of spacecraft control will be compared.

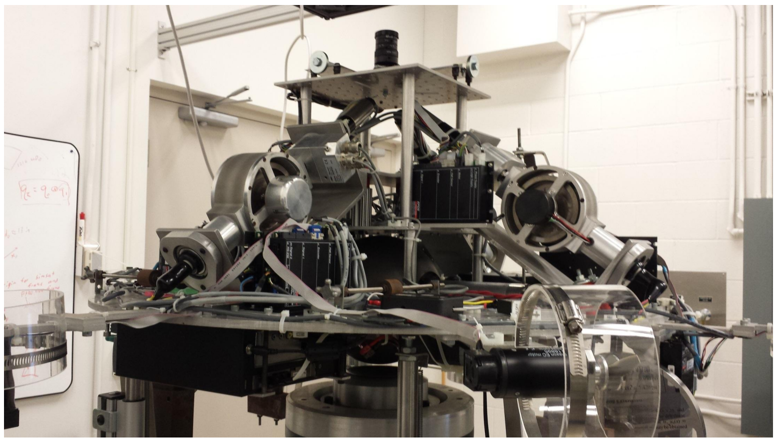

Figure 1.1: SimSat, AFIT's 2nd-generation Simulated Spacecraft

1.4 Thesis Outline

Chapter II provides an extensive literature review on spacecraft dynamics, optimal control theory, spacecraft simulators, and applications of optimal control on spacecraft. Chapter III describes SimSat's hardware and software, and methods and results of preliminary hardware integration, and the methods used for simulating and physically implementing optimal control on the spacecraft simulator. Chapter IV provides the results and analysis of optimal control simulation and experimental testing on SimSAT II. Chapter V provides conclusions on the presented research and recommendations for future research.

II. Background

This chapter presents a literature review of topics relevant to this research. The first section reviews spacecraft rigid body dynamics in order to determine state equations of motion (EOM). The next section gives an overview of optimal control theory from which solutions will be tested on a the spacecraft simulator in order to improve reorientation performance. The following section provides a historical background on spacecraft simulators with the final section covering applied optimal control research and development activities as applied to spacecraft and spacecraft simulators. These two sections will provide a background on previous research completed using optimal control theory on spacecraft simulators.

2.1 Spacecraft Dynamics

2.1.1 Rigid Body Dynamics. For deriving the equations of motion for attitude dynamics, a spacecraft is treated as a rigid body with six degrees of freedom that allows for rotational as well as translational motion. This section will only discuss rotational degrees of freedom (DOF). The angular momentum of a rigid body is obtained by integrating over the entire body as can be seen from [43]

$$\vec{H}_0 = \int_{body} \vec{\rho} \times \vec{v}\, \mathrm{d}m. \tag{2.1}$$

The inertial velocity of a particle of the rigid body mass dm relative to the origin O can be written as

$$\vec{v} = \vec{\omega}^{bi} \times \vec{\rho}. \tag{2.2}$$

Therefore, the angular momentum of the rigid body about the origin O can be computed from[41].

$$\vec{H}_0 = \int_{body} (\vec{\rho} \times (\vec{\omega}^{bi} \times \vec{\rho}))\mathrm{d}m. \tag{2.3}$$

The vector $\vec{\rho}$ from the center of mass of the rigid body to dm and the instantaneous angular velocity vector of the body frame with respect to the inertial frame $\vec{\omega}^{bi}$ are expressed in the body frame as

$$\vec{\rho}_b = x\hat{b}_1 + y\hat{b}_2 + z\hat{b}_3 \tag{2.4}$$

and

$$\vec{\omega}_b^{bi} = \omega_1\hat{b}_1 + \omega_2\hat{b}_2 + \omega_3\hat{b}_3. \tag{2.5}$$

Eq. (2.3) can be rewritten as

$$\vec{H}_0 = \int_{body} \begin{bmatrix} (\omega_1(y^2+z^2) - \omega_2 xy - \omega_3 xz)\hat{b}_1 \\ (-\omega_1 xy + \omega_2(x^2+z^2) - \omega_3 yz)\hat{b}_2 \\ (-\omega_1 xz - \omega_2 yz + \omega_3(x^2+y^2))\hat{b}_3 \end{bmatrix} \mathrm{d}m. \tag{2.6}$$

Since $\vec{\omega}^{bi}$ is the same for all points on the rigid body, it can be extracted from the body integral and the mass moment of inertia I_b is now defined as

$$I_b = \begin{Bmatrix} \int_b (y^2+z^2)dm & -\int_b yx\,dm & -\int_b zx\,dm \\ -\int_b xy\,dm & \int_b (x^2+z^2)dm & -\int_b zy\,dm \\ -\int_b xz\,dm & -\int_b yz\,dm & \int_b (x^2+y^2)dm \end{Bmatrix}. \tag{2.7}$$

Eq. (2.6) can now be expressed as

$$\vec{H}_0 = I_b\vec{\omega}_b^{bi} \tag{2.8}$$

where I_b is assumed constant in the body frame because of the rigid body constraint. The inertial derivative of $\vec{H}_0$ with respect to time is

$$\dot{\vec{H}}_0 = I_b\dot{\vec{\omega}}_b^{bi} + \vec{\omega}_b^{bi} \times I_b\vec{\omega}_b^{bi}. \tag{2.9}$$

The time rate of change of the angular momentum is $\dot{\vec{H}}_0$ of the entire spacecraft is equivalent to the net external torque shown in Eq. (2.11) which is the rotational analog to the change in linear momentum $\vec{F}$. Dropping subscripts and superscripts in Eq. (2.9) by assuming all expressions from this point forward are expressed in the body frame results in

$$\dot{\vec{H}} = I\dot{\vec{\omega}} + \vec{\omega}^{\text{x}} I\vec{\omega} \tag{2.10}$$

where ω^{x} is a skew-symmetric matrix equivalent to performing a vector cross product.

It can be shown [42] that the inertial time derivative of angular momentum is also equal to the external moment acting upon the rigid body. This results in

$$\vec{M} = \dot{\vec{H}} = I\dot{\vec{\omega}} + \vec{\omega}^{\text{x}} I\vec{\omega} \tag{2.11}$$

where $\vec{M}$ is the vector sum of all external moments on the spacecraft. This vector equation is known in component form as Euler's equations which come from the Swiss mathematician Leonard Euler's laws of motion which extend Isaac Newton's laws of motion for particles to rigid bodies.[27] Solving Eq. (2.11) for the time rate of change of angular velocity $\dot{\vec{\omega}}$ is

$$\dot{\vec{\omega}} = -I^{-1}(\vec{\omega}^{\text{x}} I\vec{\omega} - \vec{M}). \tag{2.12}$$

2.1.2 Quaternions. The most common way to transform a vector in one reference frame to another is through a direction cosine matrix (DCM), also known as a rotation matrix [41]. This is a 3x3 orthonormal matrix. $\mathbf{R^{bi}}$ transforms a vector from an inertial frame $\hat{i}$ to a body-fixed frame $\hat{b}$ and can be expressed as

$$\mathbf{R^{bi}} = \{\hat{b}\} \cdot \{\hat{i}\}^T. \tag{2.13}$$

$\mathbf{R}^{bi}$ will transform a vector $\vec{v}_i$, which has its components expressed in the inertial frame, to the vector $\vec{v}_b$ which has its components expressed in the body frame through

$$\vec{v}_b = \mathbf{R}^{bi}\vec{v}_i. \tag{2.14}$$

The nine values of this DCM are interrelated three independent DOF and six constraints. The constraints are the following: every column or row must have a magnitude of one and the columns or rows must be orthogonal.

A rotation described using Euler angles which are expressed as three successive rotations about orthogonal frame axes [37]. This allows for a 3-DOF rotation that is three successive simple rotations expressed as direction cosine matrices. However, all Euler axis rotations have a singularity [41]. This singularity can be seen when one attempts to extract of Euler angles from a rotation matrix describing the full rotation. Due to the singularities, Euler angles are not the best choice for computation but they still provide an excellent method of describing and visualizing rotations.

An alternative method of representing rigid body rotations is the Euler axis/angle or eigenaxis representation. Any rigid body rotation can be described as a rotation about an angle that is fixed to the body and is stationary in an inertial reference frame [42]. This allows the three rotation sequence of Euler angles to be performed in one step. However, this approach also has a singularity at $0°$ rotation.

A third method of attitude representation, and the one most commonly used on spacecraft today, is quaternions. Based on the Euler angle/axis set, it consists of a 3x1 vector $\vec{q}$ and a scalar component q_4. Together, they form a 4x1 quaternion $\bar{q}$ consisting of a 3x1 vector $\vec{q}$ and an independent parameter q_4. Quaternions are singularity-free and provide compuational advantages over other rotational parameter sets due to their lack of triginometric computation. From a given Euler axis $\hat{a}$ and rotation angle ϕ, a quaternion can be defined as

$$\vec{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \hat{a}\sin\left(\frac{\phi}{2}\right) \tag{2.15}$$

and

$$q_4 = \cos\left(\frac{\phi}{2}\right). \tag{2.16}$$

resulting in a full quaternion of

$$\bar{q} = [\vec{q}\ q_4]^T = [q_1\ q_2\ q_3\ q_4]^T \tag{2.17}$$

Since quaternions have four parameters and only three are required, a constraint is required which can be written as [42]

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1. \tag{2.18}$$

The way to calculate quaternion error between two quaternion is to use the quaternion transmuted matrix which is defined by

$$\begin{bmatrix} q_{1,error} \\ q_{2,error} \\ q_{3,error} \\ q_{4,error} \end{bmatrix} = \begin{bmatrix} q_4'' & q_3'' & -q_2'' & q_1'' \\ -q_3'' & q_4'' & q_1'' & q_2'' \\ q_2'' & -q_1'' & q_4'' & q_3'' \\ -q_1'' & -q_2'' & -q_3'' & q_4'' \end{bmatrix} \begin{bmatrix} q_1' \\ q_2' \\ q_3' \\ q_4' \end{bmatrix} \tag{2.19}$$

where the quaternion error values found in the left array are the error quaternion parameters between q' and q'' [26].

In a 3D rotational dynamic system, a rotational rate cannot be directly integrated to obtain rotational parameters such as the Euler Angles or quaternions due to the path dependent nature of rotations [41]. The rate of change of quaternions can be described by the following equation

$$\dot{\bar{q}} = \mathbf{Q}(\bar{q})\vec{\omega}. \tag{2.20}$$

where $\mathbf{Q}$ is defined as

$$\mathbf{Q} = \frac{1}{2}\begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix}. \tag{2.21}$$

2.1.3 Reaction Wheels. The most common angular momentum control device used on spacecraft today is the reaction wheel. They are used for attitude control on many large and small spacecraft that require accurate pointing. [36]. A reaction wheel array (RWA) usually contains three (or more) reaction wheels that consist of a flywheel and motor with independent microcontrollers to manage the required angular speed and acceleration of each wheel in the RWA. The RWAs exchange momentum between the wheels and spacecraft, allowing for the spacecraft to change in attitude without having to apply an external torque. The total angular momentum of a spacecraft, in the absence of external torques, is constant. The total spacecraft angular momentum can be written as

$$\vec{H}_{tot} = \vec{H}_{S/C} + \vec{h}_{RW} \tag{2.22}$$

where $\vec{H}_{S/C}$ is the total angular momentum of the spacecraft and $\vec{h}_{rw}$ is the total angular momentum of the reaction wheels.

Eq. (2.12) can be now be written as

$$\dot{\vec{\omega}} = -I^{-1}(\vec{\omega}^{\times} I\vec{\omega} - M + \vec{\omega}^{\times}\vec{h}_{rw} + \dot{\vec{h}}_{rw}). \tag{2.23}$$

The reaction wheel angular momentum of a RWA $\vec{h}_{rw}$ with n reaction wheels is defined as

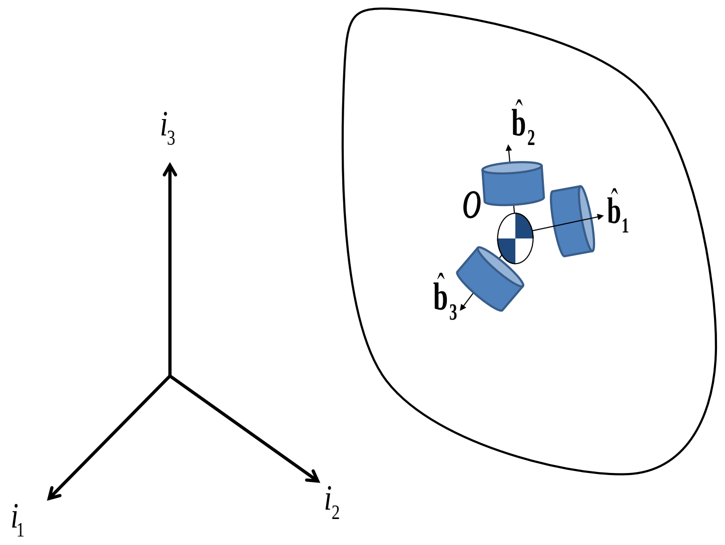

Figure 2.1: Representative Reaction Wheel Array in a Generic Spacecraft Body

$$\vec{h}_{rw} = D\vec{\psi}_1 + D\vec{\psi}_2 + D\vec{\psi}_3 + ...D\vec{\psi}_n \tag{2.24}$$

and the torque generated is

$$\dot{\vec{h}}_{rw} = \sum_{i=1}^{n} D\vec{\psi}_i. \tag{2.25}$$

where ψ_i is the angular velocity of the reaction wheel, $\dot{\psi}$ is the angular acceleration of the reaction wheel, and D is the scalar mass moment of inertia of the reaction wheel array about its spin axis. The reaction wheels are can be aligned with the body axes but often are not in the case of 4 or more wheels in order to create redundancy in case a wheel fails on orbit. A case where the reaction wheels are aligned with the body axes is shown in Fig. 2.1.

2.2 Optimal Control Theory

2.2.1 Optimization. This section will describe static optimization, such as sizing an airfoil for an aircraft or reaction wheels for a spacecraft. The act of obtaining the "best" result under given circumstances is optimization.[13] What is optimal and what is not depends on the problem at hand, but the techniques used to optimize a set of parameters are the same across engineering disciplines.[1] In engineering and mathematics, the optimization of a system usually refers to finding the desired extrema of a function. Optimal design is applying mathematical methods to find the extrema of a function that represents the defined optimality of system.

For optimal design problems, regardless of what is being optimized, there exists a standard mathematical notation. The problem consists of design variables, a cost function, and constraints. Design variables $\vec{x}$ are the parameters of the system being optimized that are varied by the optimizer to achieve the desired optimality. A cost function $f(\vec{x})$ is a function of one or more design variables. This cost function, also referred to as a performance index or an objective function $\boldsymbol{J}$ has extrema from which the optimum design variables can be determined. The equality constraint functions $h(\vec{x})$ such as $h_1(\vec{x}) = 2$ and inequality constraint functions $g(\vec{x})$ such as $g_1(\vec{x}) \leq 5$ enforce limits on one or more design variable that must be satisfied.

Optimal design problems can be linear or nonlinear. If both the cost and constraint functions contain only linear terms, the problem is classified as a linear programming (LP) problem that can be solved using LP techniques. However, if a single nonlinear term exists in either the cost or constraint functions, the problem is now classified as a nonlinear programming (NLP) problem. Most engineering optimal design problems are nonlinear.

Methods of formulating and solving NLP problems can be classified as direct or indirect. Direct methods use numerical methods to iteratively obtain and improve an optimal solution. Indirect methods use analytical solutions that involve the calculus of variations to minimize a function. While the best method to use is dependent on the optimization problem, indirect methods work best for simple problems with only a few constraints and states. Direct methods are used on most aerospace problems due to their highly coupled and nonlinear nature. This research will formulate the problem using indirect methods and then solve for optimal solutions using direct methods.

2.2.2 Optimal Control. Section 2.2.1 presents static optimization. Dynamic optimization extends the same theories to a dynamic system to determine the control and state histories over a finite time period to minimize some performance index or cost function.[7] This is also refered to as Optimal Control Theory (OCT). The constraint functions $h(\vec{x}, \vec{u})$ describing the dynamics of the system in the formulation of an optimal control problem are differential equations. The constraints can also contain a second design vector $\vec{u}$ that represent the control input to the system. This control input is able to manipulate the system's dynamics and drive the system to its desired end state. The cost function $\boldsymbol{J}$ of an optimal control problem is a functional, rather than a function, since the solution to minimize the performance index is now a function of functions.

The goal of optimal control of a system is to find a time history of the control inputs to the system that minimizes the performance index within the constraints of the system. There are four general types of optimal control problems: fixed final time and fixed final state; fixed final time and free final state; free final time and fixed final state; and free final time and free final state. This research will only consider the free final time and fixed final state problem which will be convered in depth in the next section.

2.2.3 Free Final Time and Fixed Final State Optimal Control. This section considers the methods of formulating and solving a free final time and fixed final state optimal control problem. While the problem can be solved using direct or indirect methods, this section will only consider the indirect methods to compute an analytical solution. This section will then identify where it is mathematically infeasible to use indirect methods and perferable to use direct methods. The direct methods approach used in this research will be discussed in Section 2.2.5.

The general form of a free final time, fixed final state optimal control problem is

$$\boldsymbol{J} = \phi(t_f, x_f) + \int_{t0}^{tf} L(t, \vec{x}, \vec{u})\mathrm{d}t \tag{2.26}$$

where ϕ is a function of the final time and state constraints and L is an integral cost function. The generalized system dynamics for this problem are described by the vector differential constraint

$$\dot{\vec{x}} = f(t, \vec{x}, \vec{u}). \tag{2.27}$$

This problem is subject to initial conditions

$$t_0 = t_0^* \tag{2.28}$$

and

$$\vec{x}(t_0) = \vec{x}_0^* \tag{2.29}$$

where t_0^* and $\vec{x}_0^*$ are the initial time and state, respectively. There are also constraints on the final condition given as

$$\psi(t_f, \vec{x}_f) = \vec{x}_f - \vec{x}_f^* = 0, \tag{2.30}$$

where ψ must contain at least one state with final condition $\vec{x}_f^*$.

The augmented cost functional is equal to[3]

$$\boldsymbol{J}^{'} = \Phi(t_f, \vec{x}_f, \nu) + \int_{t0}^{tf} [H(t, \vec{x}, \vec{u}) - \vec{\lambda}^T \dot{\vec{x}}] \mathrm{d}t \tag{2.31}$$

where Φ is the endpoint function equal to

$$\Phi = \phi(t_f, \vec{x}_f) + \nu^T \psi(t_f, \vec{x}_f) \tag{2.32}$$

and H is the Hamiltonian of the system defined as

$$H = L(t, \vec{x}, \vec{u}) + \vec{\lambda}^T f(t, \vec{x}, \vec{u}). \tag{2.33}$$

There are both first and second order differential conditions that need to be satisfied for a solution to be optimal. The first are the Euler-Lagrange (E-L) equations.[20] They are the state, costate, and stationary equations which are derived from the augmented performance index in Eq. (2.31). The state equation is

$$\dot{\vec{x}} = H_\lambda^T = f(t, \vec{x}, \vec{u}), \tag{2.34}$$

the costate equation is

$$\dot{\lambda} = -H_x^T(t, \vec{x}, \vec{u}, \vec{\lambda}), \tag{2.35}$$

and the stationary equation is

$$0 = H_u^T(t, \vec{x}, \vec{u}, \vec{\lambda}). \tag{2.36}$$

These three equations are subject to the boundary conditions in Eqns. (2.28), (2.29) and (2.30) and to the constraints

$$\Phi_{t_f}(t_f, \vec{x}_f, \nu) + L(t_f, \vec{x}_f, \vec{u}_f) + \Phi_{x_f}(t_f, \vec{x}_f, \nu) = 0 \tag{2.37}$$

and

$$\lambda_f = \Phi_{x_f}^T(t_f, \vec{x}_f, \nu). \tag{2.38}$$

If an optimal control problem has a state vector $\vec{x}$ of dimension n and a control vector $\vec{u}$ of dimension m, there are $2n$ differential equations from Eq. (2.34) and Eq. (2.35) and m differential equations from Eq. (2.36) that are solved while satisfying the n initial conditions in Eqns. (2.28) and (2.29) and n final conditions found by combining Eqns. (2.30), (2.37), and (2.38) [23].

If an analytical solution exists for a a free final time, fixed final state optimal control problem, the solution must meet the Weierstrass first order necessary condition and the

Legende-Clebsh second order sufficient condition in order to be considered an optimal solution for the problem [20]. The Weierstrass condition can be written as

$$H(t, \vec{x}, \vec{u}, \vec{\lambda}) - H(t, \vec{x}, \vec{u}^{*}, \vec{\lambda}) > 0 \tag{2.39}$$

and must be satisfied for all $\vec{u} \neq \vec{u}^{*}$ where $\vec{u}^{*}$ is the optimal control solution for the given performance index and $\vec{u}$ is all other admissible control paths. The Hamiltonian must be minimized at all time t along the optimal control solution. The Legendre-Clebsh condition is

$$H_{uu}(t, \vec{x}, \vec{u}, \vec{\lambda} \geq 0. \tag{2.40}$$

which also must be satisfied at all time t along the optimal control solution.

An analytical, closed-form solution may not exist when the dynamical constraints of the system are characterized by a system of nonlinear, coupled differential equations. In this research, the SIMSAT's governing dynamics and kinematics equations can be expressed by a set of seven coupled nonlinear differential equations. Instead of solving for optimal control solutions analytically, a pseudospectral-based direct method tool, GPOPS-II, is used.

2.2.4 Optimal Control Problem Formulation. Before discussing pseudospectral methods, we need to discuss the optimal control problem formulation. In order to evaluate the application of optimal control theory's application to spacecraft the methodologies developed in this research are tested on the SIMSAT, a satellite testbed that can be described as a dynamic system. The system dynamics of the attitude of SIMSAT in quaternion parameters can be found in Eq. (2.20). The kinematic equations of the body angular rates can be found in Eq. (2.12). $\vec{u}$ is the control vector of the optimal control problem and is related to the torque produced by the reaction wheels by the equation

$$\vec{u} = \dot{\vec{h}}_{rw}. \tag{2.41}$$

where

$$\vec{x} = \begin{bmatrix} \bar{q} \\ \vec{\omega} \\ \vec{h}_{rw} \end{bmatrix} \tag{2.42}$$

which has a dimension of 7x1. The performance index for the particular free final time and fixed final state control problem used in this research is

$$\boldsymbol{J} = \int_{t0}^{tf} \mathrm{d}t = t_f. \tag{2.43}$$

There is no integral cost function as in Eq. (2.26) because the only parameter being minimized is the final time.

2.2.5 Pseudospectral Methods. A pseudospectral-based direct method tool, GPOPS-II, is used in this research to generate or near-optimal optimal control solutions. Pseudospectral methods were originally used to solve partial differential equations [9] but now are commonly used to solve optimal control problems [33]. They discretize the state vector and control vector onto a Lagrange interpolating polynominal. There are a number of distribution methods for defining the locations of the discretized nodes of the Lagrange interpolating polynominal including: uniform distribution, Legendre-Gauss-Radau (LGR) points, Legendre-Gauss-Lobatto (LGL) points, Chebyshev-Gauss points, Chebyshev-Gauss-Radau points, Legendre-Gauss points, and Chebyshev-Gauss-Lobatto points [16]. GPOPS-II employs a LGR quadrature collocation method [28]. It is a Gaussian quadrature implicit integration method where collocation is performed at LGR points. GPOPS-II uses the discretized approximation of the state and control vectors to form a new approximate cost function. The cost function in Eq. (2.43) is minimized by GPOPS-II and results in a minimum-time optimal solution at the collocation points. It is not truely optimal because linear interpolation must be used outside of the collocation points. The number of minimum and maximum collocation points can be varied to form a more accurate or less accurate solution. GPOPS-II is referred to as GPOPS throughout the remainder of this document.

2.3 Spacecraft Simulators

2.3.1 Types of Spacecraft Simulators. All spacecraft, regardless of size or mission, represent a large investment in time and capital on the part of all stakeholders involved in the project. Therefore, the testing and validation of the spacecraft before it is launched into orbit is a crucial element of the design process. The attitude determination and control subsystem (ADCS) is a critical part of almost all spacecraft that allows the spacecraft to estimate inertial attitude and change its orientation. The ADCS poses unique challenges to being tested on the ground due to the difference in gravity between the testing facility and orbit.

The most common method to simulate near-orbit conditions for a spacecraft's ADCS is the use of an air bearing to create a frictionless boundary between two surfaces [10]. There are two types of air bearings: planar and rotational. Rotational air bearings resemble a ball-in-socket joint and allow for three rotational DOF but no translational DOF. This research only uses a rotational air bearing on the SimSat.

Rotational air bearings, also referred to as spherical air bearings, are the most widely used to test an ADCS subsystem. An ideal rotational air bearing would be in the shape of a sphere and allow for unconstrained motion in all three axes[35]. This would allow for a gravity torque-free environment that simulates the space enviornment found on-orbit. Creating a gravity torque-free environment of a spacecraft simulator using a rotational air bearing depends on the alignment of the center of mass of the simulated spacecraft with the air bearing's center of rotation [26]. The freedom of spin about a particular axis is dependent on the structure of the simulated spacecraft and its configuration relative to the spherical air bearing. A common configuration for spacecraft testbeds using a rotational air bearing has been the "tabletop" or "umbrella." This design allows for the Yaw axis to have full rotational range of motion but has limited rotation about the Roll and Pitch axes[19][12]. A more recent configuration is the "dumbbell" configuration that enables uninhibited rotation about the Yaw and Roll axes with limited rotation about the Pitch axis [8]. The “tabletop,” “umbrella,” and “dumbbell” configurations are shown left-to-right in Fig. 2.2. The SimSat is a ”tabletop” satellite simulator.

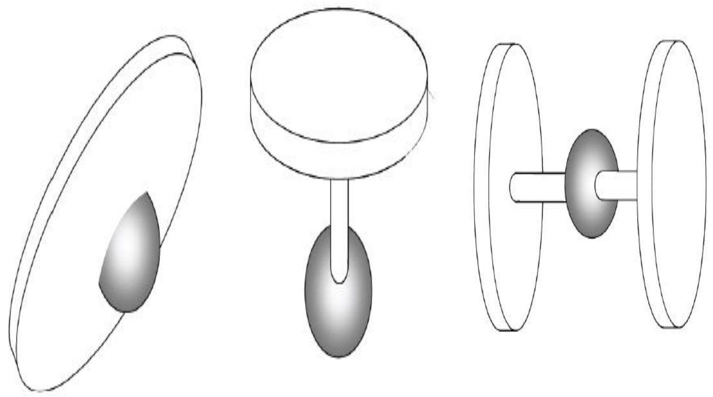

Figure 2.2: “Tabletop,” “Umbrella,” and “Dumbbell” Configurations[26]

2.3.2 System Mass Properties. Accurate knowledge of a spacecraft’s mass properties is needed for all stages of the spacecraft’s life, from design to launch to on-orbit operations to disposal [6]. The mass properties of a spacecraft are its mass, the location of its center of gravity (CG), and the mass moment of inertia (MOI) matrix which describes the distribution of the mass throughout the spacecraft as described in Eq. (2.7). For this research, the total mass of the spacecraft simulator is not a factor because air bearings are known for their ability to support large masses. However, the location of the center of gravity and MOI matrix play a key part in the spacecraft simulator’s behavior due to the rotational dyanmics involved in spacecraft reorientation.

A spacecraft simulator floating on a spherical air bearing has to have its CG at the center of rotation (CR) of the air bearing in order to simulate a torque-free enviornment [5]. If the CG is above the CR, the spacecraft is in an unstable configuration similar to an inverted pendulum which will tilt due to gravity causing a torque about the CR. If the CG is below the CR the spacecraft will tend to oscillate and will eventually come to rest with its center of mass at its lowest equilibrium point. There are two ways to attempt to locate the spacecraft’s CG at the CR of the spherical air bearing. Boynton

[6] suggests to counterbalance the satellite or split it into two halves. Ross [34] suggests actively compensating for the offset between the spacecraft CG and air bearing CR in the spacecraft's equations of motion. In this research, the adjustable counterbalance approach will be taken. A set of weights shown in Fig. 2.3 are used in conjuction with an onboard algoirthm that determines how far to move the weights to align the CG and CR. The tan weight in the center of the figure can be moved left or right to balance the spacecraft simulator on its respective axis. The X-axis and Y-axis must be balanced before every experiment on the SimSat while the Z-axis is balanced weekly as it experiences less change over time.

The mass of the spacecraft does not appear explicitly in the equations of motion for attitude control problems. However, the MOI does and has a significant impact on the dynamics of the spacecraft. A spacecraft can be designed with one MOI matrix in mind designed around the principle axis body frame with no products of inertia ion the off-diagonals can change on orbit due to structural flexing, sloshing of fuel, or other causes. Any changes to the mass distribution of the spacecraft will change the MOI. This research uses a script created by Wright for his research which runs a number of maneuvers on the spacecraft simulator and conducts post-processing to determine accurate MOI with the products of inertia included in order to use the most accurate MOI possible for attitude control [44]. The actual MOI used in this research is found in Eq. (44).

2.3.3 AFIT Simulated Spacecraft. AFIT has designed, built, and tested two spacecraft simulators whose primary purpose is attitude control simulation since 1999. The first, SimSat I, was designed using the "dumbbell" configuration as described in Sec. 2.3.1 and built in 1999 and is shown in Fig. 2.4 [10]. For the next 8 years, dynamics and control research was conducted on SimSat I [11] [17] [38]. However, due to its large mass and inertia, it could not conduct rapid slew maneuvers or achieve spin stabilization [30]. Its momentum wheels also saturated quickly and the simulator itself experienced structural deflections resulting in a preferred orientation. AFIT faculty requested that a new, more up-to-date spacecraft simulator be designed and built.

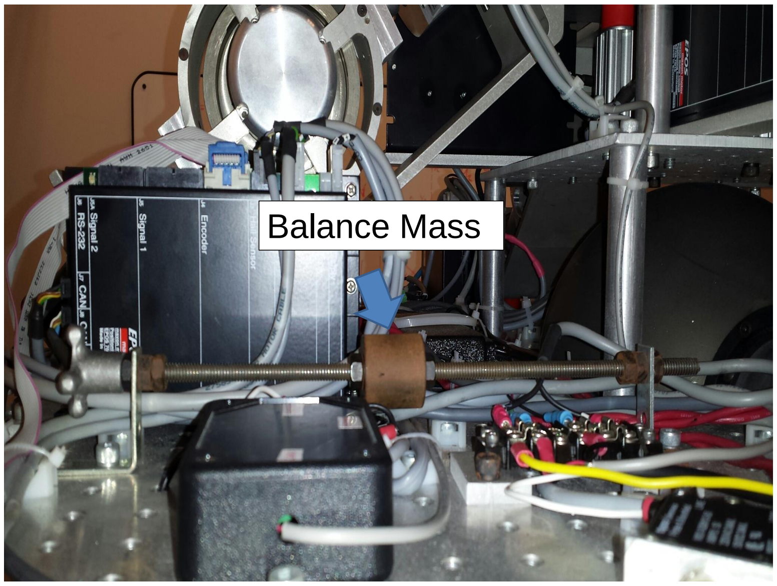

Figure 2.3: Manual Mass Balance of the SimSat

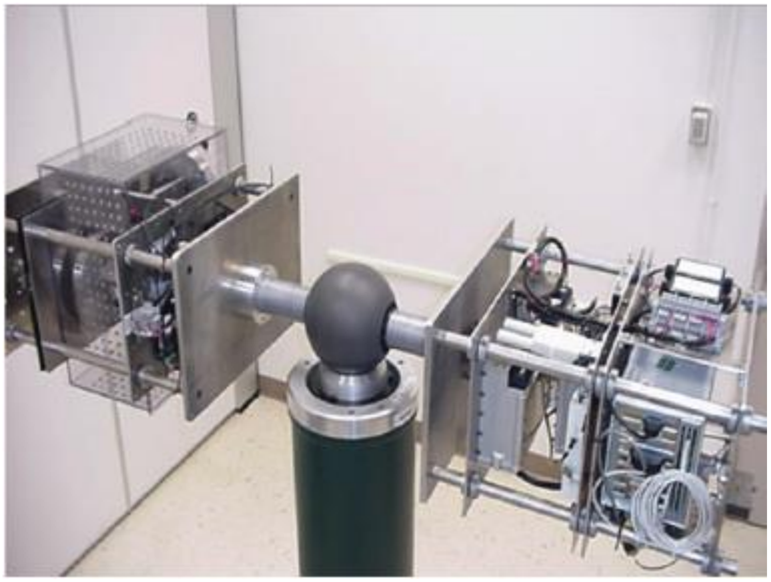

Figure 2.4: SimSat I

SimSat II, AFIT's second spacecraft simulator, was designed in 2007 and 2008 by Roach, Rohe, and Welty [30]. Since it is the most recent AFIT spacecraft simulator and the one that new research is being conducted on, the "II" was dropped and it is commonly referred to as just "SimSat." SimSat consists of a ground station which is a Windows XP-based PC that communicates with the testbed, a tri-axial air bearing which the testbed floats on, and the spacecraft simulator itself. It is a "tabletop" configuration as compared to the "dumbbell" configuration of SimSat I. The tabletop configuration allows for a full 360° degrees of freedom about the Z-axis and ±25° about the X-axis and Y-axis. SimSat has an IMU for attitude sensing and reaction wheels, control moment gyros (CMGs), and fans (to simulate thrusters on a real spacecraft) as attitude actuators. The full configuration of the spacecraft simulator will be covered in depth in Chapter III. A picture of SimSat at the time this research was conducted is provided in Fig. 1.1.

SimSat was made operational with just the fans as its only means of attitude control. Since it came online the 2nd generation SimSat has been used for seven different M.S. and Ph.D research projects involving attitude determination and control. McFarland designed and implemented a near-real time optimal control algorithm on the testbed with the fans as his actuators but was unable to transfer optimal control trajectories quickly enough from the ground station to the testbed in order to work as anticipated in 2009 [26]. His algorithm is discussed further in Sec. 3.3.3.1 because this presented research is an extension of his work. Snider developed a reaction wheel-based PID controller in 2010 [39]. McChesney designed, implemented, and tested reaction wheels and CMGs for SimSat in 2011 [25]. Padro developed a star tracker for use on SimSat in 2012 [29] which had further work on the topic performed by Grunwald in 2014 [18]. Wright investigated in-flight MOI and structural deflection algorithms in 2015 [44]. Doupe studied the optimal combination of reaction wheels and CMGs in 2015 [14].

2.4 Spacecraft Applications of Optimal Control

This section will discuss the historical uses of optimal control theory as it exists in open literature and is applied to spacecraft with an emphasis on the time-optimal reorientation problem.

2.4.1 History of Spacecraft Optimal Control. The development of numerical methods for solving optimal control aerospace problems has paralleled the exploration of space and the advancement of computing technologies [3]. Optimal control theory has been applied to spacecraft reorientation maneuvers with the objective of minimizing the time of the maneuever, the control effort required, or the structural vibrations encountered [4].

Since the 1960s, the time-optimal reorientation maneuver of a rigid body spacecraft was thought to be an eigenaxis rotation [15] as described in Section 2.1.2. This is intuitvely faster than an Euler axis rotation as the Euler axis sequence has three rotations while the eigenaxis rotation can be accomplished in just one.

In 1984, Vadali *et al.* showed that Junkins and Turner's previous work on optimal large-angle attitude maneuvers of rigid spacecraft using an integral performance index can have a closed-form solution for special cases of single-axis manuevers using Euler parameters [40]. They imposed an orthogonality constraint between the Euler parameter vector and the corresponding costate vector and proved that this is equivalent to minimizing the norm of the Euler parameter costate vector. This allowed for a closed-form solution for certain cases of the reorientation problem. The optimal control problems for the maneuvers in this research do not have a closed-form solution and must be solved using indirect methods.

Junkins and Turner published *Optimal Spacecraft Rotational Maneuvers* in 1986 as a collection of their work on manuevering modern spacecraft [22]. Their book covers both open- and closed-loop optimal control of rigid-body and flexible spacecraft but make no claims as to whether the time-optimal open-loop solutions they generate in Chapter 8 of their text are more time-optimal than an eigenaxis maneuver. They do note that open-loop optimal maneuvers are subject to disturbances and require some kind of terminal control at the end of the manuever in order to correct for errors during the reorientation. Junkins and Turner also mention work done on the NOVA spacecraft which performed open-loop time-optimal control using magnetic torques as opposed to reaction wheels. They also added that this is a much slower motion than using reaction wheels and took place over a matter of hours as opposed to seconds in this presented research.

In 1988, Wie *et al.* explored the use of a quaternion feedback regulator for eigenaxis rotational maneuvers of an asymmetric spacecraft. Their research discussed gain selection so that a quaternion feedback regulator will consistently provide a near-eigenaxis maneuver. This gain selection will also guarantee global stability of the controller using a Lyapunov function.

In 1993, Bilimora and Wie proved that the time-optimal solution for a rigid body reorientation was not the eigenaxis maneuver [4]. They refuted Li and Bainum's work which claimed that the eigenaxis rotation is time-optimal for independent three-axis control and "near" the time-optimal solution for other cases. The minimum-time rest-to-rest maneuver was found using Pontryiagin's Minimum Principle to resemble a "bang-bang" control input. When observed in the inertial frame, the rigid body experiences a geometrically complex motion that has a significant nutational component. The spacecraft is able to provide more torque along the desired inertial axis of rotation.

2.4.2 On-Orbit Maneuvers. In 2007, a team of researchers from Draper Laboratories, Rice University, and the Naval Postgraduate School (NPS) applied optimal control theory to controlling the attitude of the International Space Station (ISS) [2]. The Zero-Propellant Maneuver allowed the CMGs of the ISS to perform a full reorientation manuever without reaching rate limits by applying a control trajectory generated using pseudospectral methods. This allowed the ISS to change its attitude without having to fire its cold-gas thrusters which extends the time between refuelings.

Another example of optimal control theory applied to spacecraft in orbit was the time-optimal reorientation of a NASA spacecraft. On August 10, 2010, a NASA space telescope called TRACE executed the first ever minimum-time rotational maneuver performed in orbit [21]. Pseudospectral methods were used to generate a more efficient solution than the traditional eigenaxis maneuver. The researchers also performed a more realistic test by pointing the spacecraft in a STAR maneuver at five separate celestial targets and showed that the time-optimal control solution using pseudospectral methods was 10% more efficient than the eigenaxis solution.

2.4.3 Near Real-Time Optimal Control. What is now referred to as Real-Time Optimal Control (RTOC) was discussed by Ross *et. al* in 2006 in a conference paper discussing their application of pseudospectral feedback methods to the NPS's satellite simulator, NPSAT1 [34]. They presented a sample-and-hold technique for calculating the optimal control solution of a fixed interval by sampling the state and control measurements and feeding them through the pseudospectral software package called DIDO and then connecting the different solutions to form one piecewise trajectory. Ross *et al.* also discussed the difficulties in piecing together optimal trajectories and computational delay in generating new solutions from updated sensor data.

Ross and others examined an approach for trading computational cost with solution optimality and accuracy in what they refer to as the Bellman Pseudospectral Method [32]. This principle states that as the number of nodes used in calculating an optimal solution using pseudospectral methods increases, the computational time required to compute the solution as well as the error between the optimal solution generated using pseudospectral methods and the analytical, true solution can both increase. Ross *et al.* were able to demonstrate that using a fixed "small" set of nodes in their pseudospectral package and then implementing Bellman's principle of optimality as shown in Fig. 2.5 and recalculating the solution starting from intermediate points along the original optimal trajectory the error in the final states of the solution could be significantly reduced. This allows for calculations only requiring a few nodes which greatly decreases computational time between trajectory calculations. This research uses their method of solving the optimal control problem at different points along the spacecraft's trajectory and using fewer collocation points in the pseudospectral optimal control solver.

In 2009, McFarland's research was an attempt to further the work of Ross *et al.* by applying RTOC to SimSat [26]. He was successful in characterizing the drag model of SimSat and running open-loop optimal control and eigenaxis rotation optimal control on the spacecraft simulator but due to time constraints was forced to simulate his near real-time optimal control results using his air drag model of SimSat to simulate external disturbances to the spacecraft. His research used DIDO as an optimal solver and had only fans as a means of actuation. At this time he did not have the benefits of Wright's research on balancing the testbed and calculating a more accurate MOI with the products of inertia. McFarland only used measured body frame moments of inertia and assumed they were principal in the form of a a diagonal matrix for his state equations [44] which could have resulted in a less than optimal control solution.

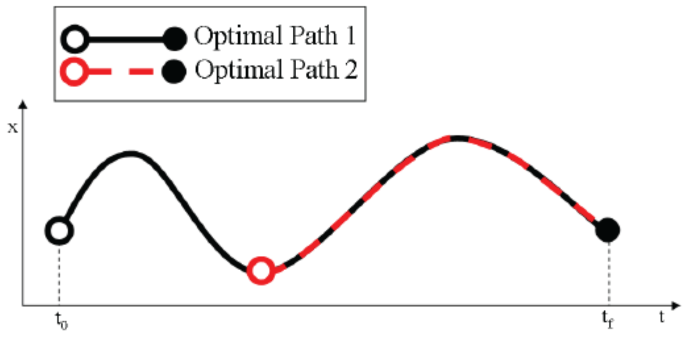

Figure 2.5: Bellman Optimality Principle

This presented research extends McFarland's work in which reaction wheels are used instead of fans and GPOPS-II instead of DIDO as the optimal control solver[26]. A near real-time controller based on McFarland's work will be implemented on SimSat. The ground station PC (faster than the one McFarland had access to) will continually compute new solutions using GPOPS-II. The on-board computer will read the new solutions and determine whether the new solution is more "optimal" than the previous one and if so switch to the new one. This controller will be compared to a PID controller and an open-loop optimal control solution with error metrics discussed in Chapters III and IV.

III. Methodology

3.1 Introduction

Chapter III presents the design, implementation, and testing of a near RTOC controller on AFIT's spacecraft simulator. The first section describes the hardware and software on SimSAT, the ground station, and how they are configured. The second section covers the different controller models used in this research. The third section discusses the design of the experiments used to evaluate the Near RTOC implementation in this research. The fourth section descibes the simulation that is compared against the hardware data. The final section is the description of error metrics used to compare the different experimental results.

3.2 Hardware and Software Configuration

SimSat is the second AFIT spacecraft attitude determination and control testbed built and tested by students as described in Sec. 2.3.3. There are three main subsytems that compose the main SimSat system:

- the SimSat spacecraft simulator
- a ground station (GS) PC
- a tri-axial air bearing.

These three components of SimSat will be discussed in depth in the next three subsections.

3.2.1 SimSat Spacecraft Simulator. The SimSat spacecraft simulator was designed, built, and tested by Roach, Rohe, and Welty between 2007 and 2008 [30]. SimSat is a tabletop configuration which came about as a result of a trade study between different hardware configurations [30] [31]. An image of SimSat is shown in Fig. 1.1.

At the time of this research SimSat has only one method of dynamic attitude determination sensors, a Northrup Grumman LN-200 Fiber Optic Gyroscope, which is a space-rated inertial measurement unit (IMU) also used on AMRAAM air-to-air missiles and Mars rovers. While there is a star dome and a star sensor currently onboard the SimSat it is insufficient for dynamic observations.

The IMU consists of three fiber optic gyroscopes and three accelerometers. The fiber optic gyroscopes generally have three pairs of long coils of fiber optic cables in which light is transmitted through them by means of a laser diode. While the gyroscope rotates with the spacecraft simulator, light traveling along the axis of rotation will experience a shorter path due to the Sagnac effect [24]. Sensors at either end of the coil will measure the effect that the shorter path has on the light, which is translated to an angular rotation rate. Since the LN-200 is aligned with the principal axes of SimSat, the angular rotation rates measured by the IMU are the same as those for the spacecraft simulator. The accelerometers are used to level SimSat prior to an experiment being run. A table of the LN-200's specifications is found in Table 1. An image of the LN-200 can be found in Fig. 6.

Table 1: Northrop Grumman LN-200 IMU [26]

Parameter	Value	Unit
Weight	700	g
Diameter	8.9	cm
Height	8.5	cm
Power Consumption	10	W
Bias Repeatability	1-10	/hr
Random Walk	0.04-0.1	$^\circ$ hr$^{\frac{1}{2}}$ power spectral density
Data Latency	<1	msec
Data Protocol	RS-485	-
Data Structure	-	Synchronous Data Link Control (SDLC)

SimSat has three attitude control subsystems: six fans powered by electric motors to simulate the effects of thrusters, three reaction wheels, and four CMGs arranged in a pyramid configuration. These actuators can be used independently or simultaneous to provide torque to the simulator.

The six orthogonal fans are composed of a Maxon EC motor model 118895 with attached Landing Products LP05050 propellers and plexiglass safety cowlings. They are the original means of attitude control when SimSat was first constructed and were used by McFarland in his research [26]. The motors are digitally controlled via a Controller

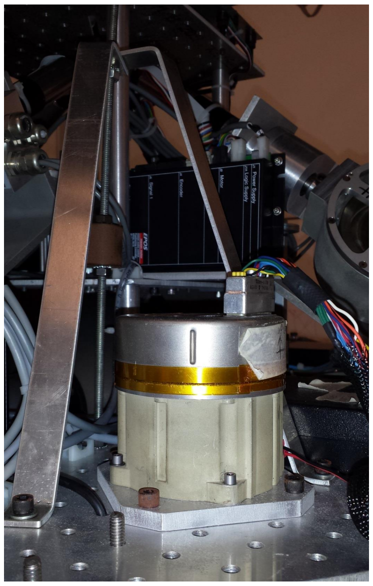

Figure 6: LN-200 Fiber Optic Gyroscope

Area Network (CAN) interface from on-board the dSPACE MicroAutoBox. Two motors operating along the principal axes of the SimSat but in opposite directions allow for full three-axis control of the spacecraft simulator. An image can be found in Fig. 7.

McFarland's research focused on using the fans as the primary means of actuation because the reaction wheels and control moment gyros were not installed at that point. This research uses the fans before and after each experiment in order to stabilize the testbed and zero the reaction wheels to 0 RPM. The reaction wheels are the only means of actuation during each test run. The control moment gyros are not used in this research.

The reaction wheels were designed, built, and tested by Snider in 2010 which were followed by McChesney's addition of control moment gyros in 2011 [25]. They took on the task of adding additional actuators to SimSat to be used by future students in attitude determination and controls research.

Snider added three reaction wheels with one on each body frame axis. They were 10.6 cm diameter solid stainless steel wheels with an MOI of $0.00261\text{kg} - \text{m}^2$. They are powered by the same Maxon EC brushless motors as the fans and are controlled by Maxon EPOS 70/10 motor control units. While the EPOS boxes support multiple modes this research uses velocity mode for the reaction wheels which uses a PI regulator to achieve

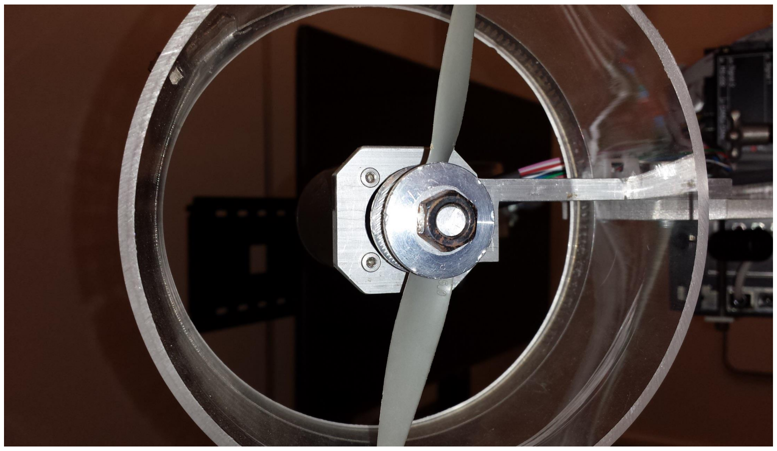

Figure 7: Fan on SimSat

and mantain the desired motor speed. The EPOS units are able to communicate with the MicroAutoBox using the CANOpen communications protocol on the CAN bus.

McChesney removed the 10.6 cm wheels as they had a poor MOI-to-mass ratio of $1.719x10^{-3}$m^2 and limited the capability of SimSat to perform large angular reorientation maneuvers. The new wheels uses spokes and have a diameter of 20.32 cm. They possess an MOI of roughly 0.008 kg-m^2 which is a significant improvement over the original design. Each wheel has a slightly different MOI but they only differ by $1 \times 10^{-4}\mathrm{kg} - \mathrm{m}^2$. An image of a 20 cm reaction wheel can be found in Fig. 8.

The reaction wheel MOIs by axis are found in Table 2. They were determined using AFIT's Space Electronics XR250 MOI measuring system [25]. The XR250 uses a torsion spring oscillator to measure the MOI about the axis of rotation.

Table 2: SimSAT Reaction Wheel Mass Moments of Inertia

Axis	MOI (kg-m^2)
X	0.0079119
Y	0.0079304
Z	0.0079204

.

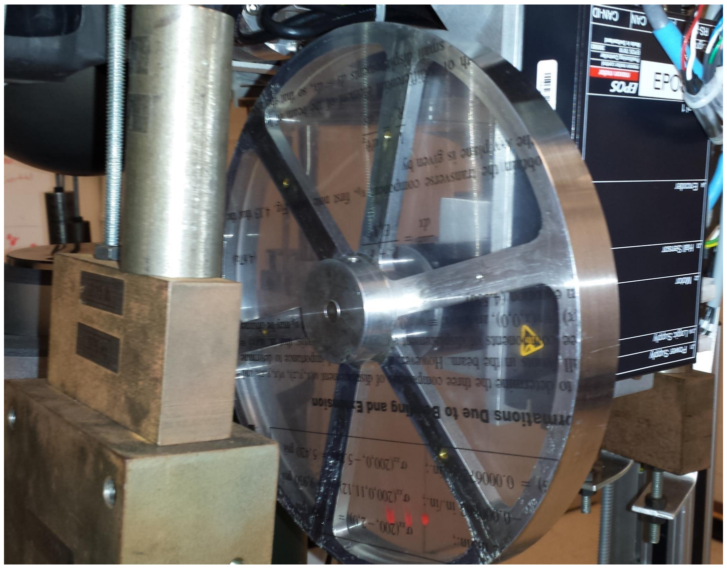

Figure 8: 20 cm Reaction Wheel Mounted on SimSat

All sensors and actuators with the exception of the star sensor are connected to a dSPACE MicroAutoBox. This is a programmable real-time microcontroller that executes programs found on the on-board computer that have been precompiled in the Simulink interface. It is the link between the MATLAB interface on the on-board computer and the motors powering the actuators on SimSat. The MicroAutoBox provides the interfaces to send and receive data from the sensors and actuators and the on-board PC. An image of the MicroAutoBox is found in Fig. 9.

The on-board computer is a Mini-Box PC. It has three purproses. The first is to provide a software environment for creating and editing the Simulink models and then compiling them to real-time programs. These real-time programs are compiled on the on-board computer which is remotely controlled by the GS PC. The second is to interface with the MicroAutoBox during real-time operations and tests using dSPACE ControlDesk or the mLib MATLAB interface. The third is to provide a wireless link between the Ground Station PC and SimSat using a 802.11b/g interface. This is done with the Windows XP Remote Desktop interface or with MATLAB UDP or TCP/IP commands.This research uses UDP instead of TCP/IP due to the need for real-time communications. The increased accuracy of TCP/IP communication is not as important for Near RTOC as than the increased speed of UDP. This PC-to-PC interface is discussed

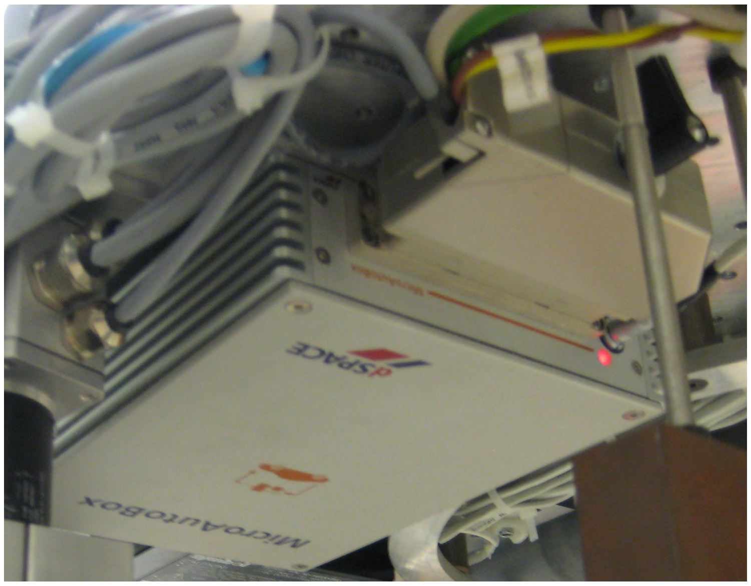

Figure 9: dSPACE MicroAutoBox

in depth in Sec. 3.3.3.1. The on-board computer runs MATLAB 7.0.4. and 2009 but only 7.0.4 is used in this research. It is shown in Fig. 10 and its specifications are in Table 4.

Table 3: SimSAT II Mini-box PC Components

Parameter	Value
Processor	x86 1500MHz
Motherboard	Jetway Hybrid MicroATX
Memory	1024 MB
Storage	40 GB
Wireless	Linksys Compact Wireless USB Network Adapter

SimSat has well defined mass properties explored by Wright in his research [44]. The values for the mass moment of inertia tensor in the body frame derived from Eq. (2.7) used in this research is

$$I_b = \left\{ \begin{array}{ccc} 6.454 & -0.197 & -0.175 \\ -0.142 & 9.716 & -0.197 \\ -0.175 & -0.142 & 12.848 \end{array} \right\} kg - m^2. \qquad (44)$$

.

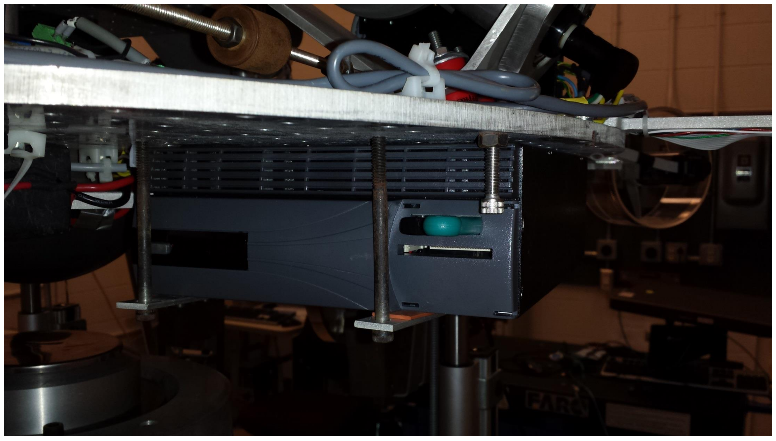

Figure 10: On-Board PC

3.2.2 Ground Station PC. The ground station consists of a custom-built PC running Windows XP Professional Service Pack 2 and MATLAB 2013b. It has an Intel Core i7 920 quad-core processor running at 2.67 GHz and 4 GB of RAM, but only 2 GB are usable by Windows XP due to the limitations of 32-bit operating systems. The full specifications of the GS PC are found in Table 4. It has a wired connection to a Linksys WRT54G wireless router. This allows for bidirectional communication between the GS and the on-board computer. This PC is not the original configuration designed for use with SimSat by Roach *et. al*; it was upgraded during McFarland's research in order to provide more processing capaiblities for resource-intensive optimal control software such as DIDO or GPOPS which is important to this presented research. Possible upgrades to the GS PC are found in Sec. 5.2.1.

Table 4: SimSAT II Ground Station PC Components

Parameter	Value
Processor	Intel Core i7 2.66GHz Quad-Core Processor
Motherboard	BIOSTAR Intel X58 ATX Motherboard
Memory	4GB DDR3 SDRAM (2GB Unused)
Storage	240 GB
Video	Radeon x850 Series
Wireless	Linksys Wireless-G Broadband Router

The GS PC serves two purposes in this research. The first is to provide the ability to control the on-board computer on SimSat without having a display cable connected to a monitor. This is crucial as any extraneous cables attached to the testbed will not only prevent the spacecraft simulator from being able to rotate its full range of motion but also create additional torques that will cause deviations in SimSat's trajectory. This capability is provided using the Remote Desktop Connection feature in Windows XP. A user can log onto the on-board PC from the GS and use the same keyboard, mouse, and monitor to control both computers.

The second purpose is to compute the optimal control solutions and transfer them to the on-board computer where they will be computationally tranlated into commands for the actuators on SimSat. The on-board computer is less powerful overall than the GS PC. The solution is to have a MATLAB instance on the GS connected via User Datagram Protocol (UDP) to a MATLAB instance on the on-board computer. The on-board computer sends state information to the GS and receives updated trajectory and control arrays from the GS. The logic of this method of near RTOC is discussed in depth in Sec. 3.3.3.1. The GS PC runs MATLAB 2013b and GPOPS-II which is depicted in Fig. 11.

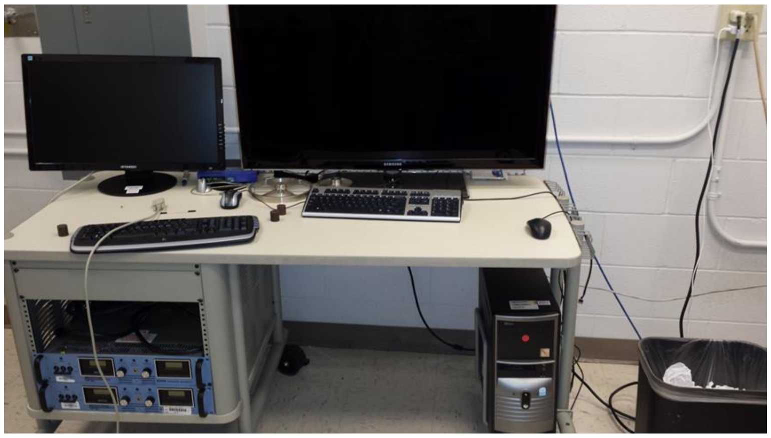

Figure 11: Ground Station PC

3.2.3 Tri-Axial Air Bearing. The air bearing used by SimSat is the Space Electronics, Inc. Model SE-9791 Tri-Axis Spherical air bearing. An image is shown in Fig. 12. It provides a friction-free enviornment for the ball bearing and has three DOF rotational motion. This allows SimSat to rotate in a near torque-free environment similar to a spacecraft on orbit. The X- and Y- axes are limited to ±30° rotation and the Z-axis has unlimited rotation. Its measureables are shown in Table 5.

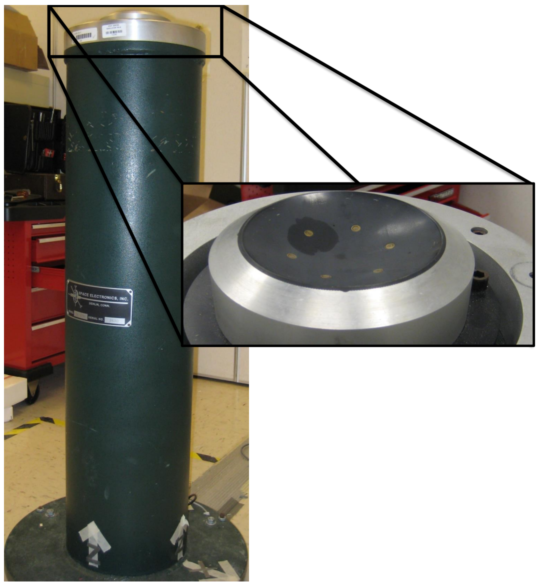

Figure 12: Tri-Axial Air Bearing

Table 5: Space Electronics, Inc. Tri-Axis Air Bearing

Parameter	Value	Unit
Ball Bearing Diameter	22.00	cm
Pedestal Cup Diameter	5.72	cm
Unloaded Ball Bearing Mass	19.05	kg
Maximum Loaded Ball Bearing Mass	136.08	kg
1-Axis Max Rotation Angle	±25	deg
2-Axis Max Rotation Angle	±25	deg
3-Axis Max Rotation Angle	-	-

3.3 Controller Models

The SimSat simulated satellite uses real-time Simulink control blocks to interact with the different hardware interfaces. The model is compiled and then executed on the dSPACE MicroAutoBox. The MicroAutoBox is the control hub of SimSat, as discussed in Sec. 3.2.1, and directly controls the sensors and actuators on-board. The Simulink model is found in Fig. 13.

For all three controllers tested, the SimSat's Simulink model:

- Receives Measured Angular Rotation Rates from LN-200 IMU,
- Converts Measured Angular Rotation Rates to Quaternions
- Commands RPM Inputs to the Fans,
- Commands RPM Inputs to the Reaction Wheels,
- Receives and Executes External Control Commands,
- Performs Closed Loop Control Calculations (PID controller only).
- Executes Open Loop Control Trajectory (Open Loop Optimal Control and Near RTOC Only)

The Simulink model connects to the MATLAB command line interface through the mLib interface allowing the model to directly interface with the sensor and actuator hardware. This interface is crucial for the near RTOC implementation as well as data collection at the end of each experiment.

For every experiment, two MATLAB files must be run. The first is a script named *SlewScript.m* which declares the target orientation, maximum time for the experiment to run, and controller type to use. For optimal control runs, the initial $\dot{\vec{h}}_{rwa}$ and time series are loaded here. All of these values are passed to the second file, a MATLAB function named *slewRW.m*. This file interfaces directly with the Simulink model loaded into the dSPACE ControlDesk software which interfaces with the dSPACE MicroAutoBox.

3.3.1 PID Controller.

For PID control of the SimSat the Simulink model computes the quaternion error between the desired quaternion and the current quaternion from which PID is computed and sent to the actuators to drive the error to 0.

The quaternion error is calculated using Eq. (2.19). The quaternion error's derivative and integral are calculated numerically. The error, integral of the error, and the

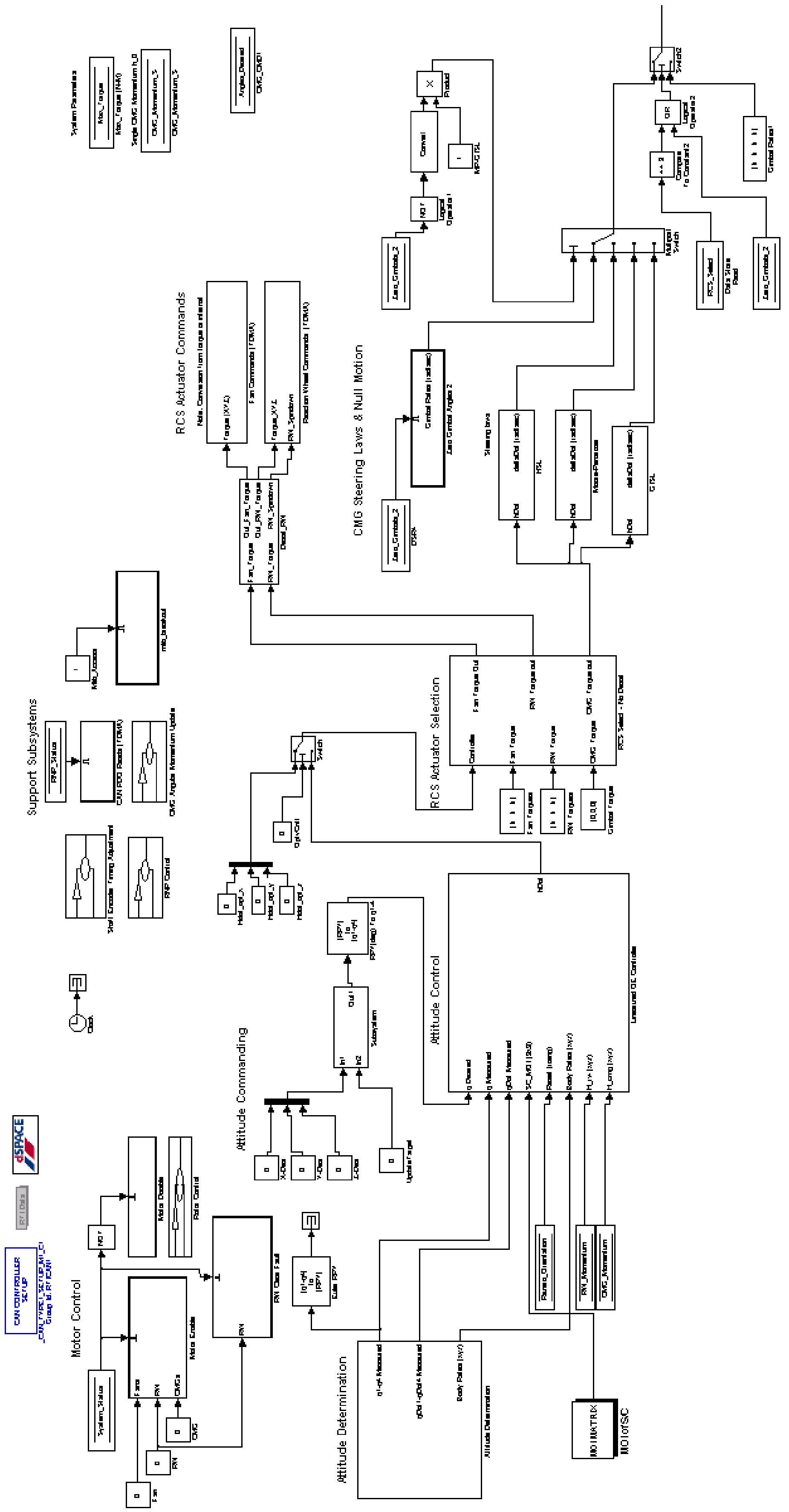

Figure 13: Hardware Simulink Model

derivative of the error are mulitplied by gain matrices calculated using a series of equations found in Wie's *Space Vehicle Dynamics and Control* [42]. The desired natural frequency and percent overshoot are both 0.707 which was found to produce a time-optimal response from simulation. The gain matrices for each are

Table 6: Gain Matrices for PID Control

Axis	K_P	K_I	K_D
1	3.4786	0.1564	22.3586
2	5.2368	0.2356	33.6591
3	6.9249	0.3116	44.5093

.

A nonlinear correction term $\omega^x(I\vec{\omega} + h_{rwa})$ is subtracted in order to account for the nonlinear term in the equations of motion. The full feedback term for the control $\vec{u}$ is found to be

$$\vec{u} = \dot{\vec{h}}_{rwa} = K_P I\vec{q}_{error} + K_D I\vec{\omega} + K_I \Delta t\vec{q}_{error} - \omega^x(I\vec{\omega} + h_{rwa})$$

This control vector is commanded to the reaction wheels and the spacecraft simulator will turn to the desired orientation. PID control is advantageous in that it has a long history of success in aerospace applications and with the proper gains will reach the desired target but it can produce an undesired overshoot of the target orientation and for a Z-axis slew will only use the reaction wheel aligned with that axis [42].

For this research, a fixed time is set for the maximum time that the experiment will run for. The PID controller will drive the spacecraft simulator to the desired orientation and hold that orientation unless the maximum time is reached.

PID testing is accomplished via a script interface in MATLAB and ControlDesk software which interacts directly with the hardware. The dSPACE mLib package is used to write a trigger directly to the Simulink model. This trigger will activate the Simulink real-time model that interfaces with ControlDesk and will level the spacecraft simulator using the accelerometer on the IMU, zero the reaction wheel speeds if necessary, and begin the experiment. SimSat will use the PID controller with the gains in Table 6 in the Simulink model to reorient itself from its initial orientation to its final orientation using only the reaction wheels. Once the experiment reaches its final time as specified in the MATLAB script, the test run will cease and state data throughout the run sampled at 100 Hz will be saved.

3.3.2 Open Loop Optimal Control. Open-loop optimal control is a control history calculated ahead of time using GPOPSII and fed-forward directly to the reaction wheels with no closed-loop feedback present. The control algorithm and MATLAB files used in this control scheme are similar to those of Near RTOC but are calculated once before the experiment is run instead of continually throughout the maneuver. The development of the equations and values used in the general open-loop optimal control solution generated using GPOPSII will be developed in this section while specifics to the near real-time application of this solution are developed in Sec. 3.3.1.

GPOPSII requires three MATLAB files to generate an optimal control solution: a main script, a function file containing the state equations and their relation to one another, and a function file containing endpoint conditions and the cost function. The costate, stationary, and stationary equations do not need to be defined explicitly. The two function files were kept constant for all optimal control calculations while the main file was changed.

In the main file, the initial state is

$$\begin{bmatrix} q_{1,initial} \\ q_{2,initial} \\ q_{3,initial} \\ q_{4,initial} \\ \omega_{1,initial} \\ \omega_{2,initial} \\ \omega_{3,initial} \\ h_{1,initial} \\ h_{2,initial} \\ h_{3,initial} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{46}$$

This represents an initial orientation of Euler angles $0°, 0°, 0°$, body rates in each axis of 0 rad/s, and initial angular momentum produced by the reaction wheels of 0 $kgms^{-1}$. The initial control vector is set to

$$\vec{u}_{initial} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{47}$$

The final state is set to the same as the initial orientation for the body rates $\omega_1, \omega_2, \omega_3$ and angular momentum h_1, h_2, h_3. The final quaternion parameters are set to the quaternion which represents the desired Euler angle orientation of the spacecraft simulator. The final control vector is set to be 0 as well. In vector form this is

$$\begin{bmatrix} q_{1,final} \\ q_{2,final} \\ q_{3,final} \\ q_{4,final} \\ \omega_{1,final} \\ \omega_{2,final} \\ \omega_{3,final} \\ h_{1,final} \\ h_{2,final} \\ h_{3,final} \end{bmatrix} = \begin{bmatrix} q_{1,desired} \\ q_{2,desired} \\ q_{3,desired} \\ q_{4,desired} \\ 0 \\ 0 \\ 0 \\ \text{free} \\ \text{free} \\ \text{free} \end{bmatrix} \tag{48}$$

and

$$\vec{u}_{final} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{49}$$

Another required input for GPOPSII is the upper and lower limits for each state variable and control variable. The four parameters that make up the quaternion, q_1, q_2, q_3. and q_4 are able to vary from -1 to 1. The body angular rates for each axis are allowed to vary from -0.5 to 0.5 rad/sec. This limit was derived from Doupe's experimental results where he found through stressing SimSat that it would not rotate in any axis more than 30 deg/s or 0.5 rad/s [14]. The limits on angular momentum produced by the reaction wheels is set to be from -2.1962 to 2.1962 $kg - m^2$. These angular momentum limits are calculated from the max reaction wheel velocities. The limits on each control is from $-.25$ to $.25$ $\text{kg} - \text{m}^2 - \text{s}^{-1}$ which is also derived from Doupe's experiments [14].

Also in the main file are the initial guesses for the state and control histories which are set as the desired initial and final states in matrix form due to the lack of knowledge of the trajectories. These guesses are used as initial conditions in the two-point boundary value problem solved by the optimal control solver.

The final settings are the options for the GPOPS-II pseudospectral solver. The options selected for open loop optimal control are different than those for near RTOC due to the lack of real-time processing requirement. Open loop optimal control trajectories can be calculated ahead of time, saved, and then fed-forward on-board the SimSat. Near RTOC trajectories are calculated during the reorientation maneuver where the time required to calculate the optimal solutions is a factor.

The three settings that make the most impact on the accuracy of the solution as well as the time required for a PC to solve the optimal control problem are the number of mesh iterations, mesh tolerance, and number of collocation points on the mesh. These settings affect the mesh that is used by GPOPS-II to solve the pseudospectral problem presented in Sec. 2.2.5. A small mesh tolerance such as 10^{-9} and a large number of collocation points such as 1,000 or 5,000 and a high number of iterations such as 100 or 200 result in a control solution that is accurate throughout the trajectory but takes a great deal of computational power. Such a solution may take minutes or even hours to compute. The reverse is also true. A large tolerance such as 10^{-3} but fewer collocation points such as 10 or 20 and fewer mesh iterations such as 20 or 30 result in a less accurate but quickly computed solution.

For this research, an open loop optimal control solution was computed using the following settings: a mesh tolerance of 10^{-6}, mesh maximum iterations of 70, and $10,000$ collocation points. These settings result in a computational time on the order of magnitude of 2 to 5 minutes.

The two function files needed for GPOPS-II are the "continuous" function which contains state information and the "endpoint" function which contains the performance index. The continuous file requires the state equations of the dynamic system in first-order differential form. The state equations were found through the expansion of Eq. (2.20) and Eq. (2.23). The time derivatives of the three angular momentum components from the reaction wheels are the three control vectors respectively. The endpoint function has the cost function from Eq. (2.43).

On the on-board PC, the optimal control history is loaded into the MATLAB workspace. A loop is created where at each timestep of 0.01 seconds a control value for each reaction wheel is interpolated from the control history generated using GPOPS-II. This is crucial as the optimal control solver does not produce a control history linearly sampled over the time required for the manuever. It has a cluster of points near the start of the maneuver, a cluster near the end, and a few points in the middle. This is a result of the two-point boundary value problem that makes up an optimal control problem and the pseudospectral methods used to solve it. The time history of the control trajectory and the sampling rate of SimSat's control system may not match up. A lookup table must be used to generate the correct $\dot{h}$ value at each timestep.

There is no feedback from the sensors to the actuators. The only purpose of the LN-200 is to collect data for later analysis. Once the control history has concluded and there are no more optimal control values the PID controller is turned on in order to hold the SimSat at its target orientation.

3.3.3 Near RTOC.

First we will discuss the near RTOC algorithm. We will discuss the decision logic used to maintain on the current optimal control trajectory or query the GS PC for an updated control trajectory. There are two loops of the Near RTOC implementation, the "outer" loop which brings SimSat from its initial conditions to its final conditions and the "inner" loop which continually recomputes optimal control solutions. After describing the algorithm, we will describe the specific implementation of that algorithm on the SimSat on-board PC and the GS PC.

3.3.3.1 Near RTOC Algorithm.

The Near RTOC algorithm used by McFarland for his research is shown below in Fig. 14 [26].

This image will be described from left to right as the algorithm, initially developed by Ross *et. al* and implementation in simulation by McFarland, has been only slightly modified for this research [34]. The optimal control solver begins computing optimal control solutions based on the initial conditions and feeds an initial optimal control trajectory to the plant. The plant starts reorienting on the received optimal trajectory and sends state information back to the optimal solver at a predetermined sampling rate. After the optimal control solver computes a solution, the solver sends the solution to the plant and immediately begins computing a new solution based on the state information

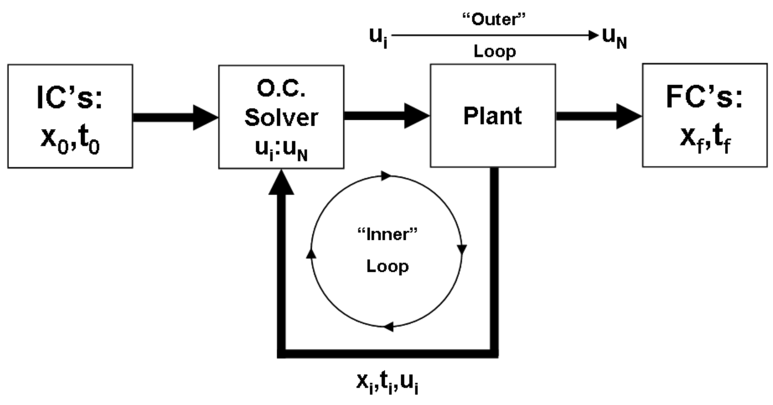

Figure 14: McFarland Near RTOC Algorithm

it most recently received from the plant. The plant receives the new optimal control solution from the optimal control solver and determines whether to stay on the trajectory it is currently on or to switch to the new trajectory. These three actions make up the "inner" loop of Near RTOC. The "outer" loop tracks how close the plant is to the final conditions and determines when to terminate control actuation when the final conditions are met.

This research takes this Near RTOC algorithm and modifies it to work on the hardware and software of SimSat. A more specific version of this algorithm used in this research is shown in Fig. 15.

The inner and outer loops are more specifically defined. Both PCs are operating on the same rate of 100 Hz. The largest difference comes in the transfer of optimal control trajectories and solutions in the "inner" loop of the Near RTOC algorithm. In McFarland's research, the orientation trajectory and control history are transferred to the plant each time they are computed in the optimal control solver. In this research, they are transferred only when the error between the actual trajectory and the optimal trajectory is larger than a set threshold value. When that occurs, the plant reads the most recently computed optimal control trajectory and control history from the optimal control solver and implements them directly in that timestep's actuation. This is a

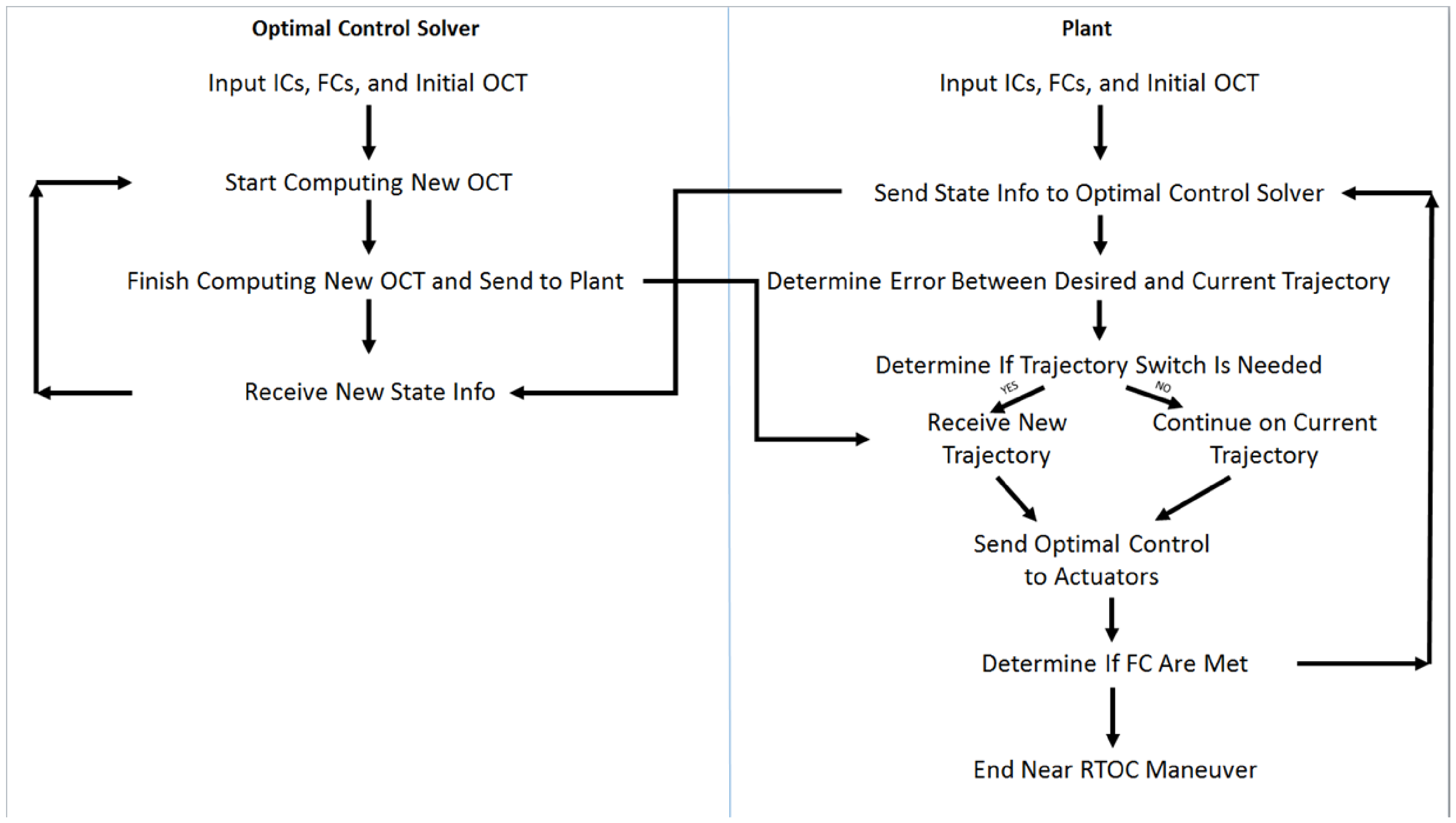
Optimal Control Solver
Input ICs, FCs, and Initial OCT
Start Computing New OCT
Finish Computing New OCT and Send to Plant
Receive New State Info
Plant
Input ICs, FCs, and Initial OCT
Send State Info to Optimal Control Solver
Determine Error Between Desired and Current Trajectory
Determine If Trajectory Switch Is Needed
YES
NO
Receive New Trajectory
Continue on Current Trajectory
Send Optimal Control to Actuators
Determine If FC Are Met
End Near RTOC Maneuver

Figure 15: Patrick Near RTOC Algorithm

necessity with the use of UDP communication as described in Sec. 3.2.1. UDP packets can be lost in transmission and not received at the plant. To limit the chance for a packet loss, as few packets as possible are transmitted over the wireless link between the plant and optimal control solver. UDP is discussed in depth in the next section.

3.3.3.2 Near RTOC Implementation. This implementation of Near RTOC requires two computers: the on-board PC on SimSat and the GS PC. The algorithm described in the above subsubsection is spread across the two computers and requires both to be operating correctly to function. The combination of the two computers' functions on one PC is something suggested in Sec. 5.2 as future work.

The Simulink model and MATLAB script for the Near RTOC implementation is the same as the one used in PID control and Open Loop Optimal Control described in Sec. 3.3.1 and Sec. 3.3.2 respectively. The difference between Near RTOC and Open Loop Optimal Control is the updated optimal control trajectory written by mLib to the Simulink model. Instead of one optimal trajectory that was solved at an earlier time and that does not change, the optimal trajectory and control history change over the course of the experiment. The function that directly interfaces with the hardware is changed to reflect the Near RTOC algorithm.

The current state of SimSat and updated quaternion time history and control time history in the form of $\dot{\vec{h}}_{rwa}$, the change in angular momentum of the reaction wheels, are transferred between the GS PC and on-board PC using UDP. In MATLAB, UDP is implemented via the Instrument Control Toolbox. Binary data in a variety of types to include as int8, int16, single, and double can be transmitted and received between computers on the same Local Area Network (LAN) running MATLAB. On each PC, a UDP object is created and then written to or read from, depending on the need.

In this research each data type is only transmitted and received unidirectionally. The on-board PC on SimSat transmits current state data:time, quaternion,body rates, reaction wheel angular rates, and change in reaction wheel angular momentum. The on-board PC receives the quaternion trajectory and control history with an associated time series from the GS PC. The GS PC transmits the quaternion trajectory, control history, and associated time series and receives the current state information. Each port and associated UDP object is only used for one data type. This lessens the chance that data intended for one purposes, such as an updated body rate to be plugged into the optimal control solver, is read into the wrong variable on the receiving computer.

A UDP object used as a trigger is created. Its purpose is to delay the optimal control solution computation on the GS PC until the on-board computer on SimSat has completed its preparation to begin the experiment so that the two computers are synced. This is necessary due to the increased speed of the GS PC, which has a much faster processor and more RAM than the on-board computer. This object is created before any of the other pre-experiment commands are run. The GS PC gets ready to start its part of the "inner" loop of the Near RTOC algorithm by declaring all of the necessary variable for GPOPS-II and waits for the on-board computer to begin its control loop. The on-board PC sends a Boolean value of "1" to the GS PC using this UDP object and the GS PC's script starts calculating its first optimal control solution.

The UDP ports are selected as to not interfere with any other Windows services or processes such as Windows Remote Desktop Manager that are also required for this research. To ensure that there are no conflicts, each UDP port number was tested separately to transfer data from one PC to another while the Remote Desktop Manager feature was active. The ports chosen do not impede any needed Windows process or service from running. The list of ports is shownin Table 7 with GS in the PC column referring to the GS PC and OB referring to SimSat's on-board computer. Different ports numbers are used for the same data type on the sending and receiving PCs to minimize the chance that data is sent to an incorrect port.

On the on-board PC, an error value is set for the "inner" loop of the Near RTOC algorithm. This is the threshold for the algorithm to determine whether to stay on the current optimal trajectory or to request and switch to a new optimal trajectory calculated on the GS PC. If $q_{error,4}$ calculated using Eq. 2.19 is more than this value, the algorithm will switch trajectories. Otherwise, SimSat will remain on the trajectory it is already on. This will be discussed further later in this section. This value will be changed for each test run in order to determine the time-optimal and most accurate value for Near

Table 7: List of UDP Ports and Functions

PC	Port	Function	Send/Receive
GS	9101	Quaternions	Receive
GS	9091	Body Angular Rate	Receive
GS	9092	Reaction Wheel Angular Momentum	Receive
GS	9093	Reaction Wheel Angular Momentum Change	Receive
GS	9094	Time	Receive
GS	10000	$\dot{\vec{h}}_1$ Trajectory	Send
GS	10001	$\dot{\vec{h}}_2$ Trajectory	Send
GS	10002	$\dot{\vec{h}}_3$ Trajectory	Send
GS	10003	q_1 Trajectory	Send
GS	10004	q_2 Trajectory	Send
GS	10005	q_3 Trajectory	Send
GS	10006	q_4 Trajectory	Send
GS	10100	Time History	Send
GS	11119	Optimal Control Computation Trigger	Receive
OB	9096	Quaternions	Send
OB	9097	Body Angular Rate	Send
OB	9098	Reaction Wheel Angular Momentum	Send
OB	9099	Reaction Wheel Angular Momentum Change	Send
OB	9100	Time	Send
OB	10013	$\dot{\vec{h}}_1$ Trajectory	Receive
OB	10014	$\dot{\vec{h}}_2$ Trajectory	Receive
OB	10015	$\dot{\vec{h}}_3$ Trajectory	Receive
OB	10016	q_1 Trajectory	Receive
OB	10017	q_2 Trajectory	Receive
OB	10018	q_3 Trajectory	Receive
OB	10019	q_4 Trajectory	Receive
OB	10101	Time History	Receive
OB	12000	Optimal Control Computation Trigger	Send

RTOC. The three values tested are 0.05, 0.1, and 0.5 which were generated from initial runs of the Near RTOC implementation on SimSat. Values smaller than 0.05 caused the spacecraft simulator to collide with the tri-axial air bearing and values larger than 0.5 did not switch optimal trajectories until SimSat was well off of its current optimal trajectory.

A timeout value is also set to 0.1 seconds. This value dictates the amount of time that the PC will wait for new data to be written to the UDP object. A smaller timeout value will cause less interruption of the algorithm but may result in no data being transferred at all between the two computers. The case where data is not received by one of the computers is discussed later in this section.

The UDP objects are created using the "udp" command. They are set to be single precision data type with the exception of the Boolean trigger for optimal control computation. This is accurate enough for the binary data transferred between the two PCs but does not create arrays with a size greater than MATLAB's UDP buffer limit of 8192 bytes. The state information sent by the on-board PC are only one capture of the state variables; with the quaternion parameters, body angular rates, and reaction wheel angular momentum having size 1x4, 1x3, and 1x3 respectively. The time sent by the on-board PC is a single precision value. On the GS PC, the collocation points for quaternion parameters, $\dot{\vec{h}}$, and the associated time series are linearly interpolated onto a time series of 100 values between the start of the optimal trajectory and the end time of the optimal trajectory.

On both the on-board PC and the GS PC an initial optimal trajectory calculated before the experiment is loaded. This trajectory contains a timeseries, the planned quaternion trajectory for SimSat to follow, and the $\dot{\vec{h}}$ control history needed to achieve that trajectory.

On the GS PC, the GPOPS-II settings and parameters discussed in Sec. 3.3.2 are declared. The values for mesh tolerance, number of iterations, and number of collocation points are set to 1×10^{-4}, 10, and 20 respectively. These values result in GPOPS-II generating an optimal control solution in roughly 10 seconds. Variation of these parameters and their effects on the peformance metrics of SimSat are discussed further in Sec. 4.4.3.

All UDP ports are opened and the GS PC waits for the on-board PC to finish leveling using the fans and despinning its reaction wheels. Once that occurs the trigger to begin calculating optimal control solutions along with the state data read using the mLib library from the first timestep are sent from the on-board PC to to the GS PC via the UDP objects previously declared. The GS PC begins calculating the first optimal control solution using GPOPS-II with the initial conditions being the state and time information sent from the on-board PC and the final conditions being the desired final orientation of the spacecraft simulator. On the on-board PC, the first $\dot{\vec{h}}$ command is sent to the reaction wheels via the Simulink model after being generated from a lookup table of the optimal control history using MATLAB's *interp1* command which linearly interpolates the correct control input for the clock value of the system from the optimal control history and timeseries in order to fit the data within the UDP data buffer limit of 8192 bytes.

For the state and time data being read on the GS PC, as well as the optimal control data that is read on the on-board PC and discussed later in this section, an algorithm is used to determine whether the data received is a new valid data point or points or empty data written in error. This is shown in Fig. 16. The MATLAB command "fread" reads the data from the UDP object created on the other PC on the network. The first check is to see if any data at all was received. If no data was received by the computer, the control model uses the most recently received data that has been checked and moves on to reading the next data type. The second check is to see if the 2-norm of the data received is equal to 0. Due to noise in the system and other factors no value received via the dSPACE MicroAutoBox, no matter how small, will have a 2-norm equal to 0. If that is the case, an empty matrix was written to the UDP object instead of the data intended. In this case, the control model uses the previously received value. These checks on the integrity of the UDP data received prevent unnecessary errors in transmission as one or both scripts will stop if an empty matrix is used instead of a state or optimal control data array. By using the previous value, despite the fact that it may not be the most accurate value needed, the experiment can continue.

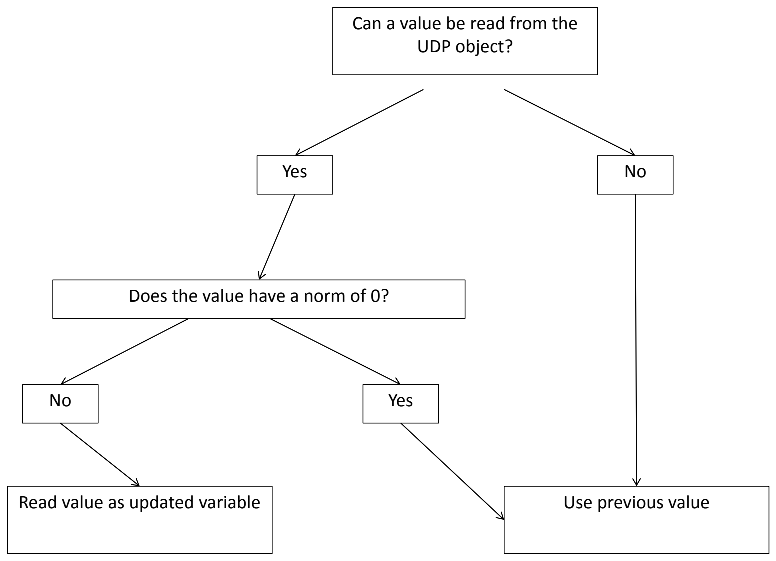
Can a value be read from the UDP object?
Yes
No
Does the value have a norm of 0?
No
Yes
Read value as updated variable
Use previous value

Figure 16: UDP Read Logic

After the $\dot{\vec{h}}$ values are written to the Simulink model and executed as reaction wheel angular accelerations, the error in orientation from the planned optimal trajectory using Eq. (2.19). The first three values of the error quaternion give knowledge of the error about each axis but do not give an overall knowledge of the total error of the orientation of the spacecraft simulator. The fourth quaternion parameter is better indicator of the total error between the desired quaternion from the optimal trajectory and the true quaternion calculated by integrating over the body angular rate of the spacecraft simulator due the relationship found in Eq. 2.18.

This error value, $q_{4\ error}$, is checked against the error threshold value discussed earlier in this section on the on-board PC. If the error quaternion parameter is smaller than the error threshold value, SimSat continues on the current optimal trajectory. If the error quaternion parameter is greater than the error threshold value, a new optimal control trajectory is requested from the GS PC. This consists of a new quaternion trajectory, new $\dot{\vec{h}}$ history, and new timeseries for the two values. The GS PC uses the initial, precalculated trajectory for its initial write to the associated UDP objects but updates the UDP objects over time with trajectories calculated using updated state and time information. Each quaternion and control array is separate to reduce reading errors on the on-board PC. The arrays are 1 by 50 in size in order to not break the UDP buffer limit of 8192 bytes for single precision data. The UDP read error checks discussed previously in this section are used on the on-board PC to determine whether the trajectory received is a valid one. If they are not met, the previous trajectory is used instead. If the checks are met, the previous trajectory is discarded and the new trajectory becomes the "old" one for comparison purposes.

The "outer" loop tracks whether the spacecraft has reached its final desired state or not. Once it has come within 1% of the target quaternion, the GS PC stops calculating new optimal control solution and the on-board PC turns on the PID controller is turned on to bring the spacecraft to its desired orientation if is not already there. This prevents collisions but that data is not used. At the conclusion of the experiment, the data is saved for analysis in the same manner as PID and Open Loop Optimal Control.

3.4 Experiment Design

Two different reorientation maneuvers were selected for this research: a 90 degree slew about the Z axis and a 180 degree slew about the Z-axis. These two maneuvers were chosen in order to study the performance of near RTOC on a reorientation slew maneuver. The PID controller will only primarily use one reaction wheel on the Z-axis to bring SimSat from its initial orientation to its final one. Optimal control methods will allow SimSat to tilt and take advantage of reaction wheels on the X and Y axes to generate additional torque that is predicted to result in for a faster orientation.

These two maneuvers will be performed by SimSat and compared to simulated data of which the generation of is discussed in the next section, Sec. 3.5. The PID data will be compared to the simulated PID data. The open-loop optimal control simulated data will be compared to the data generated by Near RTOC and Open-Loop Optimal Control test runs since there is no feasible way to simulate Near RTOC given the current SimSat configuration.

Prior to each experiment, SimSat will be manually balanced on each axis. A script created by Wright for his research will be run [44]. It calculates the imbalance of each axis and should limit errors caused by disturbance torques created by the center of gravity not being located at the center of rotation as described in Sec. 2.3.2.

3.5 Simulation

For this research, a software simulation model of the SimSat hardware was created in order to have a series of expected results that can be used to validate data collected in the experiments. This validates the state equations used in the optimal control solver GPOPS-II and gives a validation of the data generated by the hardware during tests. Simulated data for both a PID-controlled reorientation and an optimal control reorientation are generated.

The Simulink diagram for the open loop case is shown in Fig 17.

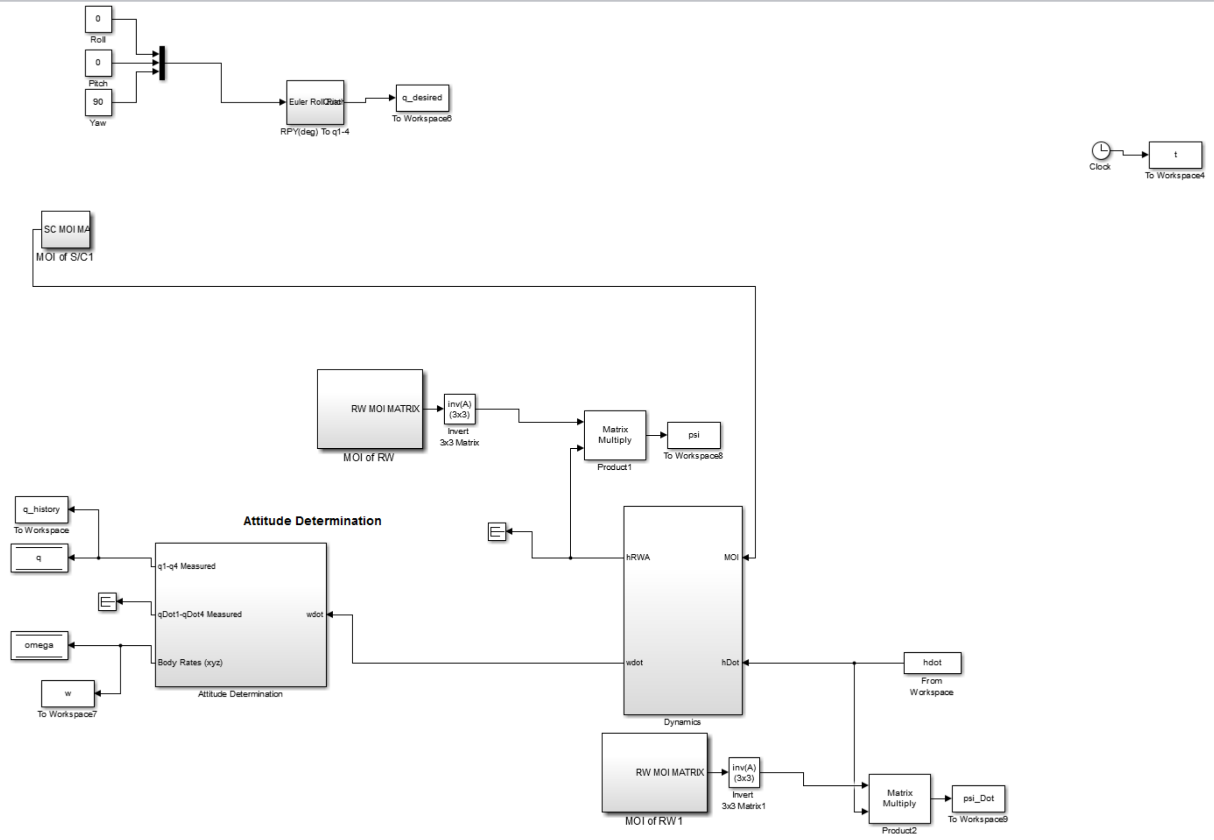

Figure 17: Open Loop Simulation Simulink

It appears similar to the on-board controller Simulink diagram because it is based on the same equations of motion. These are the same equations of motion used by GPOPS-II. The timestep is set to be fixed at 0.01 seconds, the same as the SimSat Simulimk model.

For a PID simulated run, the same gains are used as the hardware model implemented on SimSat. Initial state conditions and desired final orientations are set and the model is run. Simulated SimSat quaternion, body rate, reaction wheel rates, and reaction wheel accelerations are saved in a time history.

For an open-loop optimal control run, an optimal control trajectory generated using the methods in Sec. 3.3.2 is resampled to the 0.01 seconds used by SimSat. It is loaded into the workspace and the model is run. The same data is collected as in the PID simulation.

The data collected in simulation is discussed in Sec. 4.2 and compared to data collected on SimSat in Sec. 4.4.1.

3.6 Performance Metrics

In this research there are two performance metrics used to compare different trajectories and test runs: time and quaternion error. For the PID test runs and simulated data, the time metric is defined as the time when the spacecraft simulator's orientation is within 1 percent of the desired orientation. For the open loop optimal control and Near RTOC test runs and simulated data, this is defined as the time that the optimal control portion ends in the test run regardless of the SimSat's orientation at this time. This corresponds with the cost function of the optimal control problem found in Eq. 2.43. Data past this point will be ignored. This is a key performance characteristic as this research attempts to decrease the time in reorientation maneuvers of spacecraft. In this research, a smaller time to target orientation is desired.

The second performance characteristic is quaternion error as calcuated in Eq. 2.19. At the final time of the maneuver, the error quaternion will be calculated between the desired quarternion and actual quaternion. An additional calculation of

$$q_{error\ metric} = \sqrt{q^2_{error,1} + q^2_{error,2} + q^2_{error,3}} \tag{50}$$

will be used to find a scalar quaternion error for easier comparison. For this research, a smaller quaternion error at the final time is desired.

3.7 Summary

This chapter first discussed the hardware and software configuration of SimSat: the testbed itself, the Ground Station PC, and the tri-axis air bearing. The next section discussed the different control models implemented on the hardware. The third section covered the design of the different experiments conducted in this research. The fourth section discussed the generation of simulated data to validate data received from the hardware experiments. The final section describes performance metrics used to compare different test runs.

IV. Results and Analysis

4.1 Introduction

This chapter presents the results of the simulations and experiments described in Sec. 3.5 and Sec. 3.4, respectively. The first section discusses the results from the simualted PID and Open Loop Optimal Control reorientation maneuvers of 90 degrees and 180 degrees. The next section covers the experimental results for the three control models: PID, OLOC, and Near RTOC. The third section discusses the comparions between simulated data versus hardware data and the effects of varying different parameters in the Near RTOC script.

4.2 Simulated Results

This section discusses the results from the simulation described in Sec. 3.5 and their impact on what is expected in the hardware tests on SimSat. The first section is the results of a 90 degree and then a 180 degree Z-axis reorientation of a simulated SimSat using a PID control and the second is the same 90 degree and then 180 degree Z-axis reorientations using an open loop optimal control input that is precalculated using GPOPSII. The simulation uses the MOI matrix found in Eq. (44) and the reaction wheel MOI values found in Table 2. The equations that are used to propagate the state forward are found in Eq. (2.23) and Eq. (2.20). The simulation assumes that the SimSat starts at rest and has no external torques applied to it throughout the reorientation maneuver.

4.2.1 PID Control Simulation. This simulation uses the PID controller described in Sec. 3.3.1. The gains for the PID controller are found in Table 6.

4.2.1.1 90 Degree Z-Axis Reorientation. The first experiment is a 90 degree reorientation of SimSat about its Z axis. The SimSat starts at rest with an initial orientation of 3-2-1 Euler angles $0°, 0°, 0°$ which corresponds to quaternion parameters [0001] and uses its reaction wheels to achieve an orientation of $0°, 0°, 90°$.

The quaternion parameters for this maneuver are shown in Fig. 4.1. The body angular rates are shown in Fig. 4.2. The reaction wheel angular rates are shown in Fig. 4.3 and angular accelerations are shown in Fig. 4.4.

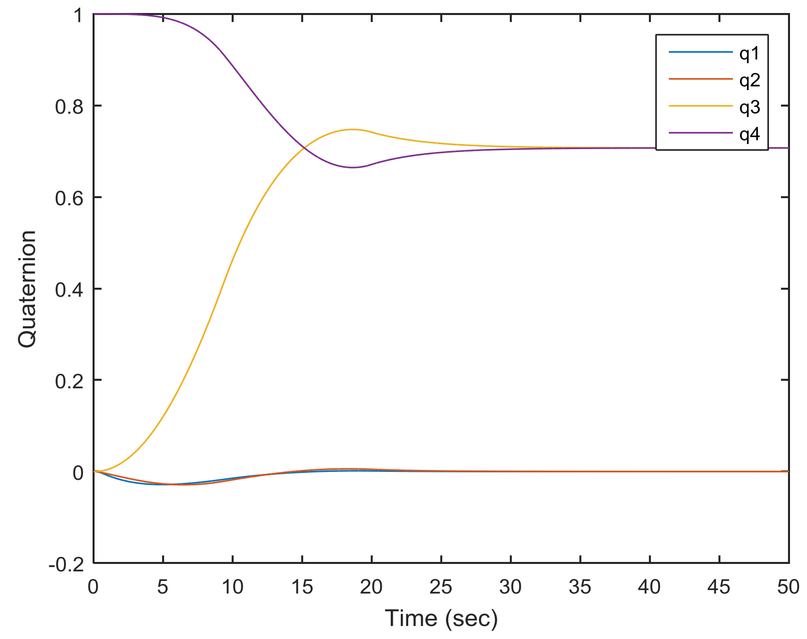

Figure 4.1: Simulated PID Controller 90 Degree Z-Axis Reorientation Quaternion Parameters

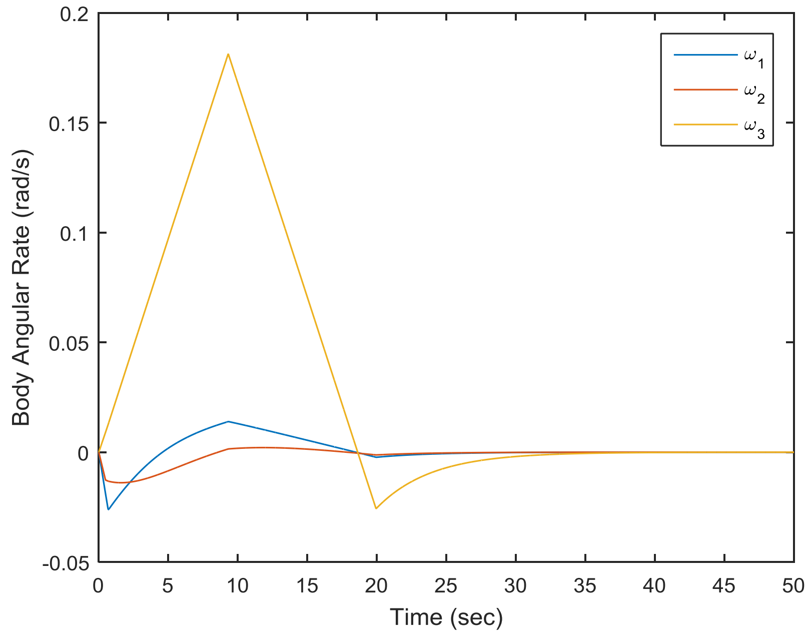

Figure 4.2: Simulated PID Controller 90 Degree Z-Axis Reorientation Body Angular Rates

The PID controller brings SimSat from its initial orientation to its final orientation within 1% in 29.9 seconds. There is no error at the final time as the PID controller

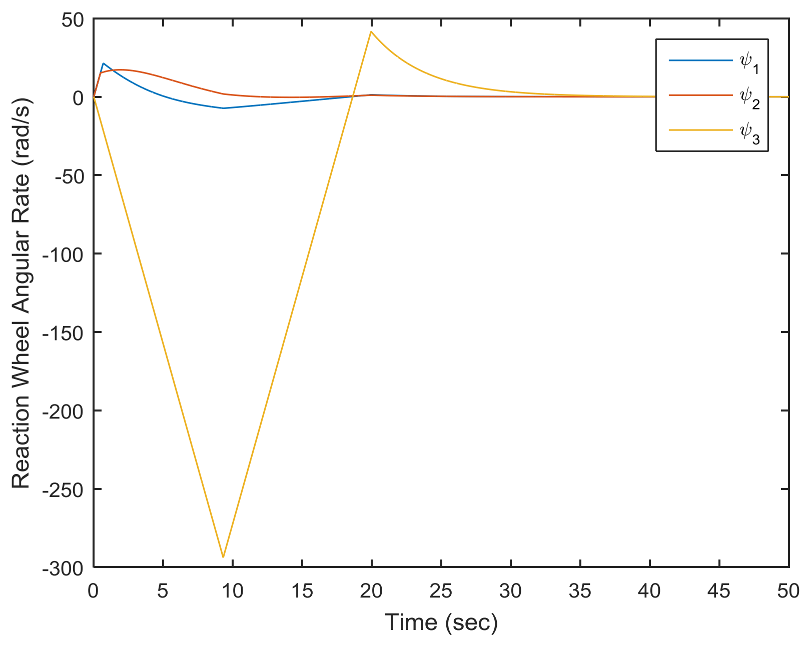

Figure 4.3: Simulated PID Controller 90 Degree Z-Axis Reorientation Reaction Wheel Angular Rates

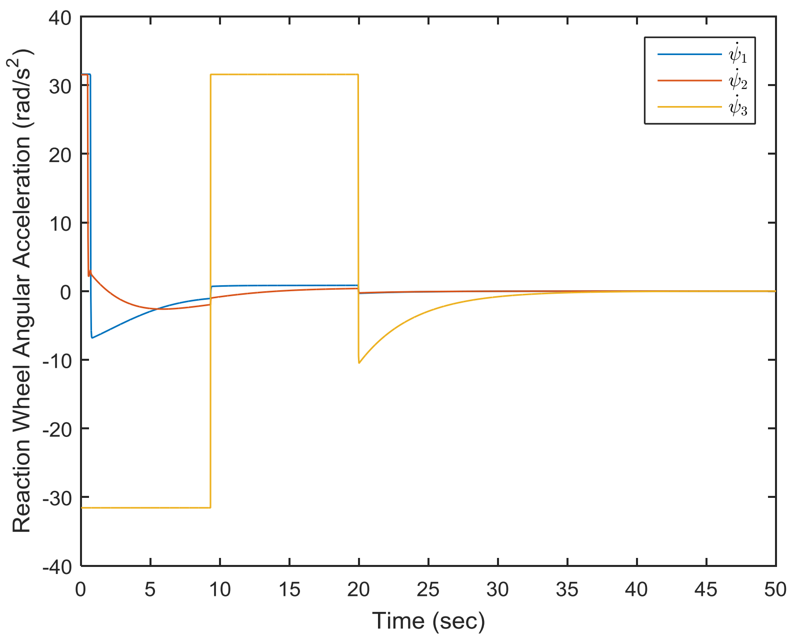

Figure 4.4: Simulated PID Controller 90 Degree Z-Axis Reorientation Reaction Wheel Angular Accelerations

brings the simulated SimSat directly to its final orientation of Euler angles $0°, 0°, 90°$. In Sec. 4.3.1.1, this simulated data will be shown to be a reasonably accurate simulation of

SimSat with respect to the experimental data which will be used as justification that the equations of motion used for the state equations in the optimal control solver GPOPS-II are reasonable. The simulation results also provide a baseline for optimal control results to be compared against as PID control is the most common means of spacecraft attitude control currently.

A point of note is that the plots of reaction wheel angular rates and accelerations, Fig. 4.3 and Fig. 4.4 respectively, show that SimSat's PID control model uses the reaction wheel aligned with the 3rd axis represented by ψ_3 almost exclusively during the maneuver. This shows that PID control on SimSat for a reorientation about one axis, in this case the Z-axis, primarily uses the reaction wheel aligned with that axis in the absence of external torques. The reaction wheels aligned with the X-axis and Y-axis, represented by ψ_1 and ψ_2 respectively, are used in the initial few seconds of the maneuver but are not generating the torque along the Z-axis during the majority of the reorientation. This limits the amount of torque produced along the axis of rotation and will be discussed further in Sec. 4.4.1.

4.2.1.2 180 Degree Z-Axis Reorientation. The second simulated experiment is a 180 degree reorientation of SimSat about its Z axis. Like with the 90 degree Z-axis reorientation, SimSat starts at an initial orientation of 3-2-1 Euler angles $0°, 0°, 0°$ and reorients itself to $0°, 0°, 180°$ using only the reaction wheels as actuators. The gain matrix used in this simulation is the same as the one in the 90 degree Z-axis reorientation and is found in Table 6.

The quaternion parameters for this maneuver are shown in Fig. 4.5. The body angular rates are shown in Fig. 4.6. The reaction wheel angular rates are shown in Fig. 4.7 and angular accelerations are shown in Fig. 4.8.

The maneuver takes 45.4 seconds to complete and there is no error at the final time for the same reasons as in the previous section. The PID model here again does not use the reaction wheels aligned with the X-axis or Y-axis to generate torque as shown in Fig. 4.7 and Fig. 4.8. The reaction wheel aligned with the Z-axis is the primary means of torque generation to reorient SimSat.

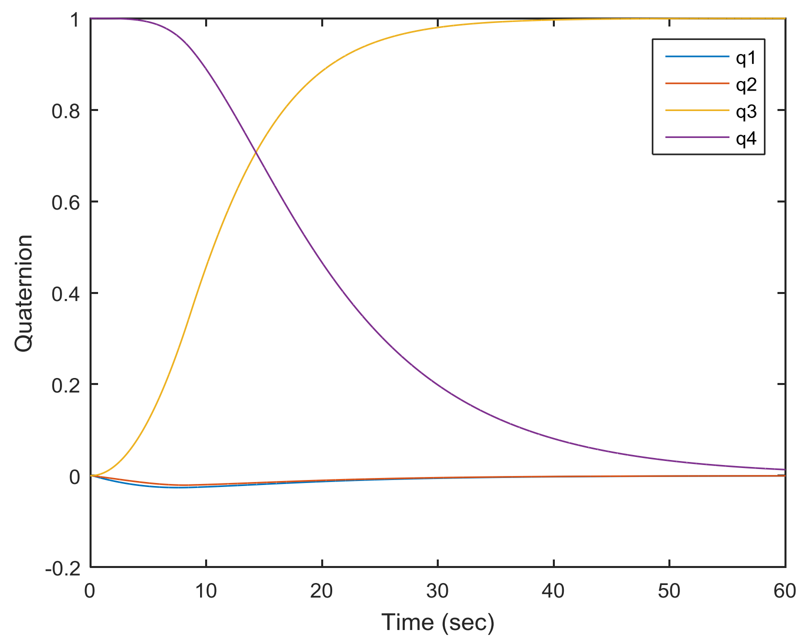

Figure 4.5: Simulated PID Controller 180 Degree Z-Axis Reorientation Quaternion Parameters

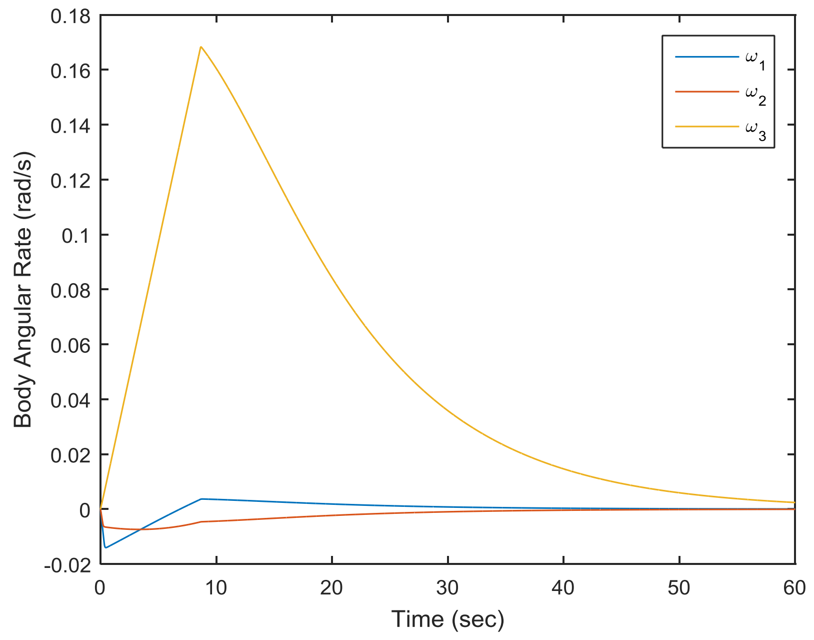

Figure 4.6: Simulated PID Controller 180 Degree Z-Axis Reorientation Body Angular Rates

4.2.2 Open Loop Optimal Control Simulation. This section describes the simulated results of 90 degree and 180 degree Z-axis reorientations using the open loop

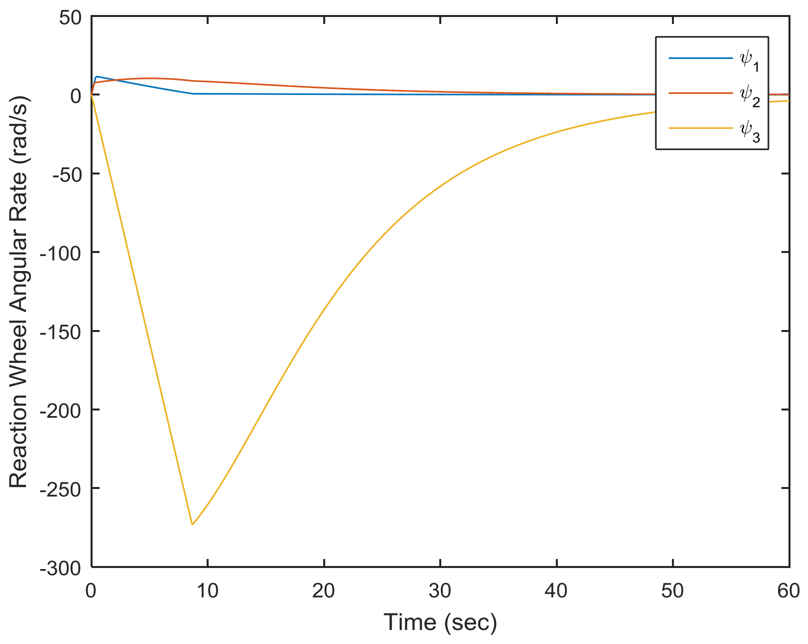

Figure 4.7: Simulated PID Controller 180 Degree Z-Axis Reorientation Reaction Wheel Angular Rates

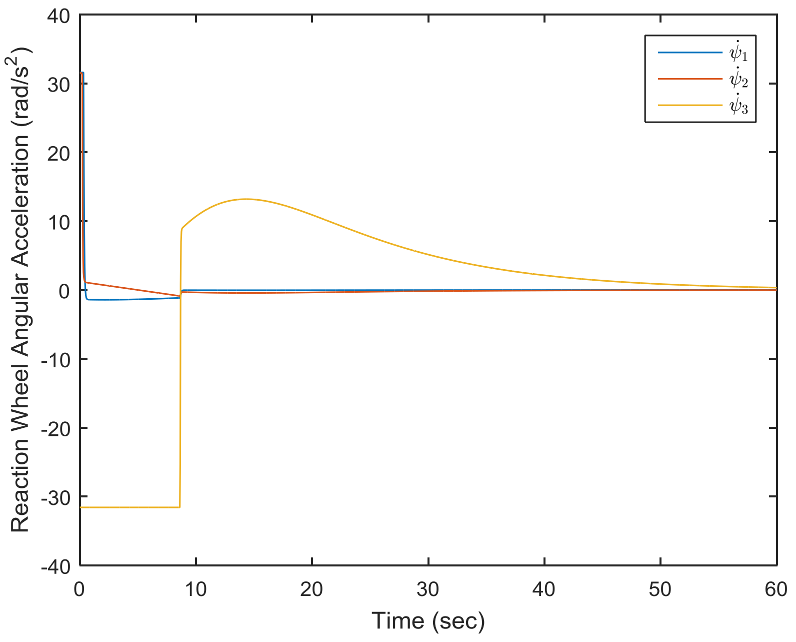

Figure 4.8: Simulated PID Controller 180 Degree Z-Axis Reorientation Reaction Wheel Angular Accelerations

optimal control model described in Sec. 3.3.2. These results are crucial as they represent the time-optimal“’best case” scenario for SimSat for these two maneuvers. The graphs here will be used again in Sec. 4.3.3 to be compared agains the hardware results of the Near RTOC controller since it is infeasible to simulate results for Near RTOC.

4.2.2.1 90 Degree Z-Axis Reorientation. The first open loop optimal control experiment is a 90 degree reorientation of SimSat about its Z-axis. The initial state is found in Eq. 46 and the final desired state is

$$\begin{bmatrix} q_{1,final} \\ q_{2,final} \\ q_{3,final} \\ q_{4,final} \\ \omega_{1,final} \\ \omega_{2,final} \\ \omega_{3,final} \\ h_{1,final} \\ h_{2,final} \\ h_{3,final} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \\ 0 \\ 0 \\ \text{free} \\ \text{free} \\ \text{free} \end{bmatrix} \tag{4.1}$$

.

Like the PID tests in the previous section this is intended to be a rest-to-rest maneuver. For this experiment the time plot of quaternion parameters is found in Fig. 4.9 and the time plot of body angular rates is found in Fig. 4.10. The reaction wheel angular rates and accelerations are found in Fig. 4.11 and Fig. 4.12 respectively.

Each of the four plots is a linear interpolation of the LGR points versus time. There are 10,000 collocation points that are linearly interpolated between to create a continuous plotof the time history of that state variable.

SimSat completes its maneuver from an initial orientation of 3-2-1 Euler angles $0°, 0°, 0°$ to $0°, 0°, 90°$ in 16.46 seconds, which is significantly faster than the PID controller which had a rest-to-rest time of 29.9 seconds. This represents a 44.9% improvement over the traditional PID-basd control for the gain setting selected for PID control found in Table 6. There is no final error in the simulated open loop result. Some error

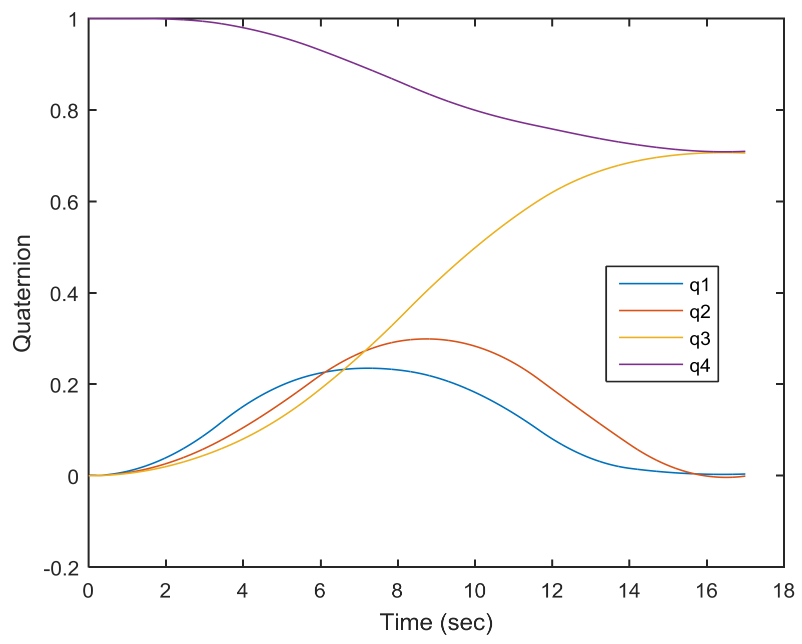

Figure 4.9: Simulated Open Loop Optimal Control 90 Degree Z-Axis Reorientation Quaternion Parameters

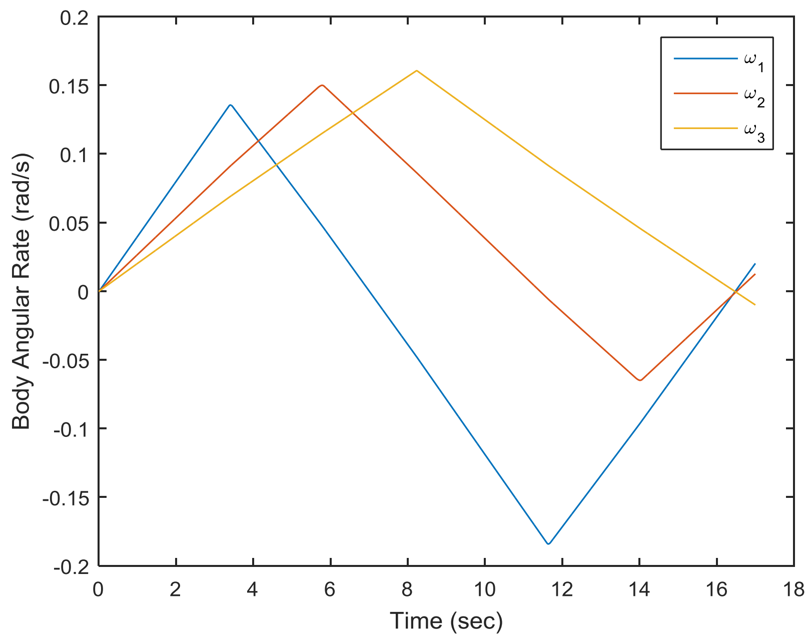

Figure 4.10: Simulated Open Loop Optimal Control 90 Degree Z-Axis Reorientation Body Angular Rates

is expected on the hardware model due to imbalances on the spacecraft simulator which generate additional torques not accounted for in the state equations and leave SimSat un-

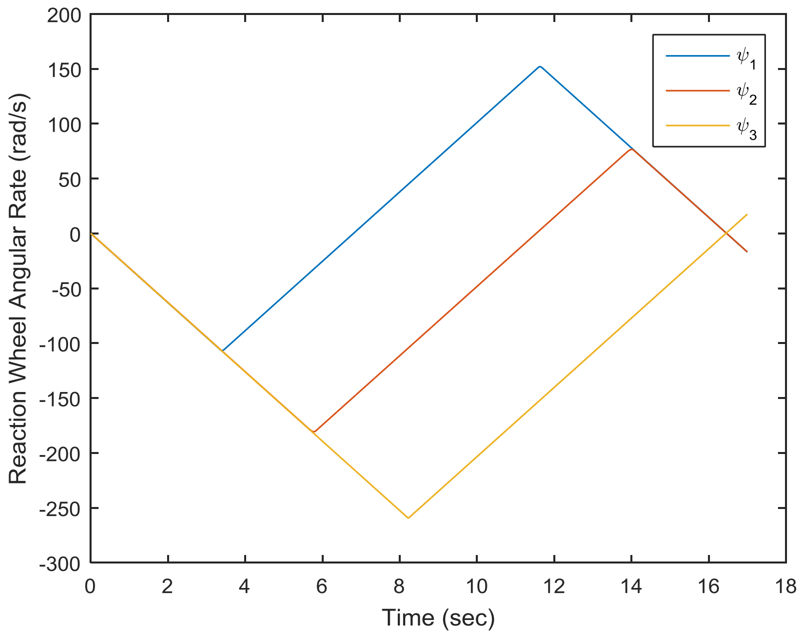

Figure 4.11: Simulated Open Loop Optimal Control 90 Degree Z-Axis Reorientation Reaction Wheel Angular Rates

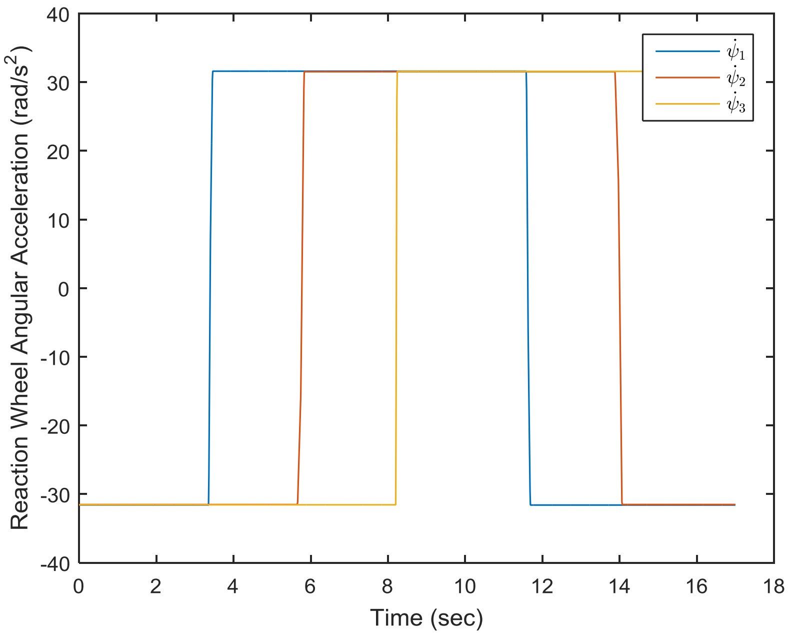

Figure 4.12: Simulated Open Loop Optimal Control 90 Degree Z-Axis Reorientation Reaction Wheel Angular Accelerations

able to respond to them due to the open loop nature of the controller. Another potential source of error is air drag which is not considered in the on-board model.

In Fig. 4.10, it can be sheen that SimSat in a 90 degree Z-axis open loop optimal control maneuver is rotating about all three axes significantly compared to most of its rotation about one axis, the Z-axis, while using a PID controller in Fig. 4.2. This is also shown in the greater variation of q_1 and q_2 in the plots of quaternions in Fig. 4.9 than in Fig. 4.1 which uses PID control. SimSat "dips down" and uses additional reaction wheels aligned with the body X-axis and Y-axis as shown in Fig. 4.11 and Fig. 4.12 to generate additional torque when using an optimal control trajectory than using a PID controller. This is why SimSat in simulation demonstrates a 44.9% improvement when using open loop optimal control. A more finely tuned PID controller may reduce the amount of improvement but is not considered in this research. However as shown in Sec. 4.3.2.1 and Sec. 4.3.2.2 on the actual hardware the open loop optimal controller generates significant error along the trajectory but especially near the terminal phase due to the disturbance torques described in the previous paragraph. Due to the cross-coupling terms found in Euler's equation rotation about one axis will produce rotation about the other two axes. These disturbance torques are why Near RTOC can apply optimal control techniques to spacecraft control to improve on PID or other traditional methods and is a better method than open loop optimal control.

4.2.2.2 180 Degree Z-Axis Reorientation. This section describes the results of a 180 degree Z-axis rotation of SimSat using an open loop optimal control trajectory. The initial state is found in Eq. 46 and the final desired state of SimSat is

$$\begin{bmatrix} q_{1,final} \\ q_{2,final} \\ q_{3,final} \\ q_{4,final} \\ \omega_{1,final} \\ \omega_{2,final} \\ \omega_{3,final} \\ h_{1,final} \\ h_{2,final} \\ h_{3,final} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \text{free} \\ \text{free} \\ \text{free} \end{bmatrix} \tag{4.2}$$

.

The quaternion parameters vs time plot is shown in Fig. 4.13. The body angular rates vs time are shown in Fig. 4.14. The reaction wheel angular rates vs time are shown in Fig. 4.15 and the reaction wheel angular accelerations vs time are shown in Fig. 4.16.

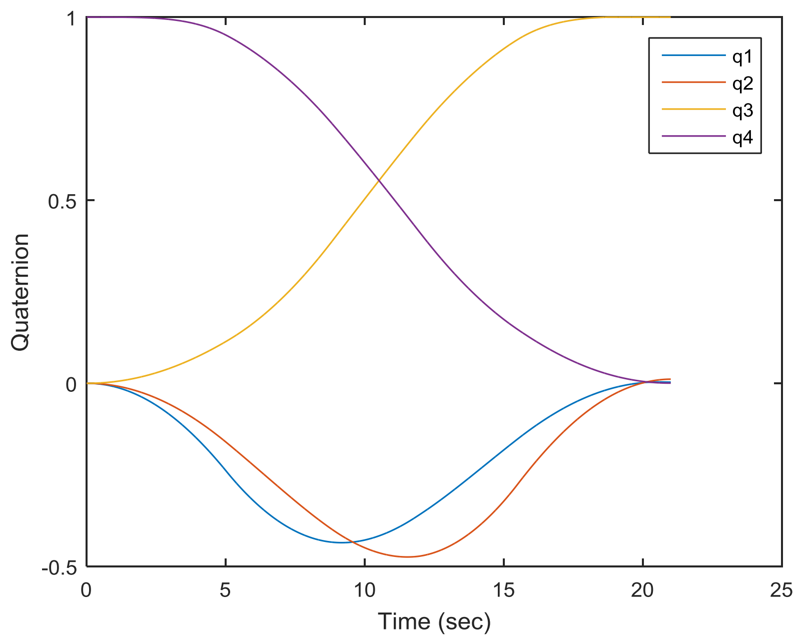

Figure 4.13: Simulated Open Loop Optimal Control 180 Degree Z-Axis Reorientation Quaternion Parameters

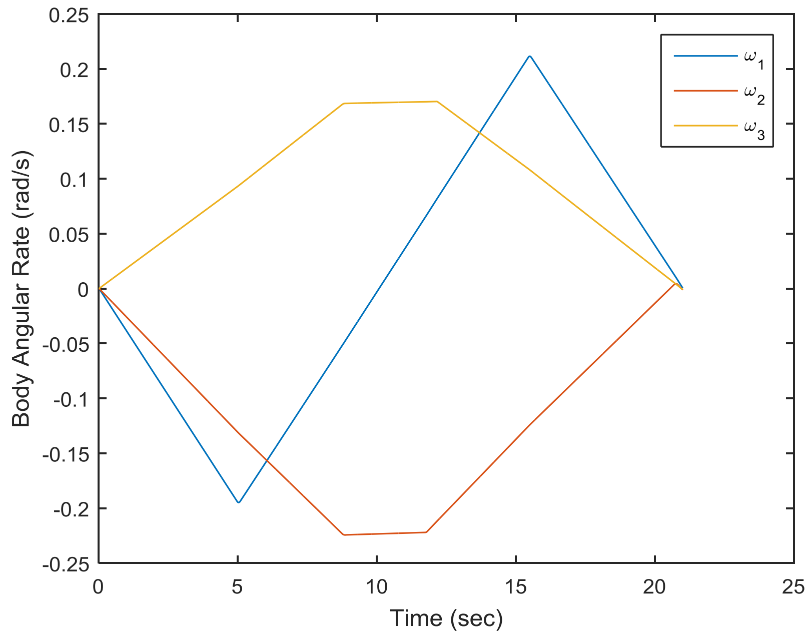

Figure 4.14: Simulated Open Loop Optimal Control 180 Degree Z-Axis Reorientation Body Angular Rates

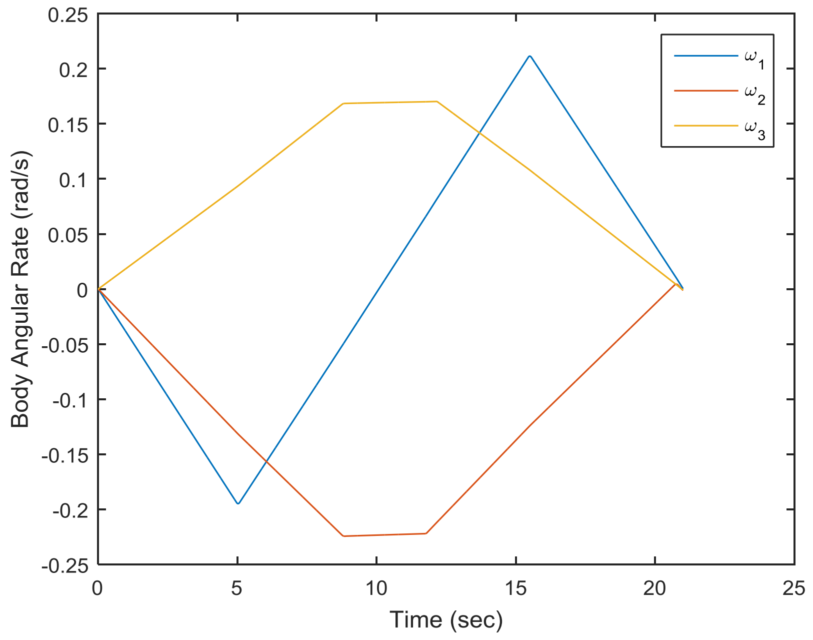

Figure 4.15: Simulated Open Loop Optimal Control 180 Degree Z-Axis Reorientation Reaction Wheel Angular Rates

The simulated SimSat completes the maneuver in 20.94 seconds, a 53.88% improvement over the PID control simulated.As in the previous section SimSat uses the reaction

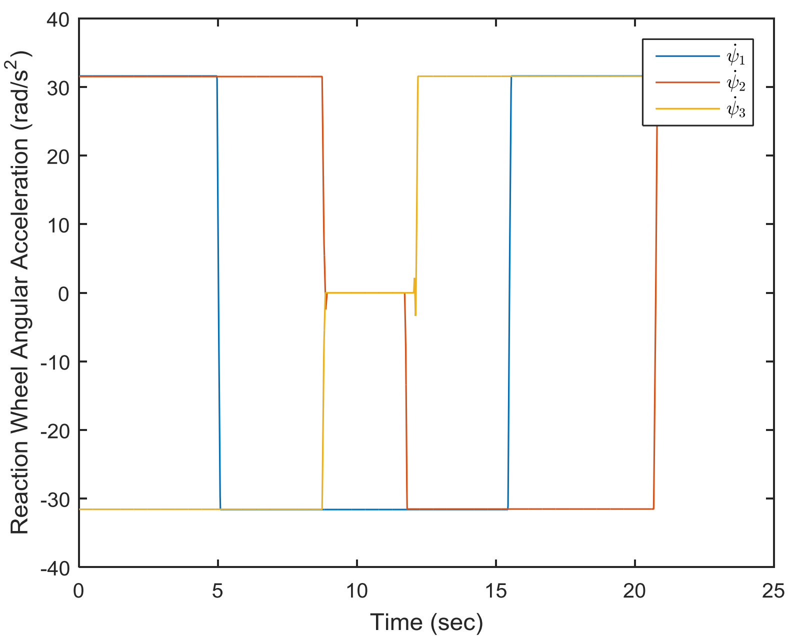

Figure 4.16: Simulated Open Loop Optimal Control 180 Degree Z-Axis Reorientation Reaction Wheel Angular Accelerations

wheels aligned with the X-axis and Y-axis, shown in Fig. 4.15 and Fig. 4.16, to generate additional torque which results in a faster maneuver.

4.3 Hardware Results

This section discusses the results from SimSat hardware test runs. The first section covers 90 degree and 180 degree Z-axis reorientations of SimSat using a PID controller and compares the results to the simulated results found in Sec. 4.2.1. The second section covers the hardware results same two maneuvers with an open loop optimal control trajectory and is compared to the simulated results found in Sec. 4.2.2. The final section describes the results of the two reorientations using Near RTOC techniques and is compared to the best-case open loop simulation also found in Sec. 4.2.2.

4.3.1 PID Control. This section covers hardware testing using SimSat's PID controller.

4.3.1.1 90 Degree Z-Axis Reorientation. The first reorientation is of 90 degrees about the Z-axis and is compared to Sec. 4.2.1.1. The plot of quaternions vs time is found in Fig. 4.17. The plot of body angular rates vs time is found in Fig. 4.18. The plot of reaction angular rates vs time is found in Fig. 4.19 and the plot of reaction wheel angular accelerations vs time is found in Fig. 4.20.

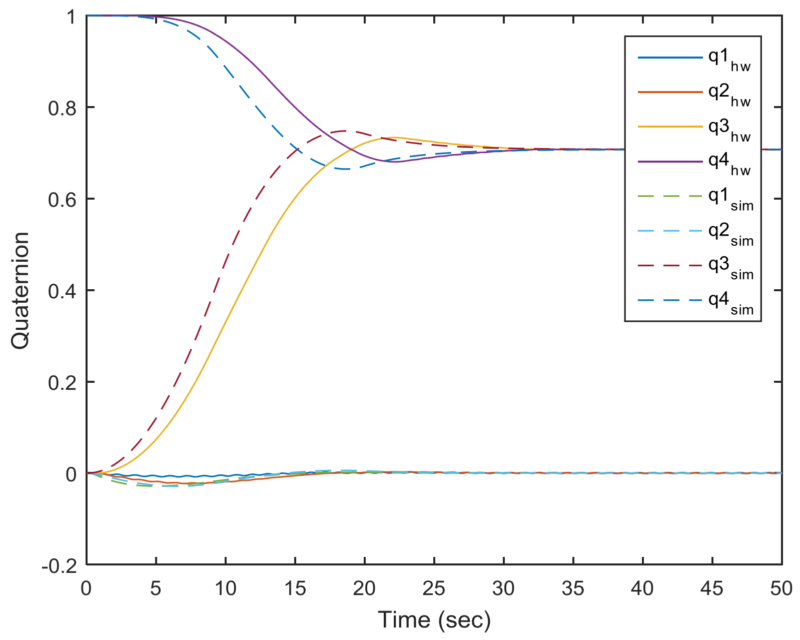

Figure 4.17: Experimental and Simulated PID Controller 90 Degree Z-Axis Reorientation Quaternion Parameters

As can be seen in Fig. 4.18 there is a great deal of variation about the X-axis and Y-axis found in the system not only with both body rates determination but also in the reaction wheel rate and acceleration measurement. This could be caused by the PID gain settings or by noise in the attitude determination system. The quaternion parameter trajectory is not affected by this. If the variation about 0 on the X-axis and Y-axis is caused by the PID gains, new gains could be calculated. If it is noise, a filter could be added to the system to improve the accuracy of measurement. Due to the limited time available for this research and the lack of effect this had on the spacecraft's trajectory neither were implemented. This idea is expanded upon in Sec. 5.2. The reaction wheel angular accelerations found in Fig. 4.20 are not the measured angular accelerations but the commanded angular accelerations. The reaction wheels do not reach their max angular speed of 274 rad/s and instead are limited to a max angular speed of just over 200 rad/s. This may be the reaction wheel motor keeping the reaction

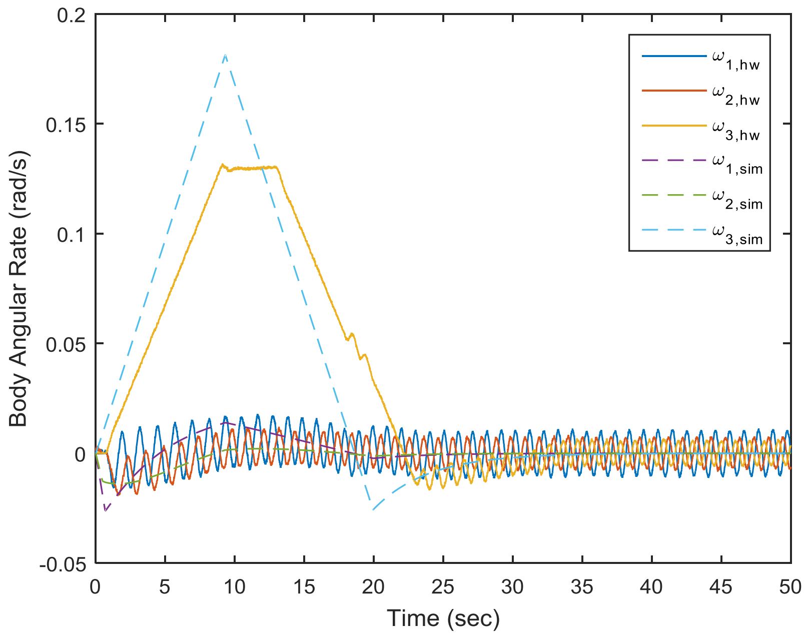

Figure 4.18: Experimental and Simulated PID Controller 90 Degree Z-Axis Reorientation Body Angular Rates

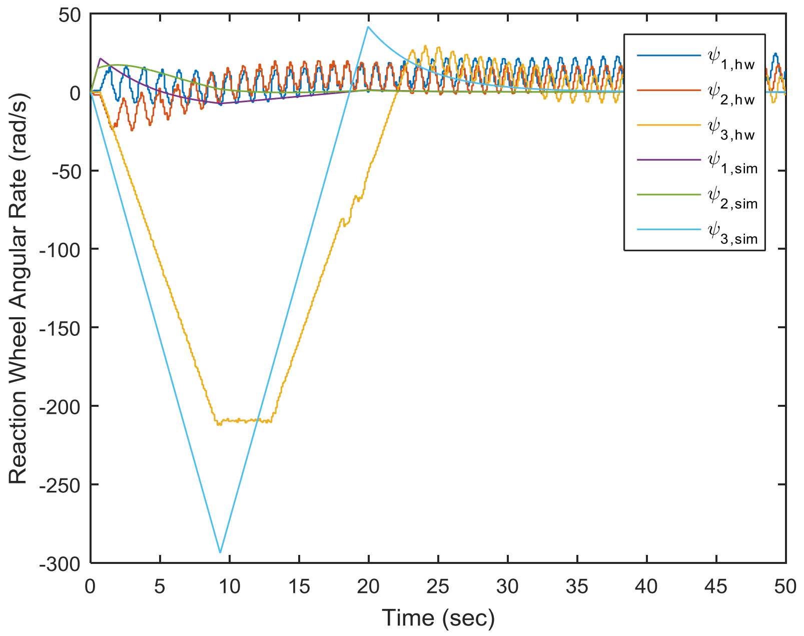

Figure 4.19: Experimental and Simulated PID Controller 90 Degree Z-Axis Reorientation Reaction Wheel Angular Rates

wheels from saturation. The fact that maximum reaction wheel rates on the SimSat are

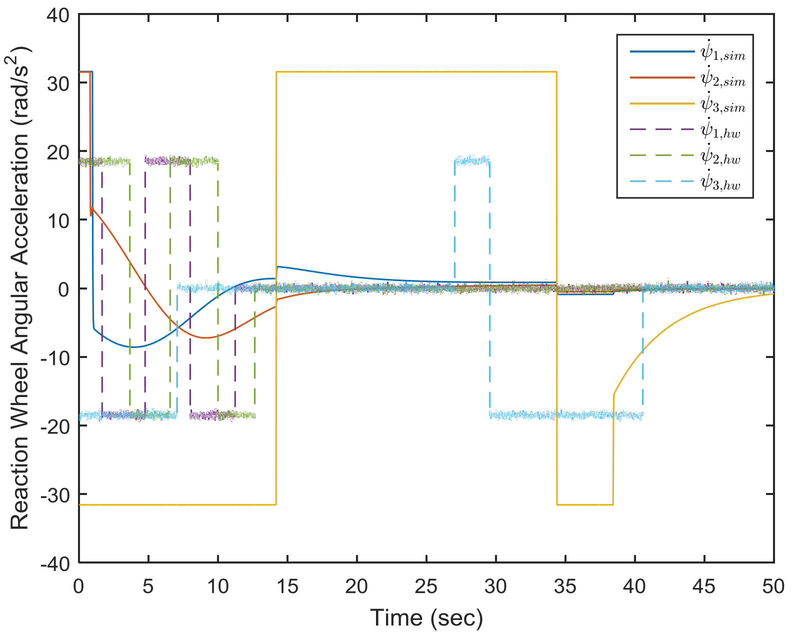

Figure 4.20: Experimental and Simulated PID Controller 90 Degree Z-Axis Reorientation Reaction Wheel Angular Accelerations

less than what the wheels and motors are intended for should be accounted for in future research.

The spacecraft simulator completes the reorientation maneuver in 30.3 seconds, slightly slower than the simulated SimSat's 29.4 seconds, and there is no terminal error at the end of the maneuver.

The hardware experiment's quaternion traejctory follows the "shape" of the simulated plot but takes a longer time to complete the maneuver. The plot of the body angular rate comparing the simulated body rates about each axis to the hardware body rates show the same difference. This could be due to a number of factors. First, the simulation as well as the hardware model in Simulink on SimSat itself do not account for air drag. This could be created by the movement of SimSat through its reorientation or by the reaction wheels spinning up or down. This would not be a factor in a space environment with negligible atmosphere but have an impact at ground level where SimSat is located. The second reason is imbalances on SimSat causing additional gravity torques not accounted for in either model but only present when the test is conducted on actual hardware. This issue is discussed further in the OLOC and Near RTOC results in Sec. 4.3.2 and Sec. 4.3.3 as it has a greater impact on those control methods. These two factors explain the discrepancy between the simulation and hardware model. The state equations used in the simulation, which are also the equations used in calculating optimal control solutions, are verified.

4.3.1.2 180 Degre Z-Axis Reorientation. This experiment is similar to the previous one but has SimSat reorient itself 180 degrees about its Z-axis instead of 90-degrees. The plot of quaternions vs time comparing the hardware results to the simulated results found in Sec. 4.2.1.2 is found in Fig. 4.21. The plot of body angular rates vs time is found in Fig. 4.22. The results of the reaction wheel angular rates is found in Fig. 4.23 and reaction wheel angular accelerations are found in Fig. 4.24.

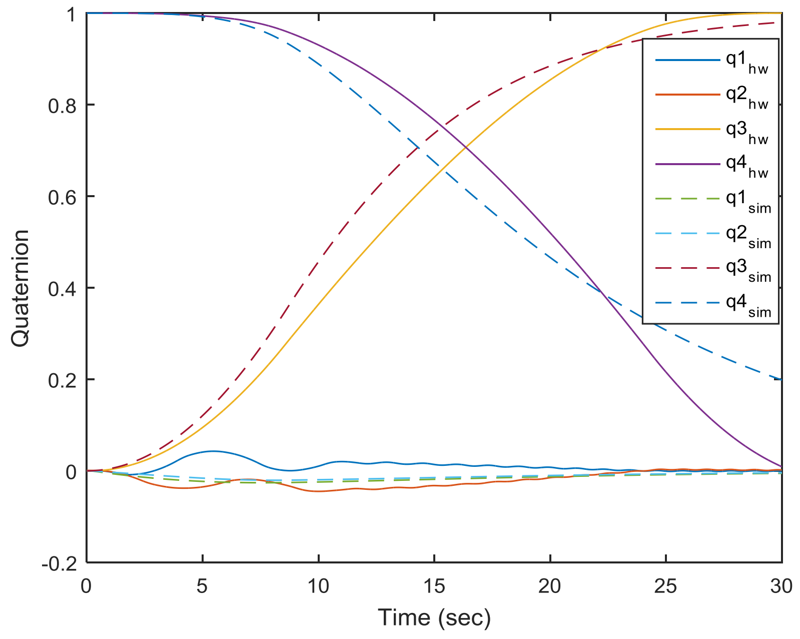

Figure 4.21: Experimental and Simulated PID Controller 180 Degree Z-Axis Reorientation Quaternion Parameters

The plot of quaternions vs time is similar to that of the 90 degree Z-axis PID reorientation found in Fig. 4.17 with the shape of the quaternion trajectory of the hardware test run being similar to that of the simulation but taking more time to complete the maneuver. The hardware takes 49.9 seconds to reach the desired orientation compared to 45.4 seconds in the simulation. There is no error in the simulation due to the nature of

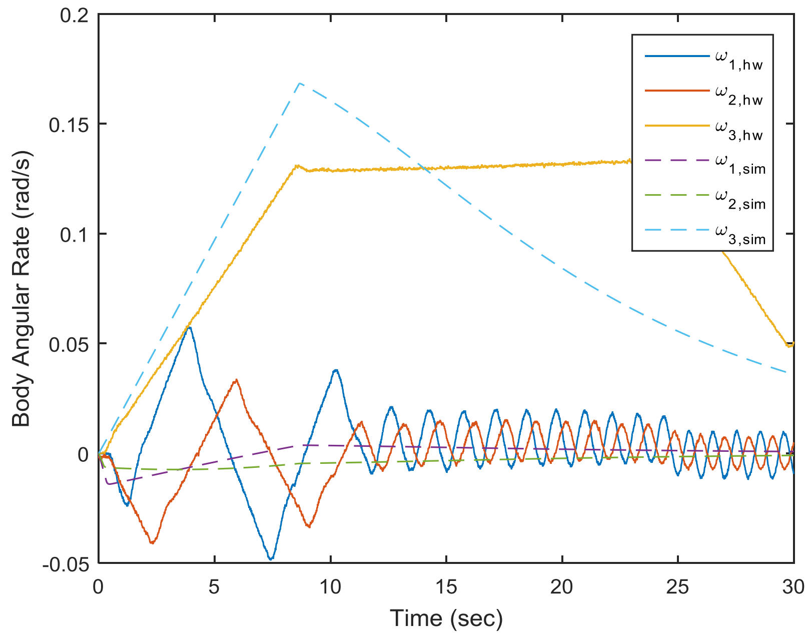

Figure 4.22: Experimental and Simulated PID Controller 180 Degree Z-Axis Reorientation Body Angular Rates

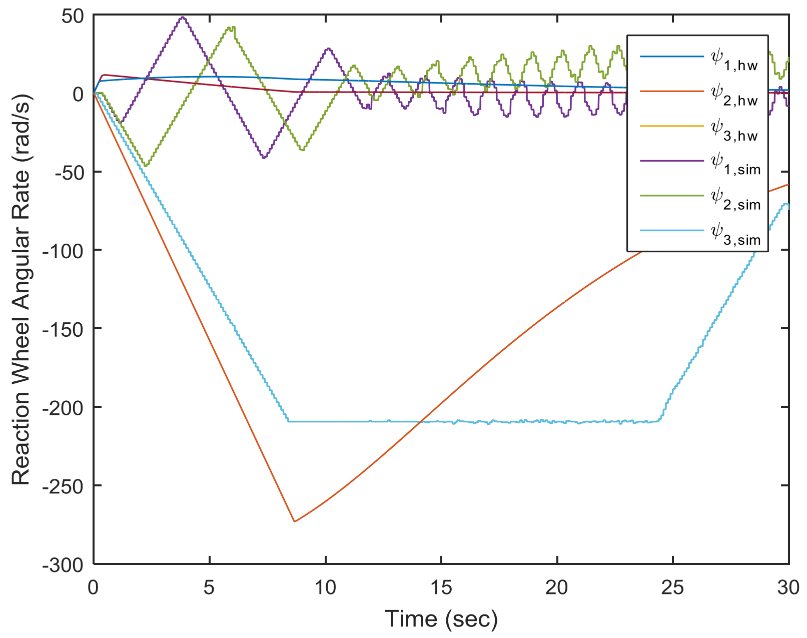

Figure 4.23: Experimental and Simulated PID Controller 180 Degree Z-Axis Reorientation Reaction Wheel Angular Rates

the PID controller. The spacecraft simulator does not reach a rest state but the quaternion meets the final error requirements. The variation found in Fig. 4.22, Fig. 4.23, and

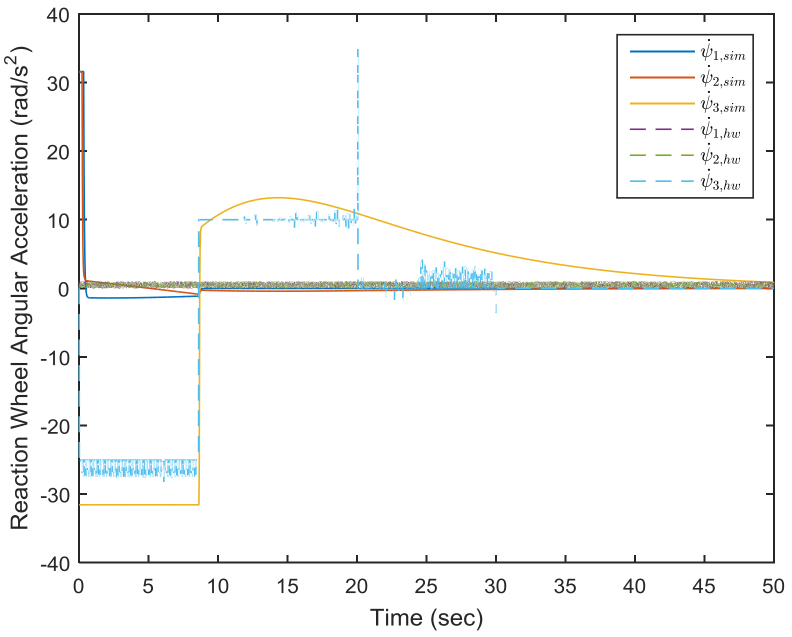

Figure 4.24: Experimental and Simulated PID Controller 180 Degree Z-Axis Reorientation Reaction Wheel Angular Accelerations

Fig. 4.24 around 0 about the X-axis and Y-axis is in this experimental result as well but does not affect the calculation of the quaternion parameters from the body rates.

The plots of the hardware body angular rates and reaction wheel rates and accelerations look very different than their simulations. This is due to the linearization assumptions used to create the PID feedback controller. Small angle assumptions, small body angular rates, and $M_{ext} = 0$ are all used in the creation of the PID controller. In this reorientation, they stop being valid as the spacecraft reorients itself 180 degrees about its Z-axis with high body rates. This works in the simulation as there are no disturbance torques but on the SimSat hardware there are due to the gravity environment. The simulation assumes a torque-free enviornment on orbit but since SimSat is on the ground gravity disturbance torques act on it during the maneuver. This is both a limitation of the hardware as well as of the PID controller. An open loop optimal control scheme is less dependent on linearization and with the additional of a near real-time component can peform large-angle slews faster and more accurately than a PID-controlled system. This idea is discussed further in the next two sections.

4.3.2 Open Loop Optimal Control. This section discusses the results of hardware testing of SimSat using an open loop optimal control trajectory found using GPOPS-II. These results are compared to the simulations run in Sec. 4.2.2.

4.3.2.1 90 Degree Z-Axis Reorientation. The first open loop optimal control experiment run is a 90 degree reorientation about the Z-axis of SimSat. As described in Sec. 3.4 a precalculated optimal control trajectory consisting of an orientation trajectory and a time history of $\dot{\vec{h}}$ values is read into the Simulink model which calculates the reaction wheel accelerations $\dot{\vec{\psi}}$ that are commanded directly to the reaction wheel motors. There is no feedback as the spacecraft simulator continues along its planned trajectory regardless of disturbances or deviations.

The plot of quaternion parameters vs time for both the simulated and actual results is found in Fig. 4.25. The body angular rate comparison of simulated vs actual is found in Fig. 4.26. The plot of reaction wheel angular rates is found in Fig. 4.27. The plot of reaction wheel angular accelerations is found in Fig. 4.28 and is compared to the simulated $\dot{\vec{\psi}}$ found in Fig. 4.12. The $\dot{\vec{h}}$ values that were fed-forward into the MicroAutoBox as optimal control inputs are shown in Fig. 4.29.

As can be seen in the plot of quaternions, Fig. 4.25, there is some error between the simulation and the hardware result. The spacecraft simulator completes its maneuver in 19 seconds which is the end of the optimal control portion of the maneuver. There is a great deal of error at the end of the maneuver compared to the desired final quaternion in Fig. 4.21. The error metric calculated using Eq. (50) is 0.1886 or 18.86% error. This is in comparison to the simulated maneuver which completed in 16.46 seconds with 0 error.

The plots of quaternions and body angular rates show that SimSat is able to use all three reaction wheels to generate torque. It "tilts" up on one side so that all three wheels' acceleration are assisting in the maneuver and can generate additional torque than the PID case where only one reaction wheel generates the majority of the torque needed.

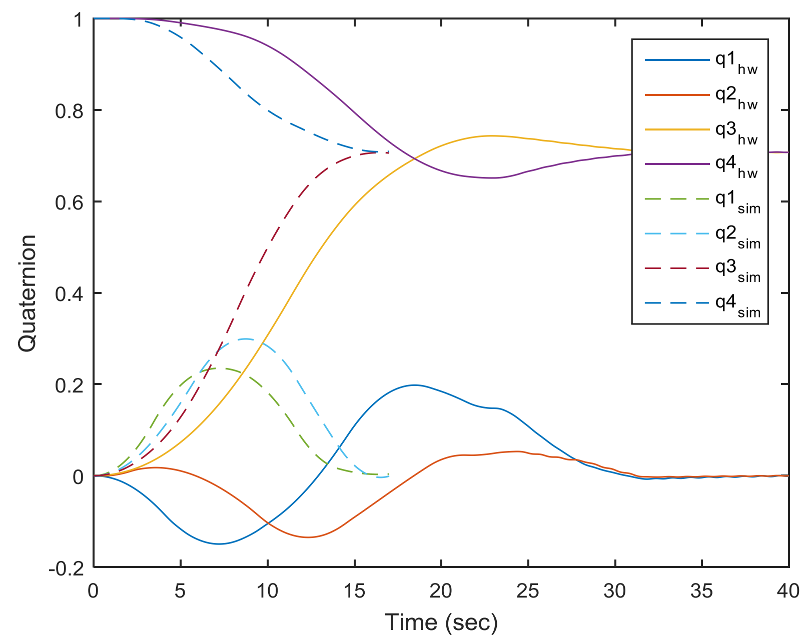

Figure 4.25: Experimental and Simulated Open Loop Optimal Controller 90 Degree Z-Axis Reorientation Quaternion Parameters

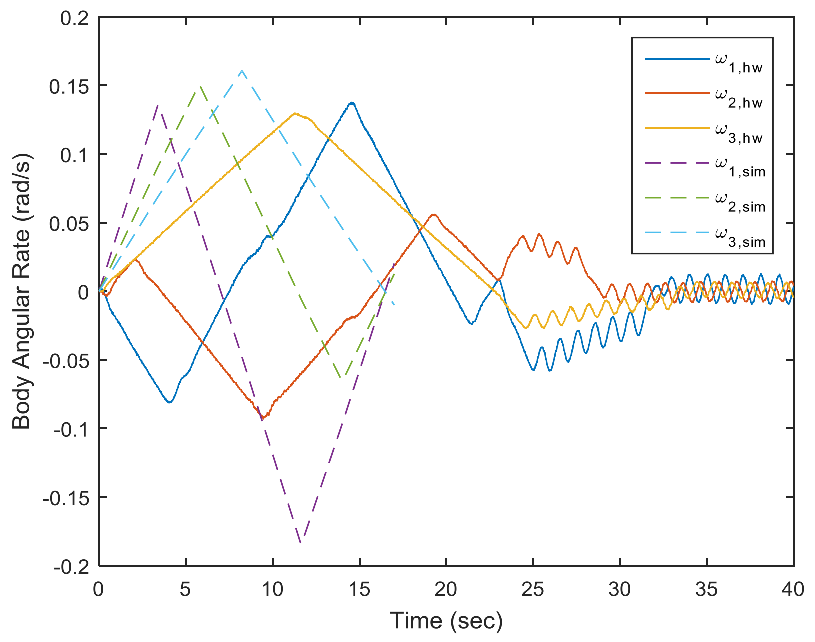

Figure 4.26: Experimental and Simulated Open Loop Optimal Controller 90 Degree Z-Axis Reorientation Body Angular Rates

The error at the end of the maneuver and the additional time required to complete it are caused by two factors. The first is gravity disturbance torques generated by SimSat

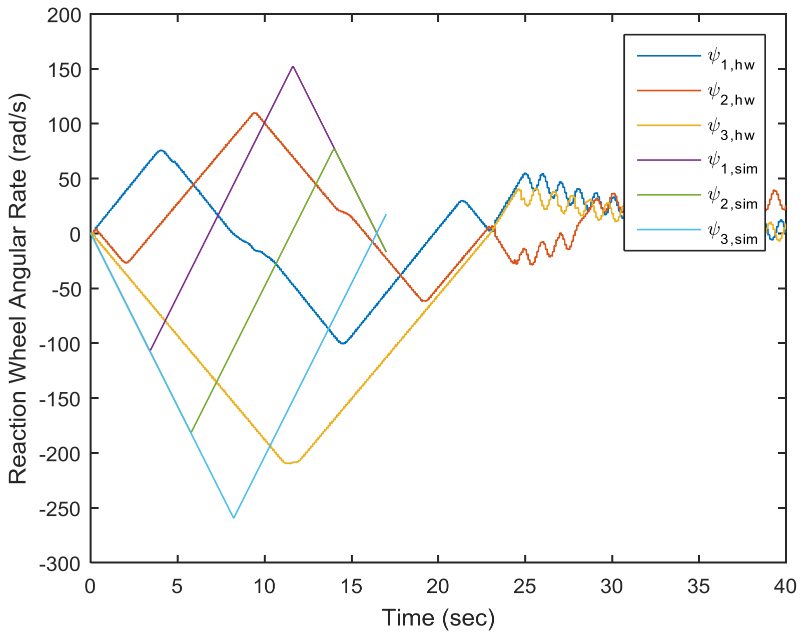

Figure 4.27: Experimental and Simulated Open Loop Optimal Controller 90 Degree Z-Axis Reorientation Reaction Wheel Angular Rates

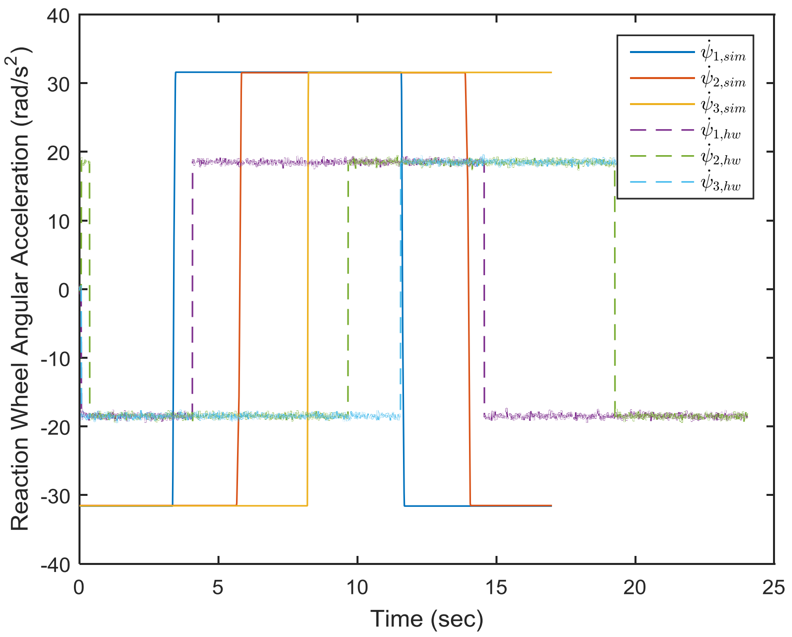

Figure 4.28: Experimental and Simulated Open Loop Optimal Controller 90 Degree Z-Axis Reorientation Reaction Wheel Angular Accelerations

as it rotates about its X-axis or Y-axis. The X-axis and Y-axis are well-balanced as shown in Fig. 4.17 and Fig. 4.21 where q_1 and q_2 do not stray too far from their simulated values

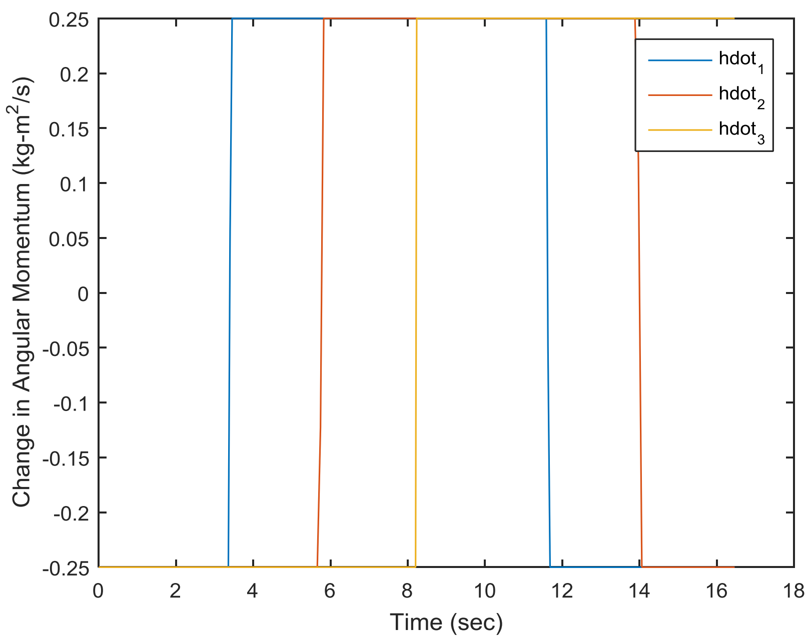

Figure 4.29: Experimental and Simulated Open Loop Optimal Controller 90 Degree Z-Axis Reorientation $\dot{\vec{h}}$ Trajectory

near 0 throughout the trajectory. However, in the open loop hardware run described in this section, q_1 and q_2 do not follow their pre-planned trajectory and show a significant amount of error through the manuever. While the mass moment of inertia is well known, the gravity gradient torques caused by SimSat not having a uniform shape and mass distribution are not accounted for.

The second is the hard-coded limits in the SimSat model. SimSat cannot rotate more than 25° about its X-axis or Y-axis. Doing so would result in a collision between a piece of SimSat and the air bearing. A $\dot{\vec{h}}$ trajectory that pushes the spacecraft simulator past these limits will find itself limited by the Simulink model. The planned quaternion trajectory violates these restrictions which amount to roughly ± 0.2 on q_1 and q_2. The model is able to account for this restriction and prevents SimSat from colliding with the tri-axial air bearing but this does change the commanded reaction wheel speeds. Fig. 4.27 shows that the commanded reaction wheel accelerations are significantly lower than the planned simulation values. This is a result of SimSat's Simulink model attempting to and succeeding in avoiding a collision. This is shown in Fig. 4.28 where the reaction wheel angular accelerations expected by the simulation are significantly less than the angular accelerations predicted by the simulation.

4.3.2.2 180 Degre Z-Axis Reorientation. This section covers the open loop optimal control reorientation of SimSat of 180 degrees about its Z-axis. The data from this experiment is compared to the simulated result found in Sec. 4.2.2.2. The plot of quaternion parameters vs time is found in Fig. 4.30. The plot of body angular rates vs time is found in Fig. 4.31. The plots of reaction wheel angular rates and accelerations vs time are found in Fig. 4.32 and Fig. 4.33 respectively. The $\dot{\vec{h}}$ trajectory used in this experiment is found in Fig. 4.34.

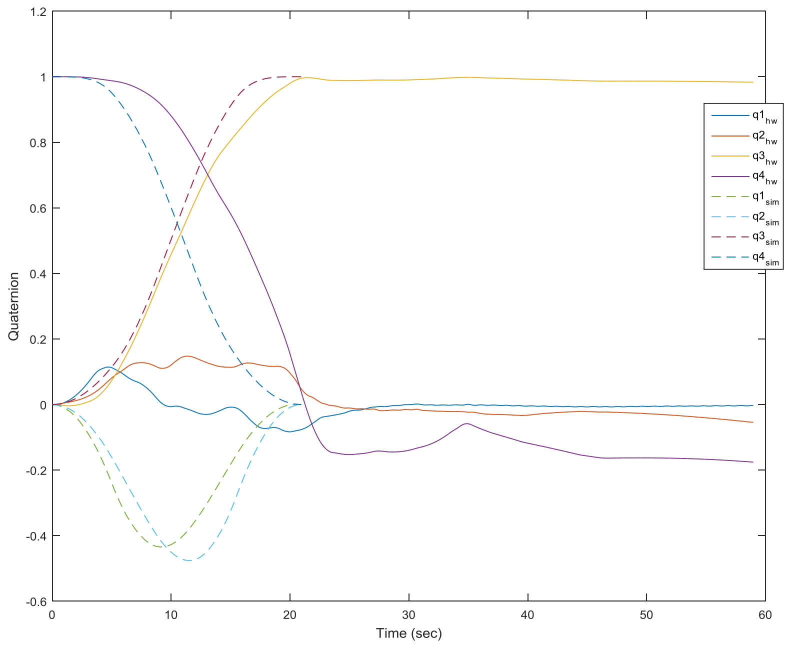

Figure 4.30: Experimental and Simulated Open Loop Optimal Controller 180 Degree Z-Axis Reorientation Quaternion Parameters

The maneuver is completed in 26.36 seconds and has 14.79% error at the end of the optimal control trajectory.

Fig. 4.30, Fig. 4.31, and Fig. 4.32 show that the planned trajectory and the actual trajectory that the hardware followed over the reorientation maneuver are very dissimilar. The previous three sections of hardware results, Sec. 4.3.1.1, Sec. 4.3.1.2, and Sec. 4.3.2.1 all show SimSat follow the shape of the planned reorientation trajectory closely, if not ex-

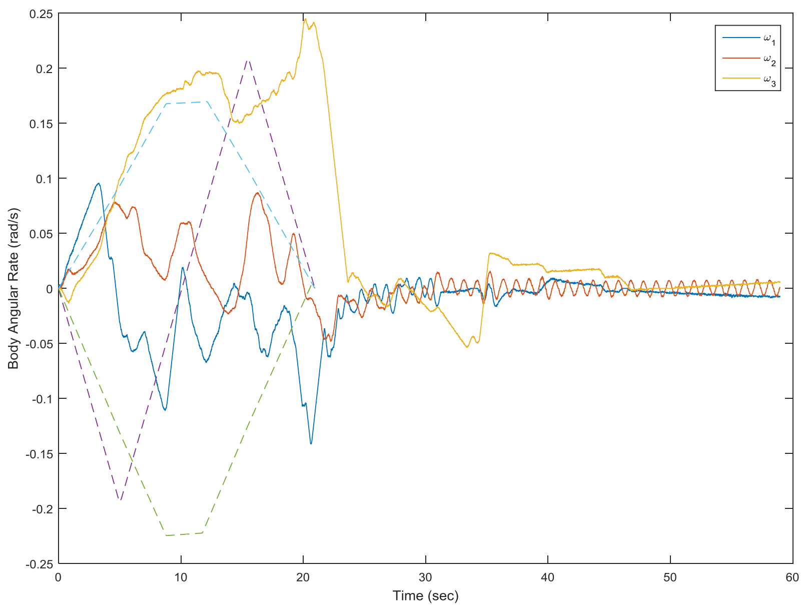

Figure 4.31: Experimental and Simulated 180 Degree Z-Axis Reorientation Body Angular Rates

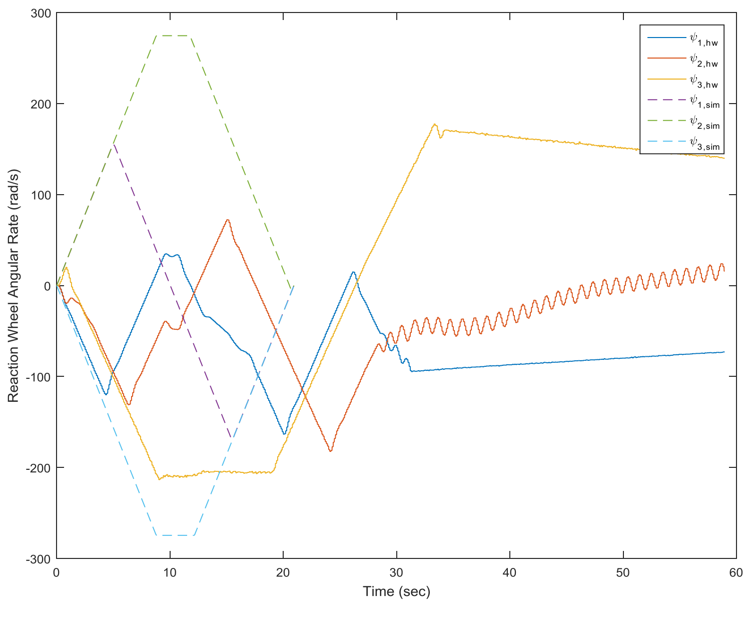

Figure 4.32: Experimental and Simulated Open Loop Optimal Controller 180 Degree Z-Axis Reorientation Reaction Wheel Angular Rates

actly, with minimal error throughout the maneuver. This deviation is due to the gravity disturbance torques and the hard-coded limits on the amount of torque provided by the

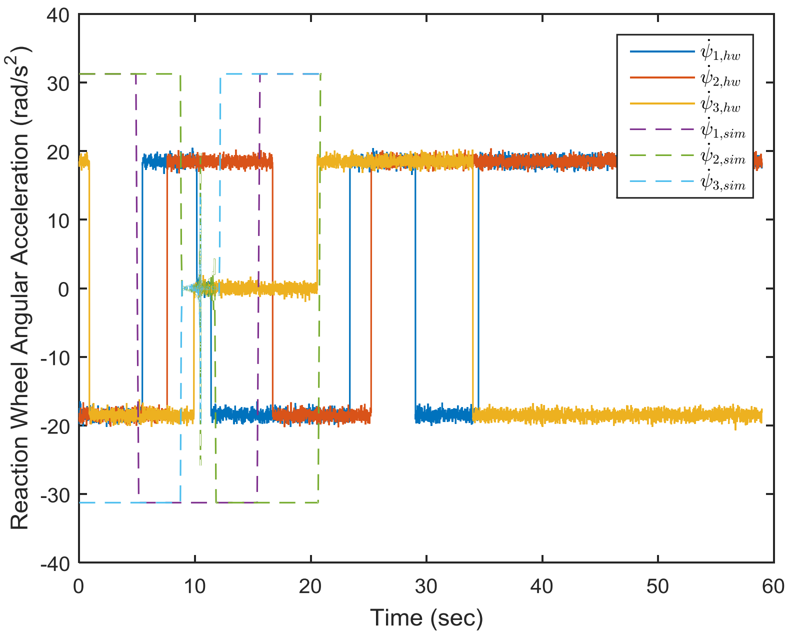

Figure 4.33: Experimental and Simulated Open Loop Optimal Controller 180 Degree Z-Axis Reorientation Reaction Wheel Angular Accelerations

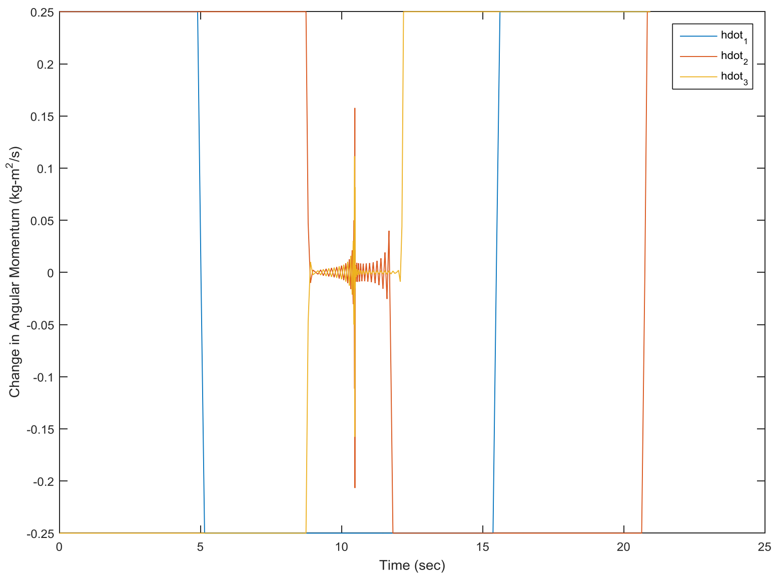

Figure 4.34: Experimental and Simulated Open Loop Optimal Controller 180 Degree Z-Axis Reorientation $\dot{\vec{h}}$ Trajectory

reaction wheels to limit collisions. Air drag is not accounted for in either the simulation or experimental model. The gravity torques on SimSat discussed in the previous section are able to act on SimSat for a longer period of time due to the greater length of the maneuver causing an increase in the error at the terminal phase of the optimal control trajectory. They are able to deviate SimSat significantly from its planned course. The simulation assumes that SimSat is able to rotate a full 360 degrees about its X-axis and Y-axis. In the hardware model, this is not the case. SimSat will generate less torque in order to keep itself from colliding with the air bearing that it is floating on.

4.3.3 Near RTOC.

This section discusses the results of experiments peformed on SimSat using a near RTOC controller. The first subsection discusses a 90 degree reorientation about SimSat's Z-axis and the second subsection covers the results of a 180 degree reorientation about SimSat's Z-axis. A cursory glance at the results of these experiments is found here in these two subsections while further in-depth analysis of the effectiveness of near RTOC and its comparison to traditional control techniques is found in Sec. 4.4.

4.3.3.1 90 Degree Z-Axis Reorientation.

A modified approach was used for the calculation of optimal control solutions in experiment than with the OLOC presented in Sec. 4.3.2.1 and Sec. 4.3.2.2. The Simulink model of SimSat included torque output constraint in order to prevent a collision between SimSat and its air bearing. The hardware results of open loop experiments deviated from the simulated results presented in Sec. 4.2.2.1 and Sec. 4.2.2.2 and had a great deal of error at the end of the maneuver, which was expected due to gravitational disturbances but not to the degree present. However, the OLOCled SimSat was still able to bring itself to the target. To counteract this issue, additional limits were set on the quaternion parameters q_1 and q_2 than was discussed in Sec. 3.3.2. Instead of being able to go between -1 and 1, they were limited to be between ± 0.2. This will keep the spacecraft simulator within its $\pm 25°$ limits on rotation about its X-axis and Y-axis. The quaternion parameters q_3 and q_4 are not limited. This constraint does not change the open loop optimal results in Sec. 4.3.2.1 and Sec. 4.3.2.2.

The plots of quaternion paramters vs time are found in Fig. 4.35. The plot of body angular rates vs time is found in Fig. 4.36. The plots of reaction wheel angular rates and accelerations are found in Fig. 4.37 and Fig. 4.38, respectively. The plots of the different $\dot{\vec{h}}$ trajcetories followed during the maneuver are found in Figs. 4.39, 4.40, and 4.41.

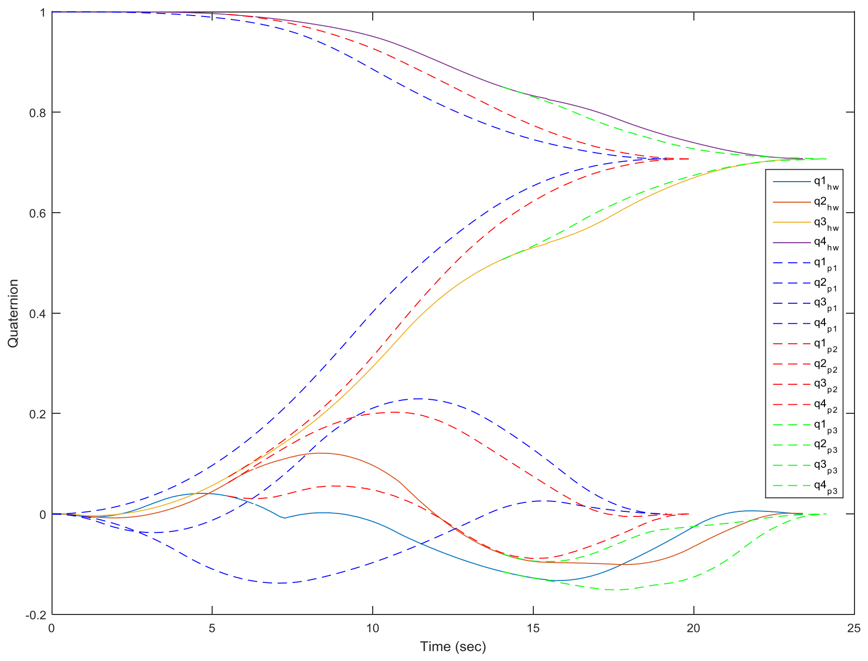

Figure 4.35: Hardware Near RTOC Controller 90 Degree Z-Axis Reorientation Quaternion Parameters

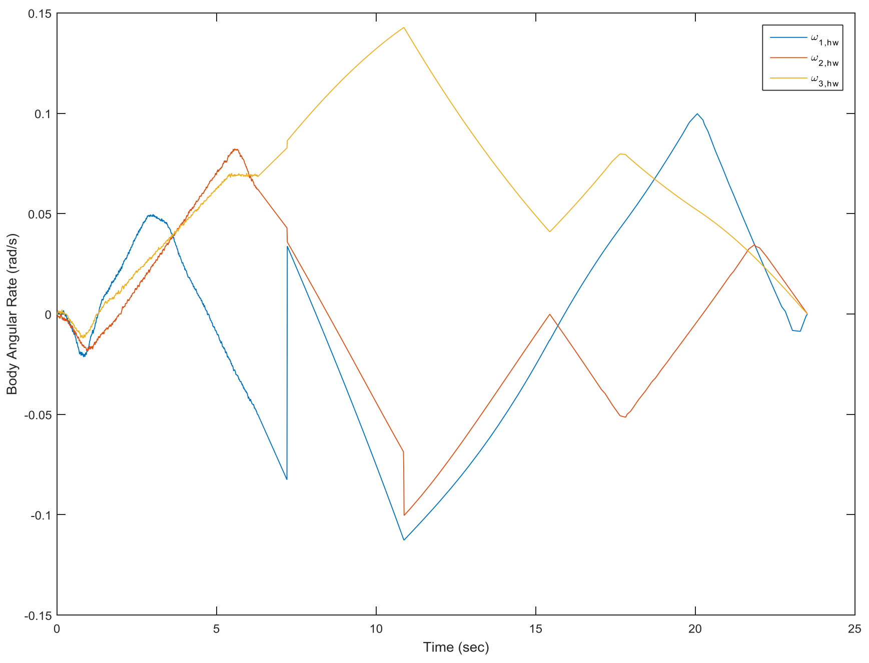

Figure 4.36: Hardware Near RTOC Controller 90 Degree Z-Axis Reorientation Body Angular Rates

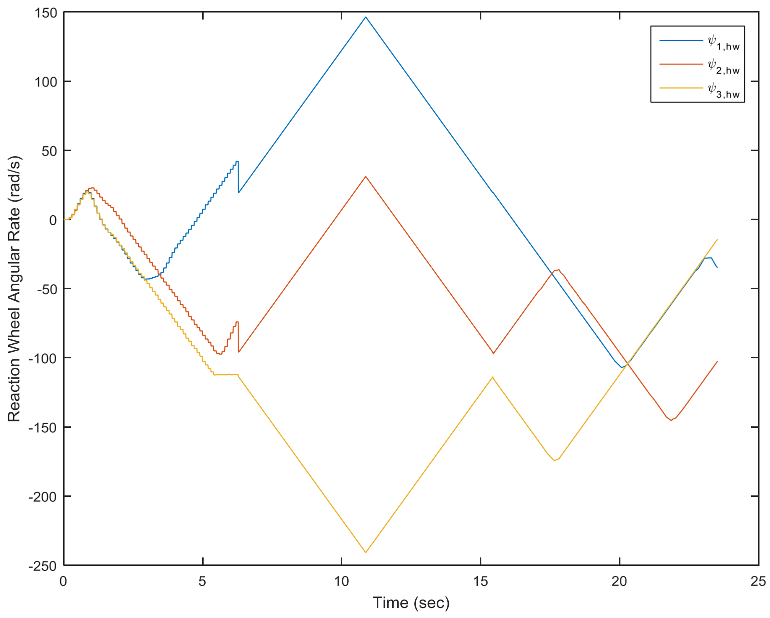

Figure 4.37: Hardware Near RTOC Controller 90 Degree Z-Axis Reorientation Reaction Wheel Angular Rates

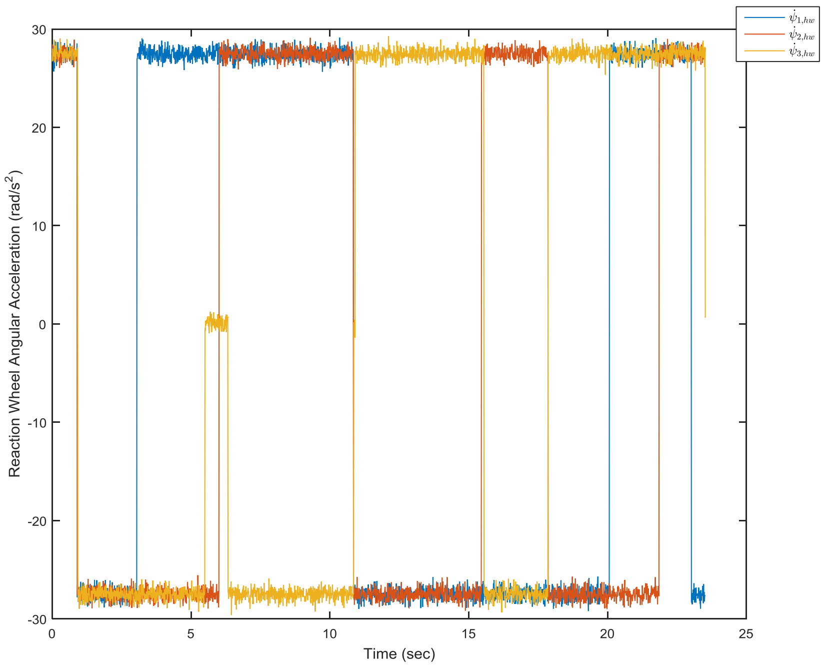

Figure 4.38: Hardware Near RTOC Controller 90 Degree Z-Axis Reorientation Reaction Wheel Angular Accelerations

In Fig. 4.35 the experimental quaternion parameters are shown in solid colors. The three calculated optimal trajectories are shown in dashed lines. The initial optimal

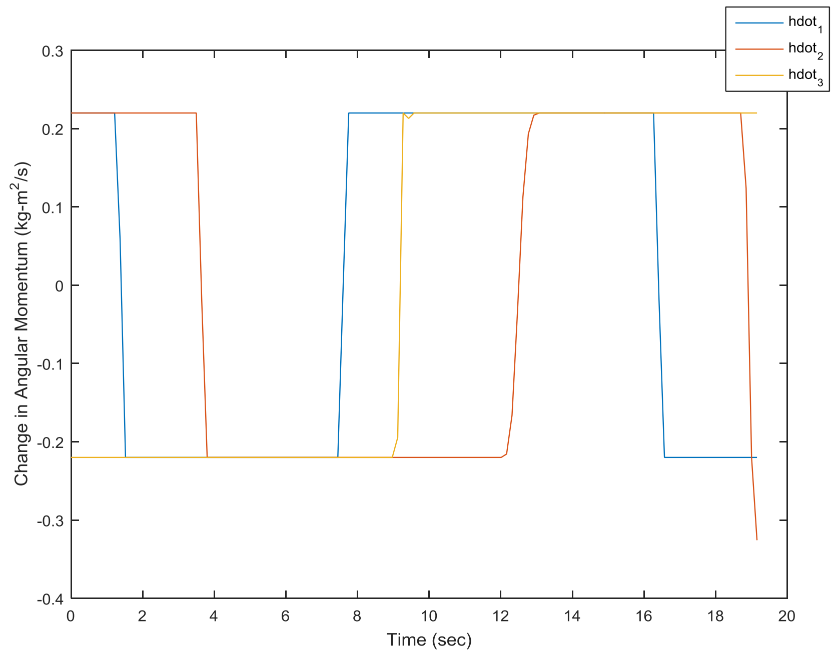

Figure 4.39: Hardware Near RTOC Controller 90 Degree Z-Axis Reorientation $\dot{\vec{h}}$ Trajectory 1

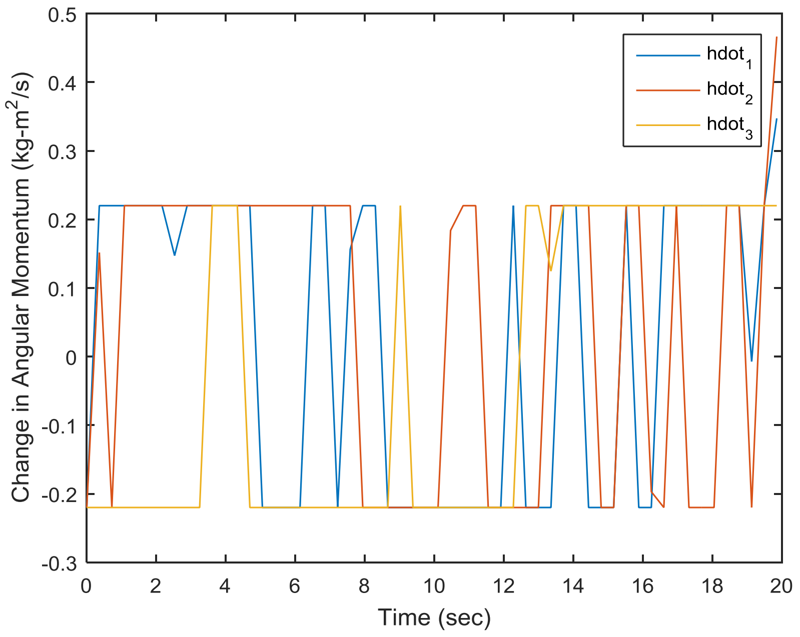

Figure 4.40: Hardware Near RTOC Controller 90 Degree Z-Axis Reorientation $\dot{\vec{h}}$ Trajectory 2

trajectory is blue, the second trajectory used (the first trajectory switched to) is red, and the third is green.

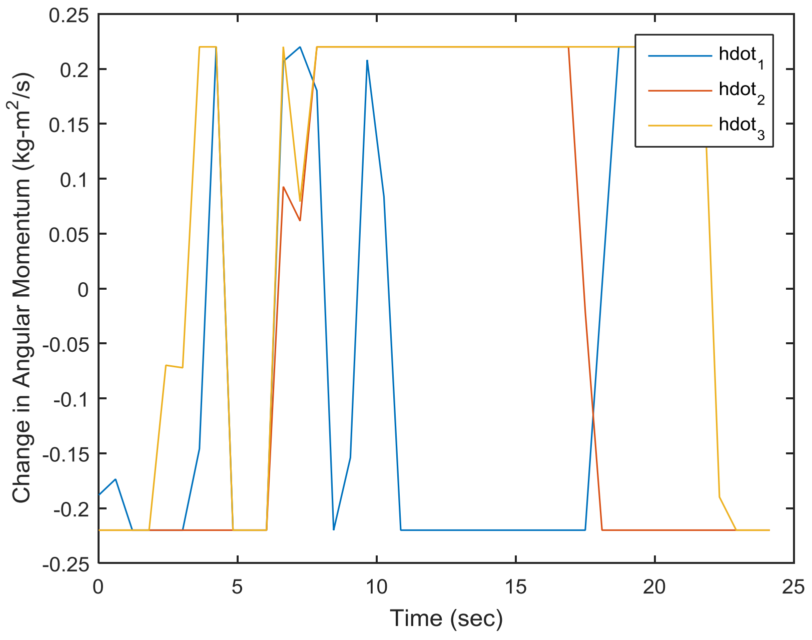

Figure 4.41: Hardware Near RTOC Controller 90 Degree Z-Axis Reorientation $\dot{\vec{h}}$ Trajectory 3

The 90 degree reorientation maneuver is completed in 23.5 seconds with 2% error at the terminus of the near RTOC portion of the maneuver. The spacecraft simulator uses three different optimal control trajectories in the maneuver, the original trajectory that was calculated before the maneuver began and used as the initial $\dot{\vec{h}}$ input and two optimal trajectories calculated on the GS PC during the maneuver. The switches occur at 6.29 seconds and 15.44 seconds respectively and can be seen on Fig. 4.35 as the quaternion parameter plots with the dashed lines. The three different $\dot{\vec{h}}$ used as inputs into the Simulink model during the maneuver are found in Fig. 4.39, Fig. 4.40, and Fig. 4.41 while all other optimal control trajectories calculated on the GS PC were discarded throughout the maneuver as an updated one was calculated. Each of these optimal trajectories resembles a "bang-bang" controller as predicted by Wie for rotation about a single axis[42]. The optimal control trajectories were calculated in an average of 2.5 seconds. Some optimal control solutions took longer than others due to the extreme nonlinearity of the spacecraft reorientation problem. The first switch is caused by error about the X-axis and Y-axis while the second switch is caused by error about the X-axis and Z-axis.

The body rates shown in Fig. 4.36 display high angular rates of 15 rad/sec or 8.6 deg/sec about all three axes demonstrating that near RTOC is able to generate torque about all three axes by using three reaction wheels throughout the maneuver as opposed to predominately just one for the PID controlled experiment. This is confirmed by the plots of reaction wheel angular rates and accelerations in Figs. 4.37 and 4.38, respectively. Note that the reaction wheel angular accelerations are the commanded accelerations $\dot{\vec{h}}$ inputs, not the accelerations read by the reaction wheel motors.

4.3.3.2 180 Degre Z-Axis Reorientation. This section discusses the results of SimSat reorienting itself 180 degrees about its Z-axis with a near RTOC controller. The plot of quaternions vs time with the solved optimal quaternion trajectories in dashed lines is found in Fig. 4.42. The plot of body angular rates vs time is found in Fig. 4.43. The plots of reaction wheel angular rates and accelerations vs time are found in Fig. 4.44 and Fig. 4.45, respectively. The $\dot{\vec{h}}$ trajectory values that were used in this maneuver are found in Fig. 4.46, Fig. 4.47, Fig. 4.48, and Fig. 4.49, .

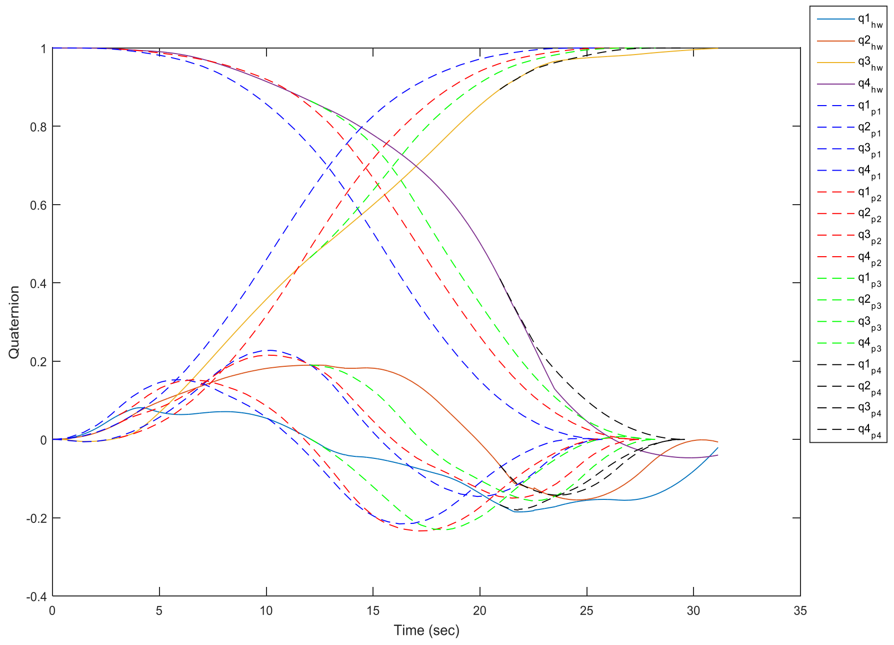

Figure 4.42: Hardware Near RTOC Controller 180 Degree Z-Axis Reorientation Quaternion Parameters

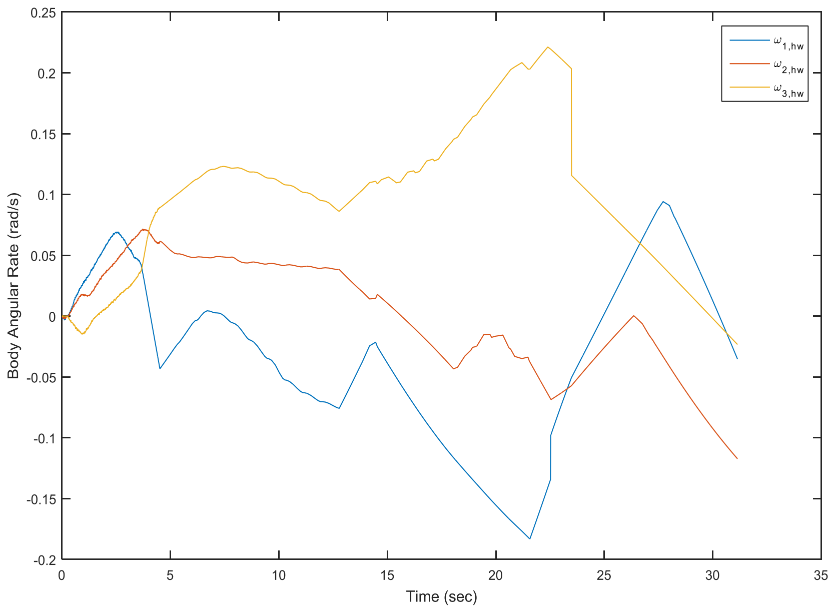

Figure 4.43: Hardware Near RTOC Controller 180 Degree Z-Axis Reorientation Body Angular Rates

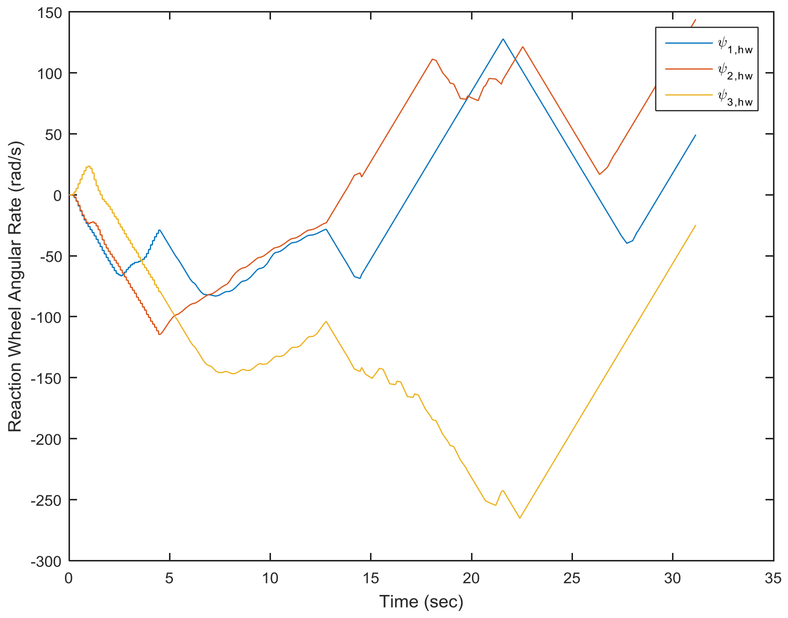

Figure 4.44: Hardware Near RTOC Controller 180 Degree Z-Axis Reorientation Reaction Wheel Angular Rates

In Fig. 4.42 the experimental quaternion parameters are shown in solid lines and the planned trajectories calculated from the optimal control solver are shown in dashed

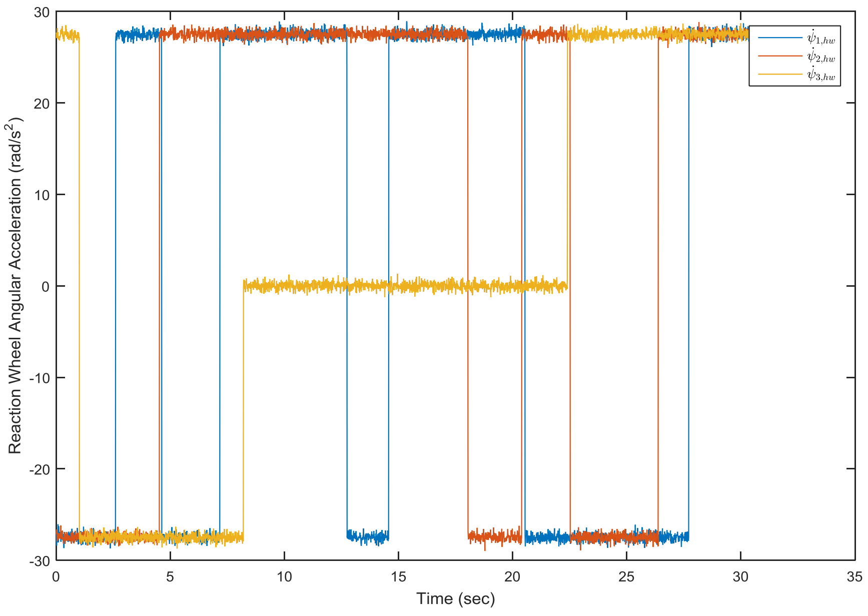

Figure 4.45: Hardware Near RTOC Controller 180 Degree Z-Axis Reorientation Reaction Wheel Angular Accelerations

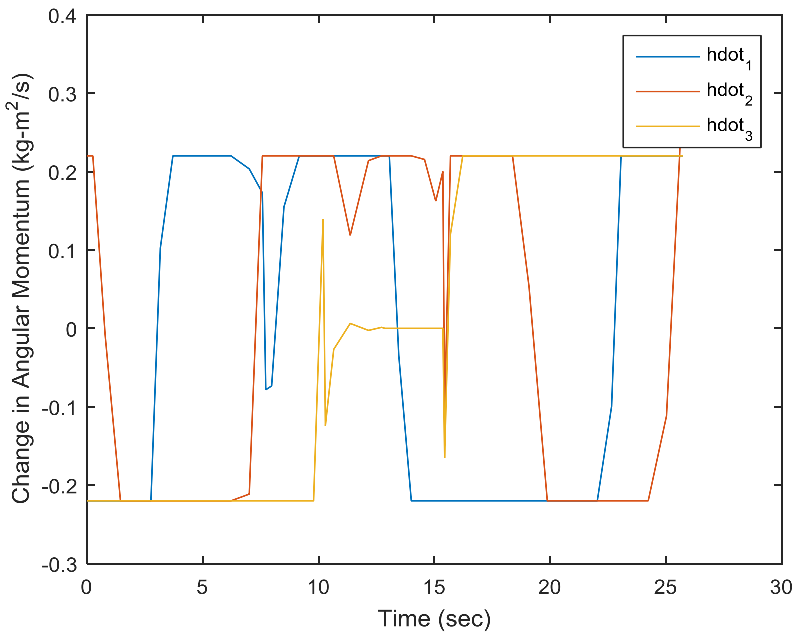

Figure 4.46: Hardware Near RTOC Controller 180 Degree Z-Axis Reorientation $\dot{\vec{h}}$ Trajectory 1

lines. The initial trajectory is blue, the 2nd trajectory switched to is red, the third is green, and the last one is black.

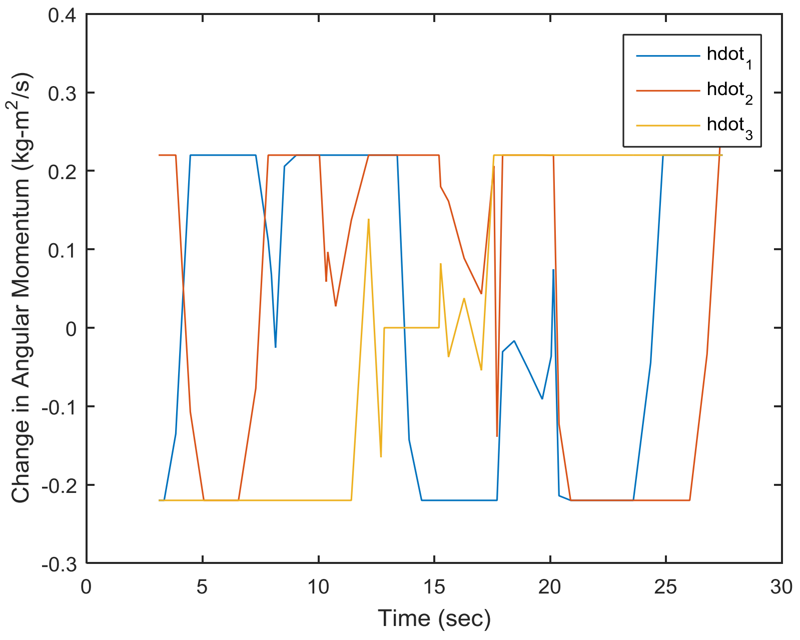

Figure 4.47: Hardware Near RTOC Controller 180 Degree Z-Axis Reorientation $\dot{\vec{h}}$ Trajectory 2

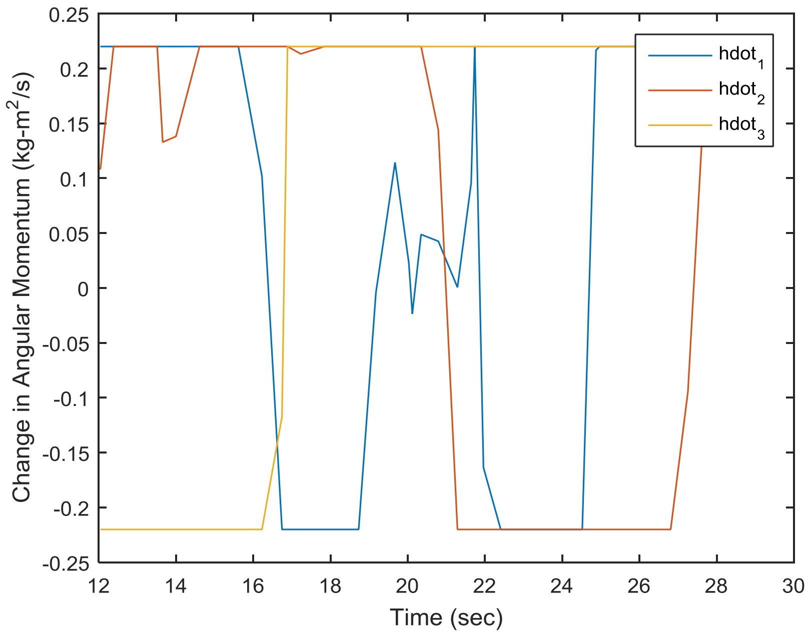

Figure 4.48: Hardware Near RTOC Controller 180 Degree Z-Axis Reorientation $\dot{\vec{h}}$ Trajectory 3

The maneuver is completed in 31.14 seconds with 7.23% error at the end of the optimal control maneuver. The plot of quaternions vs time found in Fig. 4.42 shows

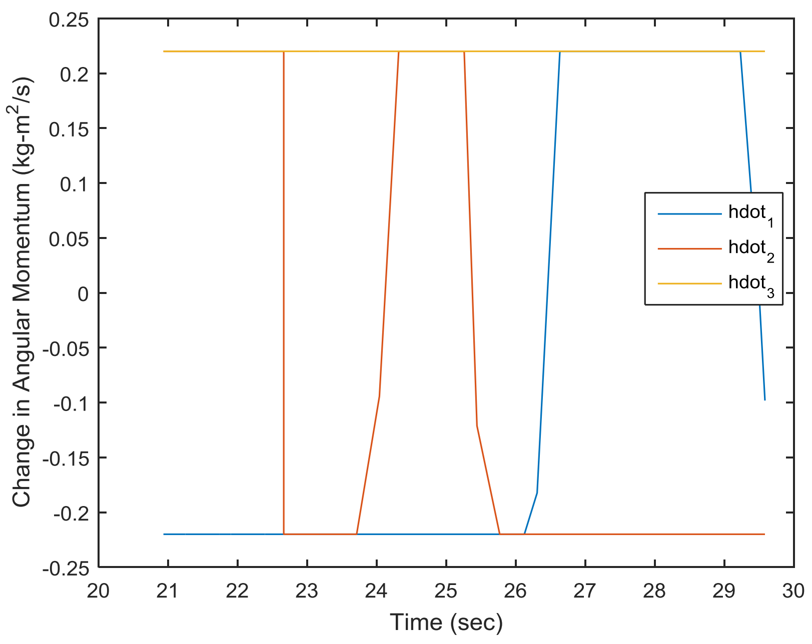

Figure 4.49: Hardware Near RTOC Controller 180 Degree Z-Axis Reorientation $\dot{\vec{h}}$ Trajectory 3

the SimSat gets off of its planned trajectory at just 4.51 seconds into the reorientation despite the fact that q_3 has not significantly deviated from its planned trajectory. The cause of this is q_1 and q_2 have significant deviations from their planned path. The error metric found in Eq. (50) takes all three quaternion parameters into account when it is calculated on the on-board PC. q_1 and q_2 deviated due to the instability about the X-axis and Y-axis. The SimSat is balanced before ever test run but the CG and CR are never collocated as they would be on orbit. The allows gravity disturbance torques to act on the spacecraft simulator and cause the SimSat to leave its planned trajectory on the X-axis and Y-axis but not the Z-axis. The near RTOC algorithm also makes trajectory switches during the maneuver at 12.76 seconds and 21.54 seconds.

The spacecraft simulator uses four different $\dot{\vec{h}}$ trajectories during the reorientation: the original one calculated prior to the start of the experiment and trajectories calculated at 3.1 seconds, 12.04 seconds, and 20.93 seconds. These trajectories resemble "bang-bang" control histories like the previous section. All other trajectories calculated throughout the maneuver are discarded as they are replaced by an updated one from the

GS PC. The only ones that are saved at the end of the maneuver are the ones used by the SimSat to reorient itself. They are found in Figs. 4.46, 4.47, 4.48, and 4.49.

The plot of the body rates about each axis vs time show that like in the previous experiment in Fig. 4.36, SimSat is able to rotate onto its side as far as its $\pm 25°$ X-axis and Y-axis limits allow it to and use the reaction wheels aligned with the X-axis and Y-axis to generate additional torque as compared to the PID controller found in Fig. 4.22 which primarily rotates about its Z-axis, the axis of rotation. The plots of reaction wheels rates and accelerations vs time in Fig. 4.44 and Fig. 4.45 support this by showing the wheels generating angular momentum with all three wheels but primarily in direction of the Z-axis.

The plots of the $\dot{\vec{h}}$ trajectories show discontinuities in all four of the plots, especially in Fig. 4.47 and Fig. 4.48. This would be corrected if the optimal control solver was able to use more collocation points but due to the increased computational power that requires it must use fewer points. This is not an issue for the hardware however as the MATLAB function *interp1* is able to use a lookup table to linearly interpolate an $\dot{\vec{h}}$ value to prevent discontinuities in the commands sent to the hardware.

Another point fo note is that the end time of the maneuver increases for each calculated optimal trajectory. The original optimal path is the time-optimal maneuver to take the SimSat from its original orientation to its final one. Any other maneuvers may be time-optimal for that state and time given by the on-board PC but are not time-optimal for the maneuver. This causes the completion time of the reorientation to increase with each new trajectory computed by the optimal control solver.

4.4 Comparison and Analysis

This section covers analysis of the 90 and 180 degree Z-axis reorientations of SimSat using a PID controller, an OLOC controller, and a near RTOC controller and variations of the optimal configuration of the near RTOC controller. The first section discusses the differences between the three controllers and the advantages and disadvantages of the near RTOC approach. The second section covers the effects of varrying the “inner” loop of the near RTOC algorithm’s error metric to determine when to switch trajectories. The third section presents the effect of changing various parameters of the optimal control solver has on the near RTOC controlled system and the optimal control solutions generated.

4.4.1 PID vs Open Loop vs Near RTOC. In this section, we will discuss the three different controllers presented and their hardware results with an emphasis on the near RTOC controller as that is the primary focus of this research.

There are two performance characteristics used in this research to determine the effectiveness of the different control schemes. The first is the time to complete the maneuver, defined for the PID controlled simulation and hardware experiment as the time from the start of the experiment to within 1% error of the final desired quaternion and for the OLOC controlled simulation and experiment and the near RTOC experiment to be from the start of the experiment of simulation to the time where the optimal control portion terminates regardless of the pointing error at that time. The second characteristic is the pointing error at the end of the optimal control portion of the experiment for the OLOC and near RTOC experiments as calculated in Eq. (50). The PID controller is assumed to have 0 error at the end of its maneuver. These performance characteristics are discussed further in Sec. 3.6.

In Sec. 4.2, simulated results of a 90 degree and 180 degree Z-axis rotation are discussed. Likewise, in Sec. 4.3, the hardware results of the same two maneuvers are presented. The performance metrics for the four simulations and six hardware test runs are presented in Table 4.1.

There are a number of observations that can be made from these results.

The first obersvation is the difference in time to complete the maneuver between the simulation and hardware results for the 180 degree maneuvers. This is a maneuver that the PID controller and the OLOC controller are ill-suited for due to the small angle assumption no longer being true and the extended time of the maneuver compared to the 90 degree maneuvers allowing disturbance torques, errors in state measurement, and other issues to compound themselves. The OLOC controller is especially effected as the simulation predicted a time to completion of 20.94 seconds while the actual hardware

Table 4.1: Performance Metrics for Simulated and Hardware Test Results

Type	Controller	Maneuver (deg)	Time to Complete (sec)	% Error Metric at End
Simulated	PID	90	29.9	0
Simulated	PID	180	45.4	0
Simulated	OLOC	90	16.46	0
Simulated	OLOC	180	20.94	0
Hardware	PID	90	30.3	0
Hardware	PID	180	49.9	0
Hardware	OLOC	90	19	18.9
Hardware	OLOC	180	26.36	14.8
Hardware	Near RTOC	90	23.5	2
Hardware	Near RTOC	180	31.14	7.2

performed the maneuver with 14.8% error in 26.36 seconds. Some of this can be explained by the spacecraft simulator avoiding a collision but a greater factor is the gravitational disturbances as SimSat tilts on its side to generate torque from the reaction wheels aligned with the X-axis and Y-axis. Additional analysis found that an OLOC trajectory that took into account the limits on SimSat's X-axis and Y-axis did not complete the maneuver. This could perhaps be a timing issue on the reaction wheels or an interpolation issue with determining the correct $\dot{\vec{h}}$ values to determine reaction wheel rates.

A general trend in the results is that the PID controller is more accurate than the OLOC controller while the OLOC controller is able to complete the 90 degree maneuver and especially the 180 degree reorientation maneuver in a shorter amount of time. This is a function of the amount of disturbance torques affecting the maneuver. This difference between the PID and OLOC controllers comes as a result of the PID's abilty to use feedback to take state information from the attitude determination system, in this case the Northrop Grumman LN-2000 Laser Gyroscope, and use that information as part of the next $\dot{\vec{h}}$ command to the reaction wheels. It is able to respond to disturbances such as gravity disturbance torques and counteract them with additional torque from the actuators. The OLOC system is "fire and forget." It cannot be changed once the experiment begins. However, OLOC does have the advantage in the end time performance parameter. It is able to use additional torque from all 3 reaction wheels to go from its initial orientation to its final orientation faster than the PID controlled system which primarily only uses one reaction wheel. In these experiments the PID controlled experiments primarily use the Z-axis reaction wheel while the OLOC controlled system uses all 3 wheels. In the PID simulation the other two wheels are accelerated slightly at the beginning of the maneuver but quickly decelerate to allow the Z-axis wheel to complete the reorienation.

The near RTOC system is able to combine the benefits of PID's increased accuracy at the end of the reorientation maneuver and OLOC's increased speed to complete the maneuver. Throughout the first part of the reorientation maneuver, the near RTOC controlled system appears to be closer to the OLOC system in quaternion trajectory as shown in Fig. 4.35 and Fig. 4.42. The first switch (or two in the case of the 180 degree reorientation) appear to be course corrections while the final switch appears to be more of a PID or feedback trajectory to bring the spacecraft back to its target. The near RTOC controlled system is able to bring the system from its initial orientation to its final orientation in 23.5 seconds in the case of the 90 degree Z-axis reorientation and 31.4 seconds in the 180 degree Z-axis reorientation with 2% and 7.23% error at the end of the maneuver respectively. It does not continue to the exact final orientation as the error metric at the end of the optimal trajectory is not enough to switch to a new trajectory; if this was the case the algorithm would request and move to a new optimal trajectory. The best-case scenarios as calculated by the optimal control solver GPOPS-II are 16.46 and 20.94 seconds for the 90 degree and 180 degree maneuvers respectively with no disturbance or other external torques on the system. In the experimental cases these times are 19 and 26.36 with the additional time due to differences in the model. The model in this research does not take into account any external torques on the system such as air drag.

The near RTOC controller is unable to reach the time-optimal trajectories and the resulting maneuver end time calculated in Sec. 4.2.2.1 and Sec. 4.2.2.2 for a 90 degree Z-axis reorientation and a 180 degree Z-axis reorientation, respectively. After the first time the near RTOC algorithm switches to a new optimal trajectory, it is no longer time-optimal for that manuever from the time the SimSat starts its reorientation to the end time but it is optimal for the case of its current orientation slewing to the final orientation. The time-optimal case is found in Sec. 4.3.2.1 and Sec. 4.3.2.2 to be faster than near RTOC but has more terminal error.

The performance of the PID controller is extremely dependent on the gains used for K_P, K_I, and K_D. Gain settings that produce a time-optimal response for one maneuver may not produce a time-optimal response in another maneuver. The gain matrix used in this research was constant throughout all simulations and experiments. While the near RTOC controller presented is more time-optimal than the presented PID controller for the 90 and 180 degree Z-axis reorientations, this may not be the case for another PID controller with different gains. Additional research is needed to determine if the near RTOC controller presented here is more time-optimal than a PID controller tuned to decrease the time needed to complete the maneuver.

The cost of using near RTOC is increased processing power. The optimal control solver on the GS PC was able to compute solutions every 2.49 seconds in both maneuvers. This is much slower than the 20 Hz sampling rate of the controller on the hardware. As can be seen in Sec. 4.4.3 this can be improved slighly by decreasing the number of collocation points but it is limited by the GS PC's lack of addressable memory. A true real-time optimal control scheme would have to be able to calculate a new optimal trajectory faster than the sampling rate of the controller and its actuators and sensors to reach the desired time-optimal trajectory.

Overall, the near RTOC controller was able to reorient itself in less time than the PID controller with the gains found in this research and have more terminal accuracy than the OLOC controller. However, it is unable to reach the true "optimal" trajectory generate by the optimal control solver. More research is needed to bring the current near RTOC implementation closer to the true optimal solution.

4.4.2 Variation of Near RTOC "Inner" Loop Error Parameter. This section discusses the effect changing the "inner" loop error parameter in the near RTOC implementation has on the speed and accuracy of SimSat performing a 90 degree reorientation maneuver.

This is the metric by which the Near RTOC algorithm on the on-board PC knows whether to stay on its current trajectory and control history or request, receive a new trajectory and control history, and implement the new control history in that timestep. If the scalar quaternion error from the current trajectory is lower than the error parameter SimSat stays on its current optimal trajectory. If the scalar quaternion error is greater than the error parameter the new trajectory is implemented.

Fig. 4.50 shows the effect changing this parameter has on q_3, the quaternion parameter associated with the Z-axis which in this maneuver is the axis of rotation. Error parameters of 0.05, 0.1, and 0.5 are used with 0.1 being the error parameter used in the previous section for the near RTOC 90 degree Z-axis reoreintation maneuver.

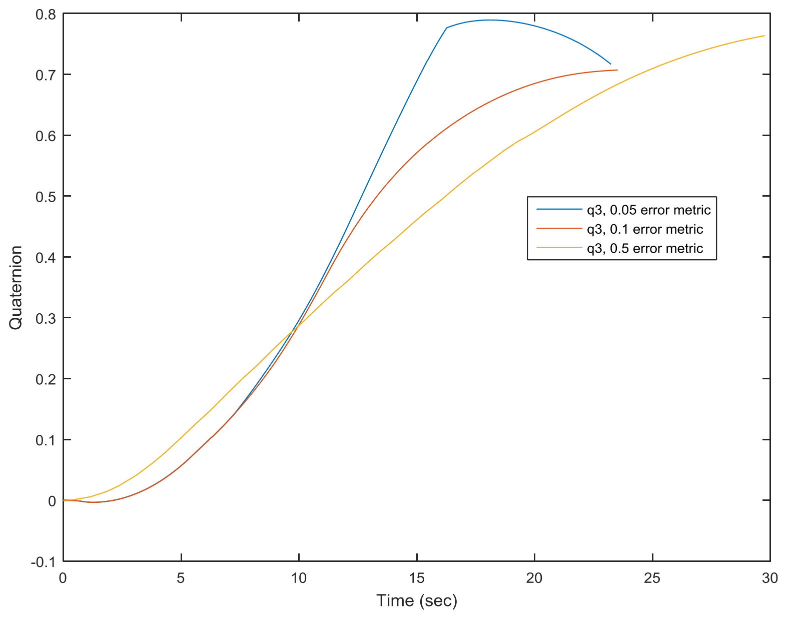

Figure 4.50: Hardware q_3 for a 90 Degree Reorientation Maneuver For Various Inner Error Metrics

The number of switches performed by the near RTOC algorithm for a 90 degree Z-axis reorientation are 3 for the 0.01 case, 2 for the 0.1 case, and 1 for the 0.5 case. The 0.01 case completes the reorientation in 23.24 seconds, the baseline 0.1 case in 23.5 seconds, and the 0.5 case in 29.75 seconds. The 0.01 case has 5.3% error at the end of its optimal maneuver, the 0.1 case has 2% error, and the 0.5 case has 10.42% error.

The 0.01 case performs the manuever faster than the other two error metric values but does has additional terminal error. This error is due to a switch at 15.14 seconds when the spacecraft simulator is almost at the desired orientation. The motors on the reaction wheels do not perform well when the rates are quickly changed and cause SimSat to overshoot its target q_3 of 0.707, faster than was anticipated by the planned optimal trajectory, and have to come back to the target. This shows that feedback in near RTOC is a large advantge over open-loop solutions as the spacecraft simulator's implementation of the near RTOC algorithm is able to correct for an error at the rate of the inner error metric.

The 0.5 case only switches once during the maneuver. It is still faster and more accurate than an open loop implementation but is slower and less accurate than the experiments with the other two error metric values.

The inner error metric is related to the rate at which optimal control solutions are calculated on the GS by the optimal control solver. The near RTOC algorithm only requests a new trajectory when the inner error metric is crossed. If the inner error metric continually is crossed by error in the quaternion parameter trajectory but a new solution has not been computed by the optimal control solver, the algorithm will continue to request and receive the same optimal control trajectory that was most recently computed. This trajectory may or may not be the most time accurate at the time of the switch. This issue can be improved by increasing the rate at which optimal control trajectories are computed.

Different equations used to calculated this inner error metric are not presented in this research. More research can be done in this area to optimize the "inner" loop's error metric. This is discussed further in Sec. 5.2.2.

4.4.3 Optimal Control Solver Parameter Variation. The three settings in the optimal control solver GPOPS-II that have the greatest effect on solution accuracy and time needed to compute the solution are number of collocation points, mesh error tolerance, and number of iterations needed to compute an optimal solution. This section will discuss the changes in the speed of the optimal control solution when these parameters are modified independently.

The default settings used in this research for mid-experiment calculations of new optimal control solutions are 1×10^{-4} for mesh tolerance, 10 for number of iterations, and 20 for number of collocation points. These values were chosen through iteration to determine a set of parameters for GPOPS-II that produced an optimal control solution in less than 5 seconds but that had enough collocation points that a complete quaternion and $\dot{\vec{h}}$ trajectories could be linearly interpolated. Throughout the two experiments of near RTOC this translated to an average computational time from receiving the state information to outputting a new optimal control solution of 2.49 seconds. The state information between the first switch and the second switch on the 90 degree near RTOC reorientation was saved as initial conditions for the optimal control solver GPOPS-II and the final state information was the same as found in Eq. (48). Each of these parameters listed above was varied independently while the other parameters were held at the default values. No information about the accuracy of the optimal solution was determined and all calculations were performed on the same GS PC used in this research.

Table 4.2 shows the effect the number of collocation points has on the time to compute an optimal solution. As can be seen in the table, computation time increases with the number of collocation points. Changing this parameter to 1000 or more takes almost as long to compute a solution as the reorientation maneuver itself. This would introduce a great deal of phase lag as the spacecraft simulator is likely not where the near RTOC controller expects it to be as it implements the new trajectory. This solution is likely more accurate but due to the Bellman Optimality Principle discussed in Sec. 2.4.3 it is more effective to use a small set of nodes and update more frequently as proved by Ross *et al.* [32].

Table 4.3 shows the effect of changing the mesh tolerance. Decreasing the mesh tolerance also has a negative impact on the speed of the optimal control solver. This parameter change also increases the accuracy of the optimal control solution due to the increased accuracy of the solution on each mesh but as the solution gets more accurate

Table 4.2: Time to Compute Solution - Change in Number of Collocation Points

Collocation Points	Time (sec)
10	1.94
20	2.49
50	4.57
100	7.60
200	10.11
500	11.43
1000	24.01

more time is needed to compute it. The setting chosen for this research seems to be the optimal one to not increase too much of a delay to the system.

Table 4.3: Time to Compute Solution - Change in Mesh Tolerance

Mesh Tolerance	Time (sec)
10^{-3}	2.24
10^{-4}	2.49
10^{-5}	5.68
10^{-6}	35.61
10^{-7}	109.64
10^{-8}	304.1

Table 4.4 shows the effect of changing the number of mesh iterations. The number of mesh iterations does not appear to have much of an affect for this optimal control solution. Less than 10 did not result in a solution being found by the solver while any number greater than 10 did not change the speed at which the solution was calculated. This is due to the number of iterations needed to calculate the optimal solution being somewhere between 5 and 10, in this case it is 8. In a different case when more than 8 meshes are needed to calculate a solution this may change the impact that the number of mesh iterations has on the computational time but in this case changing the number of mesh iterations has no impact.

4.5 Summary

This chapter covered the simulated and experimental results of this research using the methods found in Ch. III. The first section discussed the simulated 90 degree and

Table 4.4: Time to Compute Solution - Change in Number of Mesh Iterations

Number of Iterations	Time (sec)
5	N/A
10	2.49
15	2.49
25	2.49
50	2.49
100	2.49

180 degree reorientation of SimSat about its Z-axis using a PID controller and an OLOC controller. The second section presented the hardware results of the same two maneuvers using a PID controller, an OLOC controller, and a near RTOC controller, and compared these results to the simulated results in the previous section. The third section discusses the analysis of these results as well as variations of inner loop and optimal control solver parameters for the near RTOC controller.

V. Conclusions and Recommendations

5.1 Conclusions

The primary objective of this research is to implement a near RTOC controller on the SimSat hardware and determine its performance relative to PID and OLOC controllers for spacecraft. The equations needed to develop a near RTOC algorithm were outlined and the state equations needed for an optimal control solver, along with a brief history of the spacecraft time-optimal reorientation problem, were presented in Chapter II. A PID controller and an open loop optimal control scheme, as well as a near RTOC algorithm and implementation, were developed in Chapter III. In Chapter IV, in simulation and on the SimSat hardware, two different reorientation maneuvers were conducted for these control methods, a 90 degree Z-axis reorientation and a 180 degree Z-axis reorientation. The accuracy and speed at which the different control methods complete the maneuvers were analyzed and compared. The "inner" loop error metric in the near RTOC implementation was varied in order to determine its effect on the effectiveness of the near RTOC solution, as was the number of collocation points in the optimal control solver.

Near RTOC demonstrates some advantages over PID and OLOC but has its disadvantages as well. The time to complete the two reorientations for near RTOC is less than the PID controller used. However, as mentioned in Sec. 4.4.1 the gains for the PID controller are constant throughout the research. A tuned PID controller may be more effective than the near RTOC approach presented. This near RTOC controller is more time-optimal than this PID controller but this may not be the case for all reorientations or for a tuned PID controller. The true optimal control trajectory found in simulation in Sec. 4.2.2.1 and Sec. 4.2.2.2 is not realized. The OLOC controller is faster in both reorientations than the near RTOC controller but has significant terminal error of over 10% about its X-axis and Y-axis at the end of the maneuver. In this case, the near RTOC controller is more accurate with terminal errors of under 10% for both maneuvers. However, in a true torque-free environment, this will not be the case as the errors are caused by gravity disturbance torques acting on the SimSat.

The purpose of near RTOC is to bring a form optimal control to an environment with disturbances. This research shows that this is possible; however, the gains realized may not be worth the effort expanded to implement near RTOC outside of the laboratory environment. A spacecraft that is constantly slewing 90 degrees or more may see some advantages but another spacecraft that rarely rapidly changes its orientation may not find near RTOC worthwhile.

This research shows the promise of near RTOC but still needs additional work. The near RTOC algorithm is spread out over two PCs. This is not ideal and should be improved upon before future work is conducted. Recommendations are made in the following section as to how the on-board PC could be upgraded and made to be able to compute optimal control solutions instead of having the GS PC compute the solutions and transfer via UDP to the on-board PC.

The next step for this research is to implement a near RTOC solution on flight hardware. GPOPSII, the optimal control solver used in this research, is dependent on MATLAB. Most ADCS on satellites do not run MATLAB or have the computational resources to run such a computationally intensive program. Many changes must be made to the work done in this research, including rewriting most of the near RTOC code in C or another low-level programming language, in order to have the near RTOC implementation described here work on an actual satellite.

5.2 Future Work and Recommendations

This section discusses suggestions for future work and research in the fields of Near RTOC, spacecraft control, and improvements to the SimSat II testbed. The first section covers upgrades to the SimSat testbed and the second section discusses future work to be researched in Near RTOC and spacecraft control.

5.2.1 Upgrades to SimSat. This section covers upgrades to the SimSat testbed in order to improve it both in order to conduct near RTOC research and as a general spacecraft attitude dynamics and control testbed. The largest limitation in this research is the inability of the Windows XP-based MiniPC on-board SimSat to calculate optimal control solutions using GPOPSII. This greatly complicates the optimal control implementation by necessitating the need for control data to be transferred between PCs. The processor and memory on the on-board PC are insufficient to solve the optimal control problem in anything approaching a reasonable amount of time. An optimal control solution using the equations in this research that may take a minute or two to solve on a traditional desktop computer such as the GS PC takes hours on the on-board PC. This issue could be remedied by upgrading the on-board computer to a modern Windows 7-based system running an Intel Atom or comparable multicore processor and with additional addressable RAM. This upgrade would negate the need for the GS PC to receive the state information from the on-board PC, calculate a new optimal control solution, and send it back to the on-board PC for implementation. This would greatly improve the performance of the near RTOC implementation on SimSat as well as any other control research conducted.

Some integration work would have to be accomplished with the dSPACE MicroAutoBox but this would not preclude the on-board PC from being replaced.

Another upgrade that could be made is to upgrade the GS PC from the Windows XP 32-bit OS to a Windows 7 64-bit installation. While there are 8 GB of RAM installed on the GS PC, only 2 GB are seen in the operating system. 32-bit operating systems such as the one currently installed are only able to address a maximum of 2 GB of RAM. Since the primary purpose of the GS is to use the Remote Desktop feature to connnect wirelessly to the on-board PC during an experiment, this would be a low-hassle upgrade that would increase the processing power of the GS PC as GPOPS-II's processing speed for an optimal control solution is partially dependent on the amount of RAM available to MATLAB.

Another upgrade to the SimSat testbed is the PhaseSpace motion capture system that is intended for installation and integration with SimSat in the Spring of 2016. This system will provide a "truth" orientation to SimSat that is not currently available. The LN-2000 fiber optic gyroscope currently installed on SimSat is accurate but has no dynamic external "truth" orientation to be compared against. The PhaseSpace system will provide greater accuracy for future control system testing.

5.2.2 Future Research. An area of future research in the area of Near RTOC is to add a filter to the measurement data received from the sensors before it is inputted as initial conditions in the optimal control solver. Currently there is no filtering of the data received by the sensors during the maneuver or in post-processing. Any noise or other errors in the system are carried through to the optimal control problem or to the data collected. A simple moving average filter or a more capable one such as a variant of the Kalman filter should be added to the existing Near RTOC algorithm and implementation in order to increase the accuracy of the angular acceleration measurements of the LN-2000 gyroscope and the angular rates and quaternion parameters that are integrated from those measurements. The increase in processing power as suggested in Sec. 5.2.1 could offset the increased computational requirements that filtering the data during the maneuver would require.

The PID controller could have its gains tuned in order to produce a time optimal response. The gains should be tuned for the two different maneuvers. This would produce a "best-case" PID controller that the near RTOC algorithm could be compared against.

The error metric in the "inner" loop was not changed much after its implementation other than the values that it uses as a threshold to switch trajectories. No effort was made in this research to determine whether the orientation defined by the quaternion parameters, Euler angles, or the body rates was the most effective "inner" loop metric. The answer may be one or the other or some combination of, or perhaps the error metric presented in this research is sufficient with some tweaking. Each quaternion parameter was weighed the same, this could be changed to have different weights for different maneuvers depending on the axis of rotation. Another thought along this line is to determine whether one $\dot{\vec{h}}$ should be completely replaced by a new one from the optimal control solver or only certain axes need to be.

Another area for future research is optimizing the GPOPS-II settings of number of collocation points, mesh tolerance, and number of iterations. This research covers the effect of changing these properties independently has on the time it takes to compute an optimal control solution but does not take into account what the optimum settings are for near RTOC nor the effect changing one setting has on another. A static optimization problem could be formulated and solved using nonlinear programming techniques to determine the optimal settings for the time-optimal reorientation problem. Another parameter that could be optimized is the timeout value when the near RTOC algorithm decides to use an old value to compute the new optimal solution.

Different maneuvers could be conducted to test near RTOC's effectiveness for other maneuvers. For the sake of time, this research only covered large angle maneuvers about one axis. Future work could try to have a target orientation about more than just the Z-axis or small angle maneuvers. Eventually the research should go in the direction of multiple large angle maneuvers in order to test the ability of near RTOC to conduct simultaneous maneuvers that stress the ADCS of the spacecraft simulator.

Finally, additional actuators could be added to the near RTOC implementation discussed in this research in order to better simulated an on-orbit spacecraft that uses more than reaction wheels to reorient itself. The equations of state used in GPOPS-II and the Simulink model could be altered to add the CMGs, the fans, or both instead of or in addition to the reaction wheels.

Bibliography

1. Jasbir S. Arora. *Introduction to Optimum Design.* Elsevier Academic Press, second edition, 2004.

2. Nazareth Bedrossian, Sagar Bhatt, Mike Lammers, Louis Nguyen, and Yin Zhang. First Ever Flight Demonstration of Zero Propellant Maneuver^{TM} Attitude Control Concept. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 2007.

3. John T. Betts. Survey of numeric methods for trajectory optimization. *Journal of Guidance, Navigation, and Control*, 21(2):193–207, mar 1998.

4. Karl D. Bilimoria and Bong Wie. Time-Optimal Three-Axis Reorientation of a Rigid Spacecraft. *Journal of Guidance, Control, and Dynamics*, 16(3):446–452, 1993.

5. Richard Boynton. Using A Spherical Air Bearing to Simulate Weightlessness. In *55th Annual Conference of the Society of Allied Weight Engineers*, June 1996.

6. Richard Boynton. How Mass Properties Affect Spacecraft Attitude Control. In *67th Annual Conference of the Society of Allied Weight Engineers*, May 2008.

7. Arthur E. Bryson. *Dynamic Optimization.* Addison-Wesley, 1999.

8. S. Cho, J. Shen, N. M. McClamroch, and D. S. Bernstein. Equation of Motion for the Triaxial Attitude Control Testbed. In *Proceedings of the Conference on Decision and Control*, December 2001.

9. Alfio Maria Quarteron Thomas A. Zang Jr Claudio Canuto, M.Yousuff Hussaini. *Spectral Methods in Fluid Dynamics.* Springer-Verlag Berlin Heidelberg, 1988.

10. J. E. Colebank, R. D. Jones, G. R. Nagy, R. D. Pollak, and D. R. Mannebach. SIMSAT: A Satellite Simulator and Experimental Test Bed for Air Force Research. In *Proceedings of the AIAA Space Technology Conference and Exposition*, September 1999.

11. Vincent J. Dabrowski. *Experimental Demonstration of an Algorithm to Detect the Presence of a Parasitic Satellite.* M.S. Thesis, Air Force Institute of Technology, 2003.

12. S. S. F. de Cordova and D. B. DeBra. Mass Center Estimation of a Drag-Free Satellite. In *Proceedings of the 6th Triennial World Congress of the International Federation of Automatic Control*, August 1975.

13. Matthew Dillsaver. Mech 620: Systems optimization. Air Force Institute of Technology Course Notes, 2015.

14. Cole C . Doupe. *Optimal Attitude Control of Agile Spacecraft Using Combined Reaction Wheel and Control Moment Gyroscope Arrays.* Dissertation, Air Force Institute of Technology, 2015.

15. J.R. Etter. A solution of the time-optimal euler rotation problem. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, volume 2, pages 1441–1449. AIAA, 1989.

16. Bengt Fornberg. *A Practical Guide to Pseudospectral Mehtods.* Cambridge University Press, 1995.

17. David B. French. Hybrid Control Strategies for Rapid, Large Angle Satellite Slew Maneuvers. M.S. Thesis, Air Force Institute of Technology, 2003.

18. Warren C. Grunwald. Design of a programmable star tracker-based reference system for a simulated spacecraft. Master's thesis, Air Force Institute of Technology, 2014.

19. W. Haeussermann and H. Kennel. A Satellite Motion Simulator. *Astronautics*, 5:22–23, 90–91, 1960.

20. David G. Hull. *Optimal Control Theory for Applications.* Springer, 2003.

21. Mark Karpenko I. Michael Ross. A review of pseudospectral optimal control: From theory to flight. *Annual Reviews in Control*, (36):182–197, 2012.

22. James D. Turner John L. Junkins. *Optimal Spacecraft Rotational Maneuvers.* Elsevier, first edition, 1986.

23. Donald E. Kirk. *Optimal Control Theory: An Introduction.* Courier Corporation, 2004.

24. Warren M. Macek and Jr. D. T. M. Davis. Rotation rate sensing with traveling-wave ring lasers. *Applied Physics Letters*, 2:67–68, 1963.

25. Christopher G. McChesney. Design of attitude control actuators for a simulated spacecraft. M.s. thesis defense, Air Force Institute of Technology, 2011.

26. C. Douglas McFarland. Near real-time closed-loop optimal control feedback for spacecraft. Master's thesis, Air Force Institute of Technology, 2009.

27. David J. McGill and Wilton W. King. *Engineering Mechanics: An Introduction to Dynamics*. PWS Publishing Company, third edition, 1995.

28. Anil V. Rao Michael A. Patterson. *Users Manual for GPOPS-II*, 2.0 edition, may 2014.

29. Jorge Padro. Development of a star tracker-based reference system for accurate attitude determination of a simulated spacecraft. Master's thesis, Air Force Institute of Technology, 2012.

30. Neal Roach, Wayne Rohe, and Nate Welty. A Systems Engineering Approach to The Design of a Spacecraft Dynamics and Control Testbed. M.S. Thesis, Air Force Institute of Technology, 2008.

31. Neal Roach, Wayne Rohe, and Nate Welty. A Systems Engineering Approach to The Design of a Spacecraft Dynamics and Control Testbed. M.S. Thesis Defense, Air Force Institute of Technology, 2008.

32. I. Michael Ross, Qi Gong, and Pooya Sekhavat. The Bellman Pseudospectral Method. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, August 2008.

33. I. Michael Ross and Mark Karpenko. A review of pseudospectral optimal control: From theory to flight. *Annual Reviews in Control*, 36(2):182 – 197, 2012.

34. I. Michael Ross, Pooya Sekhavat, Andrew Fleming, and Qi Gong. Pseudospectral Feedback Control: Foundations, Examples and Experimental Results. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 2006.

35. Jana L. Schwartz, Mason A. Peck, and Christopher D. Hall. Historical Review of Spacecraft Simulators. *American Astronomical Society*, 2003.

36. Jerry Jon Sellers. *Understanding Space*. McGraw Hill, third edition, 2005.

37. Marcel J. Sidi. *Spacecraft Dynamics & Control*. Cambridge University Press, 1997.

38. Jason E. Smith. Attitude Model of a Reaction Wheel/Fixed Thruster Based Satellite Using Telemetry Data. M.S. Thesis, Air Force Institute of Technology, 2005.

39. Ryan E. Snider. Attitude Control of a Satellite Simulator Using Reaction Wheels and a PID Controller. M.S. Thesis, Air Force Institute of Technology, 2003.

40. J.L. Junkins S.R. Vadali, L.G. Kraige. New results on the optimal spacecraft maneuver problem. *Journal of Guidance, Control, and Dynamics*, 7(3), 1984.

41. Eric D. Swenson. Mech 632: Intermediate spacecraft dynamics. Air Force Institute of Technology Course Notes, 2015.

42. Bong Wie. *Space Vehicle Dynamics and Control*. American Institute of Aeronautics and Astronautics, Inc., second edition, 2008.

43. William E. Wiesel. *Spaceflight Dynamics*. Aphelion Press, third edition, 2010.

44. Jonathan W. Wright. *Advancements of In-Flight Mass Moment of Inertia and Structural Deflection Algorithms for Satellite Attitude Simulators*. Dissertation, Air Force Institute of Technology, 2015.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704–0188*

1. REPORT DATE (DD–MM–YYYY) 24 March 2016	2. REPORT TYPE Master's Thesis	3. DATES COVERED (*From* — *To*) August 2014 – March 2016

4. TITLE AND SUBTITLE Analysis of a Near Real-Time Optimal Attitude Control for Satellite Simulators	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Patrick, Ryan M. 1st Lt	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 2950 Hobson Way WPAFB OH 45433-7765	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-ENY-MS-16-M-232
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally Left Blank	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Statement A. Approved for Public Release;Distribution Unlimited

13. SUPPLEMENTARY NOTES
This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Dynamic optimization of spacecraft attitude reorientation maneuvers can result in significant savings in attitude determination and control system size, mass, and power. Optimal control theory is generally applied using an open loop trajectory which is vulnerable to disturbances. A closed loop implementation of optimal control has been difficult to achieve due to the computational requirements needed to quickly compute solutions to the optimal control problem. This research focuses on evaluating a near real-time optimal control (RTOC) system for large angle slew maneuvers on the Air Force Institute of Technology's spacecraft simulator called SimSat. A near RTOC algorithm computes optimal control solutions at a rate of 0.4 Hz using a pseudospectral-based solver. The solutions or trajectories are then resampled at a fixed time step of 100 Hz and fed forward to a closed loop on SimSat. This algorithm is developed and tested on the hardware and compared to simulated and hardware results of a proportional-integral-derivative (PID) controller and an open loop optimal control controller for 90 degree and 180 degree Z-axis rotations. The benefits of decreased time to complete the maneuver and increased accuracy at the end of the optimal maneuver are shown to be improvements over traditional over PID control and open loop optimal control.

15. SUBJECT TERMS
Spacecraft attitude optimal control

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Dr. Eric Swenson
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	UU	123	19b. TELEPHONE NUMBER (Include Area Code) (937)-255-3636

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18