CONTROL OF THE ATTITUDE SYSTEM FOR
THE UMR SAT PROJECT

By

DAVID RANDALL WALKER

A THESIS
Presented to the Faculty of the Graduate School of the
UNIVERSITY OF MISSOURI-ROLLA
In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN AEROSPACE ENGINEERING

May 2007

Approved by

Dr. Henry J. Pernicka, Advisor                  Dr. Robert G. Landers

Dr. K. Krishnamurthy
This thesis presents an attitude control model for the UMR SAT project. The UMR SAT project is a student design project at the University of Missouri-Rolla. This project involves the study of two satellites flying in coupled flight as well as formation flight. The control model includes a guidance law using Lyapunov stability principles and a control logic algorithm with the use of both a magnetic coil actuator and thrusters with a throttle filter. A hybrid controller using both magnetic coils and thrusters is also discussed. The magnetic coil controller and thruster controller were synthesized using a dynamic model without perturbations.
ACKNOWLEDGMENTS

I would like to thank the University of Missouri-Rolla for providing a place of growth and knowledge as I obtained a quality engineering education. It was an honor to work with so many outstanding professors and students from a variety of academic backgrounds. I would especially like to thank Dr. Pernicka for being my advisor and a first-class professor. I would also like to thank Dr. Landers and Dr. Krishnamurthy for serving on my graduate committee. A special thanks to Mike Dancer and Jason Searcy for their work with the Attitude subsystem. I would like to thank the other members of the MR SAT team. Finally, I would like to give a special thanks to my wife and the rest of my family for always being behind me 100% throughout my entire academic career.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF ILLUSTRATIONS</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>x</td>
</tr>
<tr>
<td>NOMENCLATURE</td>
<td>xi</td>
</tr>
</tbody>
</table>

SECTION

1. INTRODUCTION

1.1. UNIVERSITY OF MISSOURI-ROLLA SATELLITE PROJECT

1.1.1. Physical Description of UMR SAT

1.1.2. Mission Objectives

1.1.3. Mission Sequence

1.1.3.1 Coupled Flight phase

1.1.3.2 Formation Flight phase

1.1.3.3 Extended Mission phase

1.2. PURPOSE

1.3. ATTITUDE DETERMINATION AND CONTROL FOR UMR SAT

1.3.1. UMR SAT Attitude Determination Requirements and Methods

1.3.2. UMR SAT Attitude Control Requirements and Methods

1.4. ORGANIZATION OF THESIS

2. REVIEW OF THE LITERATURE

2.1. INTRODUCTION

2.2. ATTITUDE DETERMINATION

2.2.1. Attitude Determination Sensors

2.2.2. New Methods of Attitude Determination

2.3. ATTITUDE CONTROL

2.3.1. Proven Methods of Attitude Control

2.3.2. New Methods of Attitude Control

3. PRELIMINARY CONSIDERATIONS
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1. Structural Concept for MR SAT and MRS SAT - Coupled</td>
<td>3</td>
</tr>
<tr>
<td>1.2. Structural Concept for MR SAT and MRS SAT – Formation Flight</td>
<td>3</td>
</tr>
<tr>
<td>3.1. Earth-Centered Inertial Frame</td>
<td>10</td>
</tr>
<tr>
<td>3.2. Local Rotating Frame</td>
<td>11</td>
</tr>
<tr>
<td>3.3. Panel Layout for MR SAT</td>
<td>12</td>
</tr>
<tr>
<td>3.4. MR SAT Body Fixed Frame</td>
<td>13</td>
</tr>
<tr>
<td>5.1. Simulation Flow Chart (Coupled Flight)</td>
<td>31</td>
</tr>
<tr>
<td>5.2. Simulation Flow Chart (Formation Flight)</td>
<td>31</td>
</tr>
<tr>
<td>5.3. Attitude Position Change with No Initial Angular Velocity</td>
<td>34</td>
</tr>
<tr>
<td>5.4. No Attitude Position Change with Nonzero Initial Angular Velocity</td>
<td>35</td>
</tr>
<tr>
<td>5.5. Attitude Position Change with Nonzero Initial Angular Velocity</td>
<td>36</td>
</tr>
<tr>
<td>5.6. Attitude Position Change with No Initial Angular Velocity</td>
<td>38</td>
</tr>
<tr>
<td>5.7. Desired and Produced Torque</td>
<td>39</td>
</tr>
<tr>
<td>5.8. No Attitude Position Change with Nonzero Initial Angular Velocity</td>
<td>40</td>
</tr>
<tr>
<td>5.9. Desired and Produced Torque</td>
<td>41</td>
</tr>
<tr>
<td>5.10. Attitude Position Change with Nonzero Initial Angular Velocity</td>
<td>42</td>
</tr>
<tr>
<td>5.11. Desired and Produced Torque</td>
<td>43</td>
</tr>
<tr>
<td>5.12. Attitude Position Change with No Initial Angular Velocity</td>
<td>44</td>
</tr>
<tr>
<td>5.13. Desired and Produced Torque (Throttling)</td>
<td>45</td>
</tr>
<tr>
<td>5.14. Desired and Produced Torque (No Throttling)</td>
<td>45</td>
</tr>
<tr>
<td>5.15. No Attitude Position Change with Nonzero Initial Angular Velocity</td>
<td>47</td>
</tr>
<tr>
<td>5.16. Desired and Produced Torque (Throttling)</td>
<td>47</td>
</tr>
<tr>
<td>5.17. Desired and Produced Torque (No Throttling)</td>
<td>48</td>
</tr>
<tr>
<td>5.18. Attitude Position Change with Nonzero Initial Angular Velocity</td>
<td>48</td>
</tr>
<tr>
<td>5.19. Desired and Produced Torque (Throttling)</td>
<td>49</td>
</tr>
<tr>
<td>5.20. Desired and Produced Torque (No Throttling)</td>
<td>49</td>
</tr>
<tr>
<td>5.21. Attitude Error Angle (Thruster Controller)</td>
<td>51</td>
</tr>
<tr>
<td>5.22. Active Thrusters During Control</td>
<td>52</td>
</tr>
</tbody>
</table>
5.23. Total Thruster Firings During Control........................................................................ 52
5.24. Attitude Error Angle (Thruster Controller)................................................................. 53
5.25. Attitude Error Angle (Coil Controller) ....................................................................... 54
5.26. Current Needed For Control (Coil Controller) ............................................................ 55
5.27. Attitude Error Angle (Coil Controller) ....................................................................... 55
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1. MR SAT and MRS SAT Estimated Properties</td>
<td>1</td>
</tr>
<tr>
<td>3.1. Nominal Attitude Configuration</td>
<td>16</td>
</tr>
<tr>
<td>5.1. Runge-Kutta Constants</td>
<td>29</td>
</tr>
<tr>
<td>5.2. Functions in Coupled Flight Simulation</td>
<td>30</td>
</tr>
<tr>
<td>5.3. Functions in Formation Flight Simulation</td>
<td>32</td>
</tr>
<tr>
<td>5.4. Guidance Law Verification Test 1</td>
<td>33</td>
</tr>
<tr>
<td>5.5. Guidance Law Verification Test 2</td>
<td>34</td>
</tr>
<tr>
<td>5.6. Guidance Law Verification Test 3</td>
<td>35</td>
</tr>
<tr>
<td>5.7. Orbital Parameters</td>
<td>37</td>
</tr>
<tr>
<td>5.8. Coil Controller Verification Test 1</td>
<td>38</td>
</tr>
<tr>
<td>5.9. Coil Controller Verification Test 2</td>
<td>39</td>
</tr>
<tr>
<td>5.10. Coil Controller Verification Test 3</td>
<td>41</td>
</tr>
<tr>
<td>5.11. Thruster Controller Verification Test 1</td>
<td>44</td>
</tr>
<tr>
<td>5.12. Thruster Controller Verification Test 2</td>
<td>46</td>
</tr>
<tr>
<td>5.13. Thruster Controller Verification Test 3</td>
<td>50</td>
</tr>
</tbody>
</table>
## NOMENCLATURE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>Semi-major axis</td>
</tr>
<tr>
<td>$A$</td>
<td>Cross-sectional area</td>
</tr>
<tr>
<td>$B$</td>
<td>Magnetic field vector of Earth</td>
</tr>
<tr>
<td>$B_X, B_Y, B_Z$</td>
<td>MR SAT Body-Fixed Frame</td>
</tr>
<tr>
<td>$b_X, b_Y, b_Z$</td>
<td>MRS SAT Body-Fixed Frame</td>
</tr>
<tr>
<td>$C_\alpha$</td>
<td>Cosine of an angle $\alpha$</td>
</tr>
<tr>
<td>$e$</td>
<td>Eccentricity of orbit</td>
</tr>
<tr>
<td>$F_d$</td>
<td>Direction of the dipole</td>
</tr>
<tr>
<td>$F_t$</td>
<td>Direction of the thrust</td>
</tr>
<tr>
<td>$i$</td>
<td>Angle of inclination</td>
</tr>
<tr>
<td>$I$</td>
<td>Inertia tensor</td>
</tr>
<tr>
<td>$I_c$</td>
<td>Current (magnetic coil)</td>
</tr>
<tr>
<td>$N$</td>
<td>Applied control moment</td>
</tr>
<tr>
<td>$N_{\text{turns}}$</td>
<td>Number of turns for a coil</td>
</tr>
<tr>
<td>$n$</td>
<td>Mean motion</td>
</tr>
<tr>
<td>$q$</td>
<td>Quaternion</td>
</tr>
<tr>
<td>$r$</td>
<td>Thruster location relative to the center of mass</td>
</tr>
<tr>
<td>$R_X, R_Y, R_Z$</td>
<td>MR SAT Rotating Frame</td>
</tr>
<tr>
<td>$r_X, r_Y, r_Z$</td>
<td>MRS SAT Rotating Frame</td>
</tr>
<tr>
<td>$S_\alpha$</td>
<td>Sine of an angle $\alpha$</td>
</tr>
<tr>
<td>$\tau_c$</td>
<td>Torque produced by coil</td>
</tr>
<tr>
<td>$\tau_t$</td>
<td>Torque produced by thruster</td>
</tr>
<tr>
<td>$\tau_{\text{mag}}$</td>
<td>Magnitude of torque</td>
</tr>
<tr>
<td>$u$</td>
<td>Throttle of a thruster</td>
</tr>
<tr>
<td>$X, Y, Z$</td>
<td>Earth-Centered Inertial Frame</td>
</tr>
<tr>
<td>$\phi, \Theta, \psi$</td>
<td>Euler angles</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Gravitational parameter for Earth</td>
</tr>
</tbody>
</table>
\( \nu \)  True anomaly
\( \Omega \)  Right ascension of the ascending node
\( \theta \)  Argument of latitude
\( w \)  Argument of periapsis
\( \omega \)  Angular velocity quaternion
\( \omega_e \)  Angular velocity of the Earth’s rotation
1. INTRODUCTION

1.1. UNIVERSITY OF MISSOURI-ROLLA SATELLITE PROJECT

Students and faculty in the University of Missouri-Rolla’s (UMR) Department of Mechanical and Aerospace Engineering began designing the university’s first satellite in August 2002. In March 2005, the UMR SAT project was accepted into the Nanosat 4 competition along with satellite projects from ten other universities. This competition is sponsored by the Air Force, NASA, and the American Institute of Aeronautics and Astronautics (AIAA). The completion of the UMR SAT (University of Missouri-Rolla Satellite) project is scheduled for February 2007 with the final competition taking place in March 2007.

1.1.1. Physical Description of UMR SAT. The UMR SAT project consists of two satellites. The first of these satellites is known as MR SAT (Missouri-Rolla Satellite). The second satellite is named Missouri-Rolla Second Satellite, or MRS SAT. Table 1.1 gives a brief summary of the mass and principle moments of inertia for the UMR SAT system as of January 2007. \( I_{XX}, I_{YY}, \) and \( I_{ZZ} \) are estimated moments of inertia for the spacecraft; precise values are being calculated for final testing. Detailed dimensions for the UMR SAT project can be found in Appendix B.

<table>
<thead>
<tr>
<th>Spacecraft</th>
<th>Mass (kg)</th>
<th>( I_{XX} ) (kg-m(^2))</th>
<th>( I_{YY} ) (kg-m(^2))</th>
<th>( I_{ZZ} ) (kg-m(^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR SAT</td>
<td>28.25</td>
<td>1.57</td>
<td>1.53</td>
<td>0.79</td>
</tr>
<tr>
<td>MRS SAT</td>
<td>11.45</td>
<td>0.55</td>
<td>0.62</td>
<td>0.34</td>
</tr>
<tr>
<td>Coupled</td>
<td>39.70</td>
<td>3.75</td>
<td>3.68</td>
<td>1.11</td>
</tr>
</tbody>
</table>

1.1.2. Mission Objectives. The UMR SAT project has a number of mission objectives. The core objective for the UMR SAT project is to develop autonomous
control to maintain formation flight. Secondary objectives include developing low-cost inter-satellite wireless communication (using Bluetooth technology), utilizing commercial off-the-shelf (COTS) hardware, and implementation of the nonlinear $\theta$-D controller [1-3] for orbital control and possibly attitude determination and control (ADAC). These core objectives flow from the Distributed Space System (DSS) technology, which is a NASA Goddard Space Flight Center initiative being developed to study multiple satellites flying in formations to replace large, expensive single satellites. Using multiple satellites might also decrease overall mission failures by eliminating single points of mission failures. UMR SAT requires innovative, low-cost hardware to comply with the modest budget accompanying a university project.

1.1.3. Mission Sequence. The UMR SAT project has defined many Modes of Operation. The modes that are important to the ADAC system have been categorized into three phases as detailed next: Coupled Flight, Formation Flight, and Extended Mission.

1.1.3.1 Coupled Flight phase. MR SAT and MRS SAT will be docked together for this portion of the mission as shown in Figure 1.1. The following phases are incorporated during the coupled flight of MR SAT and MRS SAT: Initialization Mode, Power-Up Mode, Detumble Mode, and Pre-Deploy Mode. Power-Up Mode will allow the solar cells to begin charging the batteries and, as power levels permit, activate the remaining flight systems. Detumble Mode will allow time to correct any tip-off error imparted when the satellites separate from the launch vehicle. During the Pre-Deploy Mode, MR SAT and MRS SAT will execute a series of system tests to verify the status of each system.

1.1.3.2 Formation Flight phase. This portion of the UMR SAT project consists of the Separation Mode and Formation Flight Mode. The Separation Mode will release MRS SAT from MR SAT using a Qwknut separation system. For the remainder of the mission, MR SAT and MRS SAT will function as two individual entities as shown in Figure 1.2, thus commencing the Formation Flight Mode. Because MRS SAT will not be equipped with an onboard propulsion system, MR SAT will “chase” MRS SAT to maintain the desired formation. Once MR SAT is depleted of propellant, the mission will transition into the Extended Mission phase.
1.1.3.3 Extended Mission phase. The Extended Mission phase consists only of the Range-Test Mode, where the range of the wireless communication system will be tested. Once this test is completed, the remainder of the Extended Mission phase will only involve MR SAT, because MRS SAT will no longer have a communication link to
MR SAT or UMR. During this part of the mission, additional testing of the autonomous systems will be conducted.

1.2. PURPOSE

The purpose of this thesis is to present the design and simulation results for an attitude control system to be used for the UMR SAT project. This system is responsible for using the determined attitude of the satellite to compute the correction needed to alter the current attitude to the desired attitude, as well as for determining the best method of control. Because the satellite is small, only a limited amount of propellant is carried onboard making it a valuable resource. One of the main objectives for this algorithm is to determine the optimal method of correction such that propellant can be preserved. This control law is an essential element to the success of the UMR SAT project.

1.3. ATTITUDE DETERMINATION AND CONTROL FOR UMR SAT

The ADAC system is responsible for determining the orientation of the spacecraft with respect to the surrounding environment. The ADAC system continuously monitors the attitude and applies corrections when necessary to maintain the desired orientation.

1.3.1. UMR SAT Attitude Determination Requirements and Methods. For the UMR SAT project, the attitude requirement is to determine the orientation of both satellites within 2° accuracy. This will be accomplished with the use of Billingsley Magnetometers.

1.3.2. UMR SAT Attitude Control Requirements and Methods. For the UMR SAT project, the attitude of MRS SAT must be controlled to within 10° to facilitate intersatellite communication via magnetic coils. The attitude control for MR SAT is required to maintain the orientation of the satellite within 5° via magnetic coils and cold gas thrusters.

1.4. ORGANIZATION OF THESIS

From this point on, this thesis discusses the methods used for the design, synthesis, and verification of the attitude control software for the UMR SAT project. The organization of this thesis is as follows:
Section 2 – Review of the Literature

This section presents publications relevant to attitude determination and control. The current state of research for this topic is discussed.

Section 3 – Preliminary Considerations

This section provides a foundation for the work to be done with the ADAC subsystem. First, a set of coordinate frames is defined as well as the rotations that connect them. Then, the equations of motion are defined and the dynamic model is discussed. Finally, an introduction into Euler Angles and Quaternions is discussed.

Section 4 – Attitude Control for MR SAT

This section presents the main body of research. The actual hardware used for MR SAT is discussed as well as what was needed to obtain the required data and results.

Section 5 – Results

This section reviews the design and structure of the control software. This software is simulated using the MATLAB language. This simulation will be added to the simulation that takes the measurements from the hardware and obtains the attitude determination to produce a UMR SAT ADAC software package. Once the current attitude and desired attitude for the satellite are known, the guidance law will determine the necessary correction needed, and the control law will dictate the method of correction. This section also details the steps taken to verify the simulation. The simulation code can be found in Appendix A.

Section 6 – Current Status and Future Development

The final section contains the current status of the UMR SAT ADAC system and it discusses the future development.
2. REVIEW OF THE LITERATURE

2.1. INTRODUCTION

With numerous satellite missions in the planning and modeling stages, ADAC methods are continually being synthesized and validated. The ADAC system is essential to the success of any mission. Many times very precise determination and control are required, especially in cases where imaging is used or communication is vital, and any slight misalignment could be catastrophic. During the past few years, an abundance of literature regarding ADAC methods has become available. Several interesting ADAC methods relevant to this thesis are discussed and provided within this section.

2.2. ATTITUDE DETERMINATION

This section outlines some of the interesting attitude determination methods that have been flown and validated in space as well as new methods still undergoing simulation and testing.

2.2.1. Attitude Determination Sensors. Before correcting the satellite to the desired orientation, the current orientation must be found using a variety of instruments. Many instruments are available for this task. For low Earth orbits (LEO), magnetometers are often considered. They measure the angle created between their body axis and Earth’s magnetic field. The main disadvantage of magnetometers is that the inaccuracies in Earth’s magnetic field model directly affect the accuracy of the magnetometer readings.

Earth, Sun, and star sensors are also readily available. Star sensors allow for very precise measurements, but they sometimes require an excessive amount of time to acquire these attitude measurements. Gyroscopes are sometimes used alongside star sensors because they obtain attitude measurements quickly; however, they are limited by instrument drift [4]. Earth, Sun, and star sensors are too costly for the UMR SAT project and cannot be considered.

2.2.2. New Methods of Attitude Determination. Recently, many advancements have occurred in attitude determination methods. One advancement is the application of solid state gyroscopes either as Inertial Measurement Units (IMUs), which detect altitude, location, and motion, or microelectromechanical systems (MEMS), which are small
multi-axis systems used mostly in automobiles but are recently being developed as low-cost sensors for microsatellites [5]. These gyroscopes do not have an instrument drift to consider. Larger IMUs have flown in the Space Shuttle, but much smaller IMUs have been considered for use in UMR SAT.

Another method in frequent use is filtering. In some cases filtering can provide added accuracy to the measurements, and other times it can reduce the amount of needed hardware. Filters have been developed that use global positioning system (GPS) receivers, such as described in Reference [6], as well as filters that use data from magnetometer measurements, such as in Reference [7]. In Reference [7], the use of a Kalman Filter and an Extended Kalman Filter (EKF) are presented. One filters data from a magnetometer and a gyroscope, while the other filters data from a magnetometer and Sun sensor data. The filters were able to significantly improve the accuracy of the attitude determination model.

Reference [8], which is a survey paper of current attitude estimation methods, cites multiple references (~23) to EKFs as well as other determination methods. One of these methods is the quaternion estimator (QUEST), which is a point-by-point solution. Another method is implementing an Unscented Filter, which works on the premise that with a fixed number of parameters it should be easier to approximate the Gaussian distribution than to approximate an arbitrary nonlinear function [8]. There is also mention of using a particle filter that is based on sequential Monte Carlo simulations. Other filters mentioned are an orthogonal filter and a nonlinear predictive filter.

The UMR SAT project will use magnetometer-only determination for two reasons. One, the magnetometer is a relatively inexpensive, very reliable sensor that provides reasonable accuracy for a low cost. Second, the addition of Sun sensors or gyroscopes would complicate the system on a satellite that is already volume-constrained on the inside with a limited mass budget. A number of studies including Reference [9] and Reference [10] have shown that magnetometer-only determination can be accomplished with reasonable accuracy.
2.3. ATTITUDE CONTROL

This section outlines some of the interesting attitude control methods that have been flown and shown to work in space as well as new methods still undergoing simulation and testing.

2.3.1. Proven Methods of Attitude Control. Attitude control systems are needed because the spacecraft orientation will drift from the desired orientation. An attitude correction may also be needed to rotate a spacecraft to an orientation suitable for ground transmission or so the solar cells can charge. Attitude control can be either passive or active. Two passive means of attitude control include utilizing the gravity gradient, which controls the spacecraft by extending a boom and relying on gravitational properties to keep the satellite pointed towards Earth, and spin stabilization, a technique in which the satellite spins so that its angular momentum vector tends to remain fixed in inertial space [4]. Gravity gradient control and spin stabilization are less precise but much less expensive than other methods of attitude control.

Three-axis active control systems are more expensive but are also more precise. These techniques include magnetic torque rods, thrusters, momentum and reaction wheels, and control moment gyros. Reference [11] surveys multiple thruster control systems for microspacecraft. Bi-propellant systems are convenient because they have relatively high specific impulse performances; however, they are too complex for the UMR SAT mission. Instead, the UMR SAT mission will rely on a mono-propellant cold gas thruster system. Cold gas thrusters represent the smallest rocket engine technology available today [11]. When using a mono-propellant system, a benign propellant is ideal because it does not present contamination troubles; however, specific impulse is significantly reduced compared to bi-propellant systems.

The REIMEI microsatellite [12], launched in 2005, was equipped with three magnetic torquers (MTQs) and one momentum wheel, and it achieved a pointing accuracy of $0.05^\circ - 0.1^\circ$. MTQs were chosen because of their small size, mass, and cost. Several studies were performed and spacecraft using only MTQs had a pointing accuracy of about $1^\prime$ [12], which is well within the requirements for the UMR SAT mission. Reference [13] introduces theoretical results confirmed via computer simulation of an
attitude control system using only MTQs. The results show that three-axis control is achieved using an inter loop – outer loop design.

2.3.2. New Methods of Attitude Control. The control methods for spacecraft have also been improved throughout the past few years. Much like attitude determination methods, filtering is becoming more widely used with attitude control methods as well. New methods of propulsion via pulsed plasma thrusters (PPTs) ablate, ionize, and electromagnetically accelerate atoms and molecules from a bar of solid Teflon [11]. These thrusters are still in the developmental stages and have not been proven to work with the mass, volume, and cost budgets for the UMR SAT mission. Future designs of PPTs may have masses as low as 0.5 kg with predicted thrusts of 10 – 100 μNs [11]. Another improvement is that spacecraft with smaller mass, volume, and limited budgets are using magnetic coils instead of MTQs. These coils can be made to suit the individual satellite, but for much less. Another student-designed satellite, HokieSat (Virginia Tech), has been equipped with magnetic coils instead of MTQs [14]. HokieSat was presented before Air Force and NASA personnel and met the safety constraints of the Space Shuttle. Virginia Tech students are now awaiting a launch date. Also, Utah State University’s satellite USUsat III uses magnetic coils [15]. These are a few of the pioneering universities that are now using magnetic coils instead of MTQs.
3. PRELIMINARY CONSIDERATIONS

3.1. COORDINATE FRAMES

When deriving equations of motion to facilitate an analysis of the attitude dynamics for a satellite, a set of coordinate frames is first needed. These coordinate frames will be important and used throughout the UMR SAT project. Three main coordinate frames are used in this study: the Inertial Frame, the Local Rotating Frame, and the Body Fixed Frame.

3.1.1. Inertial Frame. An Earth-centered frame was chosen as the Inertial Frame. This frame is considered to be an “external” frame because the origin is the center of the Earth and not the satellite. From Reference [16], the coordinate frame was defined such that “the vernal equinox serves as the X-axis, the positive Z-axis passes through the North Pole, and the Y-axis completes the right-handed frame.” This coordinate frame does not rotate with the spin of the Earth about its axis, as shown in Figure 3.1.

![Figure 3.1. Earth-Centered Inertial Frame](image-url)
3.1.2. **Local Rotating Frame.** The Local Rotating Frame acts as an intermediate frame. “This frame orbits the Earth circularly and is fixed to the center of mass of the spacecraft. The orientation of the Local Rotating Frame is defined by its orbit about the Earth. The $\mathbf{R}_X$-axis is aligned along the spacecraft’s velocity vector, the $\mathbf{R}_Z$-axis continually points towards Earth, and the $\mathbf{R}_Y$-axis completes the right-handed frame. As a result, the Local Rotating Frame will make one rotation per orbit. This frame provides a convenient reference direction to ensure that the space-to-ground antenna always points towards Earth” [16]. Similarly, for MRS SAT the Local Rotating Frame is defined as $\mathbf{r}_X$, $\mathbf{r}_Y$, and $\mathbf{r}_Z$. Figure 3.2 shows the Local Rotating Frame on the orbit.

![Figure 3.2. Local Rotating Frame](image)

3.1.3. **Body Fixed Frame.** A standard satellite Body Fixed Frame has been established for consistent documentation. This frame is specifically important to the Attitude subsystem for the dynamic model as well as to the determination and control
algorithms. This frame is also important to the Structures subsystem for defining the center of mass location, the moments of inertia, and the drawings and component placement.

### 3.1.3.1 Body Fixed Frame MR SAT

Figure 3.3 shows a layout of the six side panels for MR SAT, and it also illustrates where the axes are located with respect to the panels. Each panel has been given a specific designation: the side panels are P1, P2, P3, P4, P5, and P6 as shown in an “unfolded” configuration in Figure 3.3, and the top and bottom panels have been labeled PT and PB.

![Figure 3.3. Panel Layout for MR SAT](image)

The axes for the Body Fixed Frame have been labeled as shown in Figure 3.3 and Figure 3.4. The x-axis, $B_X$, is positive directed out of P1. The y-axis, $B_Y$, is positive directed out between P2 and P3. The z-axis, $B_Z$, completes the right-hand frame by being positive out of PT. The origin for this frame is in the center of PB.

### 3.1.3.2 Body Fixed Frame MRS SAT

The Body Fixed Frame for MRS SAT is analogous to the Body Fixed Frame defined for MR SAT with the exception that the
nomenclature of the panels is p1, p2, p3, p4, p5, p6, pb, and pt and the origin for the MRS SAT Body Fixed Frame is in the center of pb. The axes for the MRS SAT Body Fixed Frame are $b_x$, $b_y$, and $b_z$.

![Figure 3.4. MR SAT Body Fixed Frame](image.png)

**3.1.3.3 Nominal attitude configuration for UMR SAT.** The nominal attitude configuration for MR SAT is driven by communication and power constraints. The satellite-to-ground antenna has to be placed on the satellite panel facing Earth, and the panel with the least solar cells also ideally faces Earth because it would be exposed to the least amount of sunlight. Because P1 has four thrusters on it, the satellite-to-ground antenna is also placed on P1 and the $b_x$ axis points towards Earth. The $b_y$ axis points in the direction opposite of the satellite velocity, and the $b_z$ axis completes the right-handed system by pointing in the direction of the orbit angular momentum.

Once MR SAT and MRS SAT are separated, the nominal attitude configuration for MRS SAT depends on inner satellite communication and power; however,
MRS SAT’s attitude control is limited, and the nominal attitude control requirements are relaxed to primarily prevent excessive tumble.

3.2. COORDINATE TRANSFORMATIONS

The next step in determining the equations of motion is to relate the three coordinate frames to each other through a series of rotations. To relate the Inertial Frame to the Local Rotating Frame, three rotations are required. These rotations are combined to form a direction cosine matrix (DCM). Three more rotations are required to rotate from the Local Rotating Frame to the Body Fixed Frame, which establishes another DCM. Therefore, two DCMs are used to rotate from the Inertial Frame to the Body Fixed Frame.

3.2.1. Inertial Frame to Local Rotating Frame Transformation. This transformation occurs through three rotations by performing a body 3-1-3 rotation. The first rotation is about the $\mathbf{Z}$ axis by an amount $\Omega$, the right ascension of the ascending node. This produces an intermediate frame: $X', Y', Z'$. The second rotation is about the $X'$ axis by the angle $i$, the inclination, creating another intermediate frame: $X'', Y'', Z''$. Rotation three is about the $Z''$ axis by the angle $\theta$, the argument of latitude. This angle, $\theta$, is found by taking the sum of the true anomaly, $\nu$, and the argument of periapsis, $w$. For convenience, the trigonometric terms have been condensed as

\begin{align*}
S_\alpha & \triangleq \sin \alpha \\
C_\alpha & \triangleq \cos \alpha
\end{align*}

The series of rotations produce a DCM as

\begin{equation}
\begin{pmatrix}
\hat{\mathbf{X}} \\
\hat{\mathbf{Y}} \\
\hat{\mathbf{Z}}
\end{pmatrix} =
\begin{bmatrix}
C_\Omega C_\theta - S_\Omega C_\theta S_\alpha - C_\Omega S_\theta S_\alpha C_\theta - S_\Omega S_\theta C_\alpha C_\theta & S_\Omega S_i \\
S_\Omega S_\theta + C_\alpha C_\theta S_\alpha & S_\Omega S_\theta C_\alpha C_\theta - C_\Omega S_\alpha S_i \\
S_\alpha S_\theta & S_\alpha C_\theta - C_\alpha S_i
\end{bmatrix}
\begin{pmatrix}
\mathbf{r} \\
\mathbf{\hat{\theta}} \\
\mathbf{\hat{h}}
\end{pmatrix}
\end{equation}
with \( \hat{r} \), \( \hat{\theta} \), and \( \hat{h} \) representing unit vectors in the Local Rotating Frame. These unit vectors represent the radial direction of the orbit, the tangential direction to the orbit, and completion of the right-hand frame, respectively. One useful property of a DCM is that it is orthogonal so that

\[
[DCM]^{-1} = [DCM]^T
\]  

(3.4)

Therefore, the transformation from the Inertial Frame to the Local Rotating Frame can be written as

\[
\begin{bmatrix}
\hat{r} \\
\hat{\theta} \\
\hat{h}
\end{bmatrix} =
\begin{bmatrix}
C_\Omega C_\theta - S_\Omega S_i S_\theta & S_\Omega C_\theta + C_\Omega S_i S_\theta & S_\theta S_i \\
-\Omega S_\theta - S_\Omega C_i C_\theta & -\Omega S_\theta + C_\Omega C_i S_\theta & S_\theta C_i \\
S_\Omega S_i & -\Omega S_i & C_i
\end{bmatrix}
\begin{bmatrix}
\hat{X} \\
\hat{Y} \\
\hat{Z}
\end{bmatrix}
\]  

(3.5)

From this rotation, the angular velocity of the rotating frame can be expressed as

\[
^I \omega^R = \hat{\theta} \hat{h} = \hat{\theta} \left[ S_\Omega S_i \hat{X} - C_\Omega S_i \hat{Y} + C_i \hat{Z} \right]
\]  

(3.6)

A detailed analysis of this rotation can be found in Reference [16].

3.2.2. Local Rotating Frame to Body Fixed Frame Transformation. This transformation is completed using a body 3-2-1 rotation. Two more intermediate frames are defined: \( R_x' \), \( R_y' \), \( R_z' \) and \( R_x'', R_y'', R_z'' \). The angles \( \phi \), \( \Theta \), and \( \psi \) are rotated about the \( R_z, R_y' \), and \( R_x'' \) axes, respectively. The DCM to rotate the Body Fixed Frame to the Local Rotating Frame is

\[
\begin{bmatrix}
\hat{R}_x \\
\hat{R}_y \\
\hat{R}_z
\end{bmatrix} =
\begin{bmatrix}
C_\phi C_\Theta & -S_\phi C_\psi + C_\phi S_\Theta S_\psi & S_\phi S_\psi + C_\phi S_\Theta C_\psi \\
S_\phi C_\Theta & C_\phi C_\psi + S_\phi S_\Theta S_\psi & -C_\phi S_\psi + S_\phi S_\Theta C_\psi \\
-S_\Theta & C_\Theta S_\psi & C_\Theta C_\psi
\end{bmatrix}
\begin{bmatrix}
\hat{B}_x \\
\hat{B}_y \\
\hat{B}_z
\end{bmatrix}
\]  

(3.7)
From this rotation, the angular velocity of the BFF to the Local Rotating Frame can be expressed as

\[ \mathbf{\omega}^B = R_{\omega_x} \mathbf{B}_x + R_{\omega_y} \mathbf{B}_y + R_{\omega_z} \mathbf{B}_z = (-\dot{\phi} S_\Theta S_\psi + \psi_\Theta) \mathbf{B}_x + (\dot{\phi} C_\Theta S_\psi + \dot{\psi} C_\psi) \mathbf{B}_y + (\dot{\phi} C_\Theta C_\psi - \dot{\Theta} S_\psi) \mathbf{B}_z \] (3.8)

The angular rates \( \dot{\phi}, \dot{\Theta}, \) and \( \dot{\psi} \) are needed in order to solve the equations of motion.

Solving Equation 3.8 for these rates gives

\[ \dot{\phi} = \frac{R_{\omega_x} S_\psi}{C_\Theta} + \frac{R_{\omega_z} C_\psi}{C_\Theta} \] (3.9)

\[ \dot{\Theta} = R_{\omega_y} C_\psi - R_{\omega_z} S_\psi \] (3.10)

\[ \dot{\psi} = R_{\omega_x} + \frac{S_\psi S_\Theta R_{\omega_z} C_\Theta}{C_\Theta} + \frac{C_\psi S_\Theta R_{\omega_z}}{C_\Theta} \] (3.11)

A detailed analysis of this rotation can be found in Reference [16].

**3.2.3. Nominal Attitude Configuration for UMR SAT – Euler Angles.** The nominal attitude configuration for MR SAT represented in the Local Rotating Frame is given in Table 3.1.

| \( \phi \) | 180° |
| \( \Theta \) | 0° |
| \( \psi \) | 0° |
| \( \omega_x \) | 0°/sec |
| \( \omega_y \) | 0°/sec |
| \( \omega_z \) | 0°/sec |
These Euler angle values correspond with the physical nominal attitude configuration defined in Section 3.1.3.3.

3.3. EQUATIONS OF MOTION

Equations of motion were previously derived for the UMR SAT project and documented in Reference [16]. These equations were derived based upon Euler’s EOMs shown in general form as

$$\begin{align*}
N_x &= I_{xx} \dot{\omega}_x^B - \dot{\omega}_y^B \omega_z^B (I_{yy} - I_{zz}) \\
N_y &= I_{yy} \dot{\omega}_y^B - \dot{\omega}_x^B \omega_z^B (I_{zz} - I_{xx}) \\
N_z &= I_{zz} \dot{\omega}_z^B - \dot{\omega}_x^B \omega_y^B (I_{xx} - I_{yy})
\end{align*}$$

The six equations of motion for the satellite are given by Equations 3.9 – 3.11 and the three equations

$$\begin{align*}
\dot{\omega}_x^B &= \frac{N_x + \dot{\omega}_y^B \omega_z^B (I_{yy} - I_{zz})}{I_{xx}} \\
\dot{\omega}_y^B &= \frac{N_y + \dot{\omega}_x^B \omega_z^B (I_{zz} - I_{xx})}{I_{yy}} \\
\dot{\omega}_z^B &= \frac{N_z + \dot{\omega}_x^B \omega_y^B (I_{xx} - I_{yy})}{I_{zz}}
\end{align*}$$

found by applying Equations 3.12 – 3.14, where

$$\begin{align*}
\dot{R}_x^B &= \dot{\omega}_x^B + \dot{\theta} S_\phi C_\theta \\
\dot{R}_y^B &= \dot{\omega}_y^B + \dot{\theta} (C_\phi C_\psi + S_\phi S_\theta S_\psi) \\
\dot{R}_z^B &= \dot{\omega}_z^B + \dot{\theta} (-C_\phi S_\psi + S_\phi S_\theta C_\psi)
\end{align*}$$
A detailed derivation of these equations of motion can be found in Reference [16]. Equations 3.18 – 3.20 are substituted into Equations 3.15 – 3.17, which are simultaneously numerically integrated along with Equations 3.9 – 3.11. Note that \( \dot{\theta} \) is independent of the spacecraft attitude and is determined by the orbit. If the orbit is circular, then \( \dot{\theta} \) is equal to the mean motion, \( n \), where

\[
n = \sqrt{\frac{\mu}{a^3}}
\]

(3.21)

where \( \mu \) is the gravitational parameter for Earth \((3.986 \times 10^5 \text{ km}^3/\text{s}^2) \) and \( a \) is the semi-major axis of the orbit.

**3.4. EULER ANGLES AND QUATERNIONS**

Unlike vectors in \( \mathbb{R}^3 \) space, quaternions, sometimes known as Euler parameters and a 4-tuple of real numbers, are denoted in \( \mathbb{R}^4 \) space [17] and represented as

\[
q = (q_0, q_1, q_2, q_3)
\]

(3.22)

One alternative method of representing a quaternion is to define a scalar component, \( q_0 \), and a vector component, \( \mathbf{v}_q \) or \( \mathbf{ar{q}} \), shown as

\[
q = \langle q_0, \mathbf{v}_q \rangle = q_0 + \mathbf{ar{q}} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}
\]

(3.23)

The scalar component is a real number, while the vector component is taken as imaginary. One unique property of a quaternion is that the magnitude is equal to unity so that

\[
|q| = 1
\]

(3.24)
When two quaternions are multiplied together, much like vectors or matrices, the order is important. The following equation shows an example of one quaternion, $\omega$, multiplied with another quaternion, $q$.

$$\omega q = \langle q_0\omega_0 - v_\omega \cdot v_q, \omega_0 v_q + q_0 v_\omega + v_\omega \times v_q \rangle \quad (3.26)$$

The complex conjugate of a quaternion is denoted by

$$q^* = \langle q_0, -v_q \rangle \quad (3.27)$$

If a quaternion is multiplied by its complex conjugate, then only the real part of the quaternion remains, i.e.

$$q^*q = qq^* = \langle q_0^2 + |v_q|^2, 0 \rangle \triangleq |q|^2 \quad (3.28)$$

### 3.4.1. Euler Angles Transformed to Quaternions.

When using Euler angles to simulate ADAC systems, singularities often arise in the trigonometry. To bypass these singularities, the ADAC Model uses quaternion notation instead of Euler angles for the analysis. The Euler angles $\phi$, $\Theta$, and $\psi$ can be transformed to quaternion form as found in Reference [17] using

$$\begin{bmatrix}
q_0 \\
q_1 \\
q_2 \\
q_3
\end{bmatrix} = 
\begin{bmatrix}
\cos\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) \\
\cos\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) - \sin\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) \\
\cos\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) \\
\sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) - \cos\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)
\end{bmatrix} \quad (3.29)$$
3.4.2. Quaternions Transformed to Euler Angles. Once an ADAC simulation has been run, the quaternions are transformed back to Euler angles. This is done because it is difficult to readily recognize how the satellite is rotating in quaternion form; however, Euler angles clearly show the satellite orientation. Reference [17] outlines the necessary transformation matrix needed to transform quaternion form to Euler angles, given by

\[
R = \begin{bmatrix}
2(q_0^2+q_1^2)-1 & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\
2(q_1q_2-q_0q_3) & 2(q_0^2+q_2^2)-1 & 2(q_2q_3+q_0q_1) \\
2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & 2(q_0^2+q_3^2)-1
\end{bmatrix}
\]

(3.30)

\[
\phi = \tan^{-1}\left(\frac{R(1,2)}{R(1,1)}\right) = \tan^{-1}\left(\frac{2(q_1q_2+q_0q_3)}{2(q_0^2+q_1^2)-1}\right)
\]

(3.31)

\[
\theta = \sin^{-1}(R(1,3)) = \sin^{-1}\left(-2(q_1q_3-q_0q_2)\right)
\]

(3.32)

\[
\psi = \tan^{-1}\left(\frac{R(2,3)}{R(3,3)}\right) = \tan^{-1}\left(\frac{2(q_2q_3-q_0q_1)}{2(q_0^2+q_3^2)-1}\right)
\]

(3.33)

Equation 3.32 consists of an inverse sine, so a quadrant check must be completed because two values of \(\theta\) qualify as the answer. However, because of the 3–2–1 rotation, the second rotation is limited to between -90° and 90°.
4. ATTITUDE CONTROL FOR MR SAT AND MRS SAT

Once the attitude of MR SAT and MRS SAT has been determined, the following procedure reveals the optimum control rotation as well as the control method.

4.1. GUIDANCE LAW

The first element of the attitude control system is the guidance law. The satellite attitude kinematics and dynamics are defined using quaternion notation as

\[ q = \langle q_0, v_q \rangle \]  \hspace{1cm} (4.1)

\[ \omega = \langle \omega_0, v_\omega \rangle \]  \hspace{1cm} (4.2)

\[ \dot{q} = -\frac{1}{2} \omega q \]  \hspace{1cm} (4.3)

\[ \dot{v}_\omega = \frac{N - v_\omega \times I v_\omega}{I} \]  \hspace{1cm} (4.4)

where \( q \) is the attitude quaternion, \( \omega \) is the angular velocity quaternion, \( I \) is the satellite inertia tensor, and \( N \) is the applied control moment. To determine the required \( N \), a guidance law has been developed using an inner/outer loop design. The inner loop determines the desired angular velocity, \( \omega_d \), so that the current attitude quaternion, \( q \), will track the desired attitude quaternion, \( q_d \). The outer loop then finds the required control moment, \( N \), which ensures that the current angular velocity, \( v_\omega \), tracks the desired angular velocity, \( \omega_d \). The guidance law also guarantees that the final angular velocity will be zero with respect to the Inertial Frame.

The Lyapunov Stability principle was applied to demonstrate controller stability. A detailed description and analysis of this process can be found in Appendix C. To perform a Lyapunov stability assessment, a positive definite Lyapunov function must be identified with a negative definite derivative. If the derivative is found to be negative semi-definite, then the zero points must be equilibrium points to prove system stability.

4.1.1. Guidance Law – Inner Loop. To determine the desired angular velocity that tracks the current attitude quaternion to a desired attitude quaternion, an error
quaternion, \( e_q = q - q_d \), is defined. Then a positive definite Lyapunov function is selected as half of the square of the error quaternion magnitude. The Lyapunov function and its derivative are

\[
V = \frac{1}{2} \text{Re} \left( e_q^* e_q \right) = \frac{1}{2} \left| e_q \right|^2 \tag{4.5}
\]

\[
\dot{V} = \text{Re} \left( e_q^* \left( \dot{q} - \dot{q}_d \right) \right) \tag{4.6a}
\]

where \( e_q^* \) is the complex conjugate of \( e \), and \( \text{Re}: \mathbb{H} \rightarrow \mathbb{R} \) extracts the real part of the quaternion \( q \in \mathbb{H} \), with \( \mathbb{H} \) and \( \mathbb{R} \) representing the set of all quaternions and all real numbers, respectively. The derivative of the desired quaternion, \( \dot{q}_d \), is assumed to be zero (corresponding to a fixed attitude), which leads to

\[
\dot{V} = \text{Re} \left( e_q^* \dot{q} \right) = \text{Re} \left( e_q^* \left( -\frac{1}{2} \omega q \right) \right) \tag{4.6b}
\]

Using Equation 3.25 and rearranging,

\[
\dot{V} = -\frac{1}{2} \text{Re} \left( e_q^* \left\{ q_0 \omega_0 - v_q \cdot v_\omega, \ q_0 v_\omega + \omega_0 v_q + v_\omega \times v_q \right\} \right) \tag{4.7}
\]

Because \( \omega \in \mathbb{H}_0 \), where \( \mathbb{H}_0 \) is the set of all quaternions for which the real part is zero,

\[
\omega_0 = 0 \tag{4.8}
\]

The derivative of the Lyapunov function can be shown to be

\[
\dot{V} = -\frac{1}{2} \left[ e_{\omega_q} \cdot \left( q_0 v_\omega + v_\omega \times v_q \right) - e_{\omega_q} \left( v_\omega \cdot v_q \right) \right] \tag{4.9}
\]

The derivative of the Lyapunov function can be made negative semi-definite by choosing
\[ v_{\omega} = \omega_{d} = q_{0}v_{e_{q}} - e_{0}v_{q} - \left( v_{e_{q}} \times v_{q} \right) \]  

(4.10)

Using the above desired angular velocity, the zero points can be shown to be equilibrium points because they occur only when \( q \) is collinear to \( q_{d} \).

4.1.2 Guidance Law – Outer Loop. To ensure that the necessary \( \omega_{d} \) is obtained, a moment, \( N \), can be found such that \( v_{\omega} \) will be driven to \( \omega_{d} \). This is done by defining another error, \( e_{\omega} = v_{\omega} - \omega_{d} \). This error function must quickly go to zero for the inner/outer loop method to function effectively. When \( v_{\omega} = \omega_{d} \), the system is stable and \( q \to q_{d} \).

Again, a selected definite Lyapunov function and its derivative are

\[ V = \frac{1}{2} e_{\omega}^{T} e_{\omega} \]  

(4.11)

\[ \dot{V} = e_{\omega}^{T} \left( N - v_{\omega} \times I v_{\omega} \right) \]  

(4.12)

From this, the necessary control moment, \( N \), which ensures that the derivative is negative definite and that \( v_{\omega} \) will approach \( \omega_{d} \), is

\[ N = \omega_{d} \times I \omega_{d} - e_{\omega} \]  

(4.13)

4.2. CONTROL LOGIC

MR SAT relies on magnetic coils and cold gas thrusters as its two methods of attitude control, while MRS SAT uses only magnetic coils. The needed control moment as determined from the guidance law is passed on to the control logic algorithm. The control logic algorithm determines the method of control as well as the optimal control for that method.

4.2.1. Magnetic Coils. The primary method of attitude control for MR SAT and MRS SAT in the docked configuration, during Detumble mode, will be MR SATs three magnetic coils. The magnetic coils function much like a solenoid; when a current is passed through a coil of wire, a magnetic dipole is created. The generated magnetic dipole then interacts with the Earth's magnetic field, producing a torque, as it attempts to
align itself with Earth’s magnetic field. The three magnetic coils in MR SAT are positioned such that different dipole directions are obtained. The governing equation for the torque generated by a single magnetic coil is

$$\boldsymbol{\tau}_c = N_{\text{turns}} A I_c F_n \times \mathbf{B}$$

(4.14)

where $\boldsymbol{\tau}_c$ is the torque produced by the coil, $N_{\text{turns}}$ is the number of turns of the coil, $A$ is the coil cross-sectional area, $I_c$ is the current applied to the coil, $F_n$ is a unit vector in the direction of the created dipole, and $\mathbf{B}$ is Earth’s magnetic field vector. Because the MR SAT spacecraft is equipped with three magnetic coils, there exists a combination of possible currents, sent to the three coils, which will result in the torque required by the guidance law. The coil control login then determines the coil currents, which minimizes the total power sent to the three coils. The method of Lagrange Multipliers was applied to find these optimal currents, as detailed next.

The optimal currents to be sent through the magnetic coils, producing the desired torque from the guidance law, were calculated using Lagrange Multipliers. The optimal currents are defined as the currents that minimize the cost function

$$J = \frac{1}{2} I_c^T I_c$$

(4.15)

where $I_c$ is now a 3x1 vector of currents input to the three coils. The cost function is subject to the constraint that the net torque produced by the coils must be the torque required by the guidance law. The quadratic cost function was chosen because it is desired to conserve power (which is directly proportional to the square of current). Next, the Hamiltonian was defined as

$$H = \frac{1}{2} I_c^T I_c + \lambda^T \left( (N_{\text{turns}} A F_n \times \mathbf{B}) I_c - \mathbf{N} \right)$$

(4.16)

The partial derivatives of the Hamiltonian were then found for both $I_c$ and $\lambda$ as
\[
\frac{\partial H}{\partial I_c} = I_c + (N_{\text{turns}}A F_n \times B)^T \lambda = 0
\] (4.17)

\[
\frac{\partial H}{\partial \lambda} = (N_{\text{turns}}A F_n \times B) I_c - N = 0
\] (4.18)

Solving for \(I_c\) and \(\lambda\), the Lagrange multipliers, gives

\[
\begin{bmatrix} \lambda \\
I_c \end{bmatrix} = \begin{bmatrix} (N_{\text{turns}}A F_n \times B)^T & 1 \\
0 & (N_{\text{turns}}A F_n \times B) \end{bmatrix}^{-1} \begin{bmatrix} 0 \\
N \end{bmatrix}
\] (4.19)

This solution, however, cannot be solved because the matrix inverse does not exist. This occurs because each coil can only produce a torque that is orthogonal to Earth's magnetic field. The torque tensor, produced by the three coils, is linearly dependent, thus

\[
\begin{vmatrix} (N_{\text{turns}}A F_n \times B)^T & 1 \\
0 & (N_{\text{turns}}A F_n \times B) \end{vmatrix} = \left|(N_{\text{turns}}A F_n \times B)\right|^2 = 0
\] (4.20)

The magnetic field changes as the satellite orbits the Earth; therefore, if the magnetic coils continually produce the component of the needed torque perpendicular to the magnetic field, then the satellite attitude is still controllable because a torque can be applied in any direction with some delay. To determine the currents that produce the required torque normal to the magnetic field, two nonzero basis vectors that are linearly independent and orthogonal to the magnetic field are calculated. The first basis vector is calculated by crossing the magnetic field vector body-fixed coordinate axis with the largest angular separation from the magnetic field. Use of the cross product ensures the resulting vector is orthogonal to the magnetic field. The second basis vector needs to be orthogonal to both the first basis vector and the magnetic field vector, so the cross product between the two gives the second basis vector. The resulting torque is then a linear combination of these two basis vectors; therefore, the constraint is transformed such that the resulting torque and desired torque have the same coefficients in the linear
combination. In order to prevent the numerical solution of the linear system from being singular, a scaling factor of 100,000 was introduced to control the matrix determinant. The transformation matrix can be shown as

\[ R_C = KR_c \]  

(4.21)

where K is the scale factor and \( R_C \) is the transformation matrix created by stacking the basis vectors in subsequent matrix rows. Equation 4.19 then becomes

\[
\begin{bmatrix}
\mathbf{A} \\
\mathbf{I}_c
\end{bmatrix} = \begin{bmatrix}
R_C \left( N_{\text{turns}} AF_n \times \mathbf{B} \right)^T & 1 \\
0 & R_C \left( N_{\text{turns}} AF_n \times \mathbf{B} \right)
\end{bmatrix}^{-1} \begin{bmatrix}
0 \\
R_c N
\end{bmatrix}
\]  

(4.22)

Once the needed currents are calculated, they are then scaled, as needed, to hardware limits of between ±1 amp. By simply scaling the currents, the resulting torque is in the same direction as the required torque with possibly a smaller magnitude.

### 4.2.2. Cold Gas Thrusters.

Eight small cold gas thrusters will be used on MR SAT to provide additional attitude and orbit control. These thrusters will be essential for the Formation Flight phase when MR SAT will fly in a formation with MRS SAT. Each thruster produces a torque that is equivalent to

\[ \tau_i = \tau_{\text{mag}} r \times F_T \]  

(4.23)

where \( \tau_i \) is the torque produced by the thruster, \( \tau_{\text{mag}} \) is the maximum thrust produced by each thruster, \( r \) is the location of the thruster relative to the satellite center of mass, and \( F_T \) is a unit vector in the direction of force for the thruster.

#### 4.2.2.1 Linear programming.

The appropriate throttle settings for each of the eight thrusters are determined using linear programming [18]. While the thrusters cannot be throttled, the thrusters can be used in a pulse-width modulation (PWM) type scheme to obtain a desired throttle setting. This procedure is discussed in the next section. This method allows calculation of the optimal throttle settings that minimize the cost function.
\[ J = \sum \theta_i \ (i = 1, 2, \ldots, 8), \]
where \( 0 \leq \theta_i \leq 1 \) is the throttle setting (the percentage of thrust needed during a fixed time interval, \( \Delta t \)) for the \( i \)th thruster. This cost function effectively minimizes the net propellant consumption rate for the entire thruster control system. In the linear programming solution, the throttle settings are constrained by \( F = \sum F_i \theta_i \) and \( \tau = \sum \tau_i \theta_i \), which ensures that the force and torque needed by the orbit and attitude control systems are achieved.

### 4.2.2.2 Throttle filter
Because the thrusters for MR SAT cannot be throttled, the system cannot provide the performance commanded by the linear programming solution. A method is needed to determine whether the thrusters should be on or off during a given time interval, \( \Delta t \). The method used compares a simple filter estimate of the current throttle setting, \( \hat{\theta} \), with the desired throttle setting calculated from the linear programming thruster solution. If the estimated throttle setting is higher than the desired throttle setting, then the thruster is not fired during the next \( \Delta t \) time interval. However, if the estimated throttle setting is lower than the desired throttle setting, then the thruster is fired during the next \( \Delta t \) interval. Thus, the thruster logic is given by

\[
u = \begin{cases} 
0 & \theta < \hat{\theta} \\
1 & \theta > \hat{\theta}
\end{cases}
\]  

(4.24)

where \( u = 1 \) indicates the thruster is fired, and \( u = 0 \) means the thruster is not fired. Once the \( u \) is determined for each thruster, the throttle estimates are updated using the filter equation

\[
\hat{\theta}_{k+1} = \hat{\theta}_k + K \left( u - \hat{\theta}_k \right)
\]  

(4.25)

where \( K \) is a filter gain. Using the discrete on/off values for each thruster, the appropriate thrusters are turned on during the next time interval, \( \Delta t \), after which Equation 4.24 is used to determine which thrusters are activated during the next time interval. During attitude control testing, an appropriate value for the filter gain, \( K \), was found to be between 1/3
and 1/4, although further parametric studies are continuing. An example of the throttle filter process is shown in Appendix E.

**4.2.3. Hybrid Design.** A controller is under development for MR SAT that utilizes both the magnetic coils as well as the thrusters. This method of control is a continuous work in progress; however, two sample models are discussed here.

In the first example of a hybrid control design, the coils and thrusters would simultaneously correct the satellite. This would be done by calculating the torque needed to make the correction, finding the torque that can be produced from the coils, and producing the remainder of the torque via the thrusters. While this method uses both power and propellant, it will not use as much power as the coils alone or as much propellant as the thrusters alone, and the correction should take less time.

Another hybrid model to be considered only uses the thrusters when deemed necessary from a new set of logic. The main constraint for this model would most likely be the time of correction. If the satellite were preparing for a scheduled pass over the ground station, then an attitude correction might be required to be completed more quickly than other corrections. A new set of logic would need to be created that takes into consideration more of the orbital elements in order to predict passes over the ground station. Perhaps propagating ahead in time would allow the controller additional time to make the correction. This would allow the coils, which are unable to correct the attitude as quickly, to be used; as a result, propellant would be conserved.
5. RESULTS

5.1. INTRODUCTION

The computer simulation created to simulate the UMR SAT attitude control law was written using the MATLAB programming language. The simulation software was organized into multiple functions rather than one large file to provide easier software updating and modifying. The initial development effort focused on considering MR SAT and MRS SAT in coupled flight. Extensions to MR SAT or MRS SAT in formation flight only require minor modifications. Before the attitude control algorithms are loaded on the onboard computer, they will be translated to the C programming language.

5.2. SIMULATION STRUCTURE

Each of the simulations were run using a Runge-Kutta 4th order integrator with constants presented in Table 5.1.

The relative tolerance and absolute tolerance for the simulations are both 1e-9, and the fixed step size of the integrator is one second for the magnetic coil controller and one-tenth of a second for the thruster controller.

<table>
<thead>
<tr>
<th>Table 5.1. Runge-Kutta Constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>b</td>
</tr>
<tr>
<td>w</td>
</tr>
</tbody>
</table>

5.2.1. Coupled Flight Simulation. Once the coupled MR SAT and MRS SAT are released from the launch vehicle, the first priority of the ADAC system is to correct the
tip-off error. This small error usually consists of unwanted angular velocity imparted by separation from the launch vehicle. In order to conserve propellant, this error will be corrected using the magnetic coils. Table 5.2 outlines the required inputs and outputs for each function of the simulation and Figure 5.1 shows a flow chart describing the simulation. For the magnetic coil simulation, the propagated states (from the EOM function) replace the attitude determination measurements that will be taken when MR SAT is in orbit. The time step, $\Delta t$, for the simulation is set at one second.

5.2.2. Formation Flight Simulation. Once the tip-off error is stabilized and the satellites separate, MR SAT will have to execute many corrections in short periods of time. Because the thrusters are able to correct the satellite much more quickly than the magnetic coils, they will provide the main method of control during the Formation Flight phase. Figure 5.2 shows a flow chart describing this simulation, and Table 5.3 outlines the required inputs and outputs of each function used. For the thruster simulation, the propagated states (from the EOM function) replace the attitude determination measurements that will be taken when MR SAT is in orbit. The time step interval, $\Delta t$, for the simulation is set at one-tenth of a second.

### Table 5.2. Functions in Coupled Flight Simulation

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Found in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Attitude</td>
<td>$X$</td>
<td>$q$, $\omega$</td>
<td>Controllawtest.m</td>
</tr>
<tr>
<td>Desired Attitude</td>
<td>$x_d$</td>
<td>$q_d$, $\omega_d$</td>
<td>Controllawtest.m</td>
</tr>
<tr>
<td>Attitude Error Angle &gt; $5^\circ$</td>
<td>$q$, $q_d$, $\omega$</td>
<td>YES/NO</td>
<td>Controllawtest.m</td>
</tr>
<tr>
<td>Guidance Law</td>
<td>$q$, $q_d$, $\omega$</td>
<td>$\tau$</td>
<td>Guidancelaw.m</td>
</tr>
<tr>
<td>Coil Logic</td>
<td>$\tau$</td>
<td>$I_c$</td>
<td>Coils.m</td>
</tr>
<tr>
<td>EOMs</td>
<td>$I_c$, $q$, $\Delta t$</td>
<td>$x$, $t$</td>
<td>Controleom.m</td>
</tr>
</tbody>
</table>
Figure 5.1. Simulation Flow Chart (Coupled Flight)

Figure 5.2. Simulation Flow Chart (Formation Flight)
Table 5.3. Functions in Formation Flight Simulation

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Found in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Attitude</td>
<td>X</td>
<td>q, ω</td>
<td>Controllawtest.m</td>
</tr>
<tr>
<td>Desired Attitude</td>
<td>qₜ, ωₜ</td>
<td>qₜd, ωₜ</td>
<td>Controllawtest.m</td>
</tr>
<tr>
<td>Attitude Error Angle &gt; 5º</td>
<td>q, qₜd, ω</td>
<td>YES/NO</td>
<td>Controllawtest.m</td>
</tr>
<tr>
<td>Guidance Law</td>
<td>q, qₜd, ω</td>
<td>τ</td>
<td>Guidancelaw.m</td>
</tr>
<tr>
<td>Throttle Filter</td>
<td>τ</td>
<td>θ</td>
<td>GetThrottle.m</td>
</tr>
<tr>
<td>Thruster Logic</td>
<td>θ, θₖ</td>
<td>u, ³θₖ₊₁</td>
<td>Controllawtest.m</td>
</tr>
<tr>
<td>EOMs</td>
<td>u, q, Δt</td>
<td>x, t</td>
<td>Controleom.m</td>
</tr>
</tbody>
</table>

Once complete, the hybrid controller should also be ideal for this phase of the mission because it would control the satellite quickly while conserving more propellant than the thruster controller alone.

5.3. PROGRAM VERIFICATION

To establish basic functionality of the software, the guidance law, magnetic coil controller, and thruster controller were first verified using simulations without perturbations and without taking the attitude error angle into consideration. Additional verifications and validations are advisable but beyond the scope of this work.

5.3.1. Attitude Guidance Law Verification. The guidance law is initiated with an initial attitude orientation and angular velocity as well as the desired attitude orientation, and it outputs the torque needed to rotate the satellite to the desired attitude. Two control gains were used to tune the controller and were applied to Equations 4.10 and 4.13 such that

\[
v_ω = ω = K_1\left(q₀ν = e₀ν - (ν × νₚ)\right)
\]  

(5.1)
Using gains $K_1$ and $K_2$ of 0.2 and 0.1, respectively, the controller performed well within the requirements. To verify that the guidance law functions properly, test simulations were performed as follows.

In the first simulation, the initial and final attitudes were selected as different orientations, and the initial and final angular velocities were both set to zero as shown in Table 5.4.

<table>
<thead>
<tr>
<th>Initial Conditions</th>
<th>Final Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>0’</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>0’</td>
</tr>
<tr>
<td>$\psi$</td>
<td>0’</td>
</tr>
<tr>
<td>$\omega_x$</td>
<td>0/sec</td>
</tr>
<tr>
<td>$\omega_y$</td>
<td>0/sec</td>
</tr>
<tr>
<td>$\omega_z$</td>
<td>0/sec</td>
</tr>
</tbody>
</table>

Figure 5.3 shows that the satellite rotates from an initial attitude to a desired attitude when the torque from the guidance law is used as the control torque input to the equations of motion. The spike in the angular velocity represents the torque being added to the system.

For the second simulation, the initial and final attitudes were the same as Test 1, and the initial angular velocity was given a nonzero value as shown in Table 5.5.
Figure 5.3. Attitude Position Change with No Initial Angular Velocity

![Graphs showing attitude changes over time](image)

Table 5.5. Guidance Law Verification Test 2

<table>
<thead>
<tr>
<th></th>
<th>Initial Conditions</th>
<th>Final Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>φ</td>
<td>0°</td>
<td>0°</td>
</tr>
<tr>
<td>Θ</td>
<td>0°</td>
<td>0°</td>
</tr>
<tr>
<td>ψ</td>
<td>0°</td>
<td>0°</td>
</tr>
<tr>
<td>ωₓ</td>
<td>1°/sec</td>
<td>0°/sec</td>
</tr>
<tr>
<td>ωᵧ</td>
<td>3°/sec</td>
<td>0°/sec</td>
</tr>
<tr>
<td>ωᶻ</td>
<td>2°/sec</td>
<td>0°/sec</td>
</tr>
</tbody>
</table>

Figure 5.4 shows that the satellite rotates until the angular velocity asymptotically approaches zero, and the attitude position returns to the desired configuration.
Figure 5.4. No Attitude Position Change with Nonzero Initial Angular Velocity

Finally, a third simulation was run in which the initial and final attitudes and angular velocities were different as shown in Table 5.6.

<table>
<thead>
<tr>
<th></th>
<th>Initial Conditions</th>
<th>Final Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>0°</td>
<td>45°</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>0°</td>
<td>45°</td>
</tr>
<tr>
<td>$\psi$</td>
<td>0°</td>
<td>45°</td>
</tr>
<tr>
<td>$\omega_X$</td>
<td>1°/sec</td>
<td>0°/sec</td>
</tr>
<tr>
<td>$\omega_Y$</td>
<td>3°/sec</td>
<td>0°/sec</td>
</tr>
<tr>
<td>$\omega_Z$</td>
<td>2°/sec</td>
<td>0°/sec</td>
</tr>
</tbody>
</table>
Figure 5.5 shows that the satellite rotates and the angular velocity goes to zero along all three directions when the torque from the guidance law is used as the control torque input to the equations of motion.

Figure 5.5. Attitude Position Change with Nonzero Initial Angular Velocity

These three test simulation formats were run multiple (~10) times using different values for the current attitude and angular velocity, as well as the desired attitude. In each case, the guidance law was able to provide the necessary torque to successfully control the system.
5.3.2. Attitude Control Logic Verification. Much like the guidance law, the control logic was verified by running simulations without any perturbations. The magnetic coil model was first verified followed by the thruster model.

5.3.2.1 Magnetic coil controller verification. The magnetic coil controller operates by computing the required currents so that the magnetic coils produce the torque requested by the guidance law. Before the magnetic coil controller is initiated, the guidance law computes the needed torque. This torque is sent to the magnetic coil controller along with the current and desired attitudes. Using the method given in Section 4.2.1, values for the current through each coil are found and then scaled to hardware specifications. Typically the necessary torque cannot be obtained (because it will be higher than what the coils can produce); therefore, the magnetic coils will take longer to correct the attitude error compared with the guidance law. The magnetic coil controller then computes the correction torque that is produced by sending the currents to the coils, and it uses that as the control torque in the equations of motion.

When finding the magnetic field vector from the magnetic field model, a set of orbital parameters were defined as shown in Table 5.7.

<table>
<thead>
<tr>
<th>Table 5.7. Orbital Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
</tr>
<tr>
<td>$e$</td>
</tr>
<tr>
<td>$i$</td>
</tr>
<tr>
<td>$w$</td>
</tr>
<tr>
<td>$\Omega$</td>
</tr>
</tbody>
</table>

In the first simulation, the initial and final attitudes were selected as different orientations, and the initial angular velocities were both set to zero as shown in Table 5.8.
Table 5.8. Coil Controller Verification Test 1

<table>
<thead>
<tr>
<th></th>
<th>Initial Conditions</th>
<th>Final Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>$0^\circ$</td>
<td>$45^\circ$</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>$0^\circ$</td>
<td>$45^\circ$</td>
</tr>
<tr>
<td>$\psi$</td>
<td>$0^\circ$</td>
<td>$45^\circ$</td>
</tr>
<tr>
<td>$\omega_x$</td>
<td>$0$/sec</td>
<td>$0$/sec</td>
</tr>
<tr>
<td>$\omega_y$</td>
<td>$0$/sec</td>
<td>$0$/sec</td>
</tr>
<tr>
<td>$\omega_z$</td>
<td>$0$/sec</td>
<td>$0$/sec</td>
</tr>
</tbody>
</table>

Figure 5.6 shows that the satellite rotates from an initial attitude to a desired attitude when the currents through the magnetic coils are used to obtain the control torque for the equations of motion. Figure 5.7 shows the amount of desired input torque produced by the coils.

Figure 5.6. Attitude Position Change with No Initial Angular Velocity
In the second simulation, the initial and final attitudes were the same, and the initial angular velocity was set to a nonzero value as shown in Table 5.9.

Table 5.9. Coil Controller Verification Test 2

<table>
<thead>
<tr>
<th></th>
<th>Initial Conditions</th>
<th>Final Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>$0^\circ$</td>
<td>$0^\circ$</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>$0^\circ$</td>
<td>$0^\circ$</td>
</tr>
<tr>
<td>$\psi$</td>
<td>$0^\circ$</td>
<td>$0^\circ$</td>
</tr>
<tr>
<td>$\omega_x$</td>
<td>$1$/sec</td>
<td>$0$/sec</td>
</tr>
<tr>
<td>$\omega_y$</td>
<td>$3$/sec</td>
<td>$0$/sec</td>
</tr>
<tr>
<td>$\omega_z$</td>
<td>$2$/sec</td>
<td>$0$/sec</td>
</tr>
</tbody>
</table>
Figure 5.8 shows that the satellite rotates until the angular velocity asymptotically approaches zero and the attitude returns to the desired configuration. Figure 5.7 shows the amount of desired input torque produced by the coils.

![Figure 5.8. No Attitude Position Change with Nonzero Initial Angular Velocity](image)

Finally, a third simulation was run in which the initial and final attitudes and angular velocities were different as shown in Table 5.10.
Figure 5.9. Desired and Produced Torque

Table 5.10. Coil Controller Verification Test 3

<table>
<thead>
<tr>
<th></th>
<th>Initial Conditions</th>
<th>Final Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi )</td>
<td>0(^\circ)</td>
<td>45(^\circ)</td>
</tr>
<tr>
<td>( \Theta )</td>
<td>0(^\circ)</td>
<td>45(^\circ)</td>
</tr>
<tr>
<td>( \psi )</td>
<td>0(^\circ)</td>
<td>45(^\circ)</td>
</tr>
<tr>
<td>( \omega_x )</td>
<td>1(^{/})sec</td>
<td>0(/)sec</td>
</tr>
<tr>
<td>( \omega_y )</td>
<td>3(/)sec</td>
<td>0(/)sec</td>
</tr>
<tr>
<td>( \omega_z )</td>
<td>2(/)sec</td>
<td>0(/)sec</td>
</tr>
</tbody>
</table>

Figure 5.10 shows that the satellite rotates and the angular velocity goes to zero along all three directions when the currents through the magnetic coils are used to obtain
the control torque for the equations of motion. Figure 5.7 shows the amount of desired input torque produced by the coils.

Figure 5.10. Attitude Position Change with Nonzero Initial Angular Velocity

These three test simulation formats were run multiple (~10) times using different values for the current attitude and angular velocity, as well as the desired attitude. In each case, the magnetic coil controller was able to deliver the necessary torque to successfully control the system.
5.3.2.2 Thruster controller verification. The thruster controller operates by selecting which thrusters to activate so that the torque required by the guidance law is produced. Before the thruster controller is initiated, the guidance law computes the needed torque. This torque is sent to the thruster logic algorithm to determine the throttle setting of each thruster. These throttle settings are then sent through a filter to determine which thrusters should be activated. Often the necessary torque cannot be obtained all at once, and therefore the thruster controller requires more time to correct the attitude compared with the guidance law. The throttle filter gain, introduced in Section 4.2.2.2, was set to 0.25 with an initial filter estimate of 0.5.

In the first simulation, the initial and final attitudes were selected as different orientations, and the initial and final angular velocities were both set to zero as shown in Table 5.11.
Table 5.11. Thruster Controller Verification Test 1

<table>
<thead>
<tr>
<th></th>
<th>Initial Conditions</th>
<th>Final Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>$0^\circ$</td>
<td>$45^\circ$</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>$0^\circ$</td>
<td>$45^\circ$</td>
</tr>
<tr>
<td>$\psi$</td>
<td>$0^\circ$</td>
<td>$45^\circ$</td>
</tr>
<tr>
<td>$\omega_x$</td>
<td>$0/\text{sec}$</td>
<td>$0/\text{sec}$</td>
</tr>
<tr>
<td>$\omega_y$</td>
<td>$0/\text{sec}$</td>
<td>$0/\text{sec}$</td>
</tr>
<tr>
<td>$\omega_z$</td>
<td>$0/\text{sec}$</td>
<td>$0/\text{sec}$</td>
</tr>
</tbody>
</table>

Figure 5.12 shows that the satellite rotates from an initial attitude to a desired attitude when the thrusters are used to produce the control torque to the equations of motion. Figure 5.13 and Figure 5.14 show the amount of desired input torque produced by the thrusters. Figure 5.13 assumes that the thrusters can be throttled, and Figure 5.14 shows the torque produced using the throttle filter.

Figure 5.12. Attitude Position Change with No Initial Angular Velocity
Figure 5.13. Desired and Produced Torque (Throttling)

Figure 5.14. Desired and Produced Torque (No Throttling)
In the second simulation, the initial and final attitudes were the same, and the initial angular velocity was set to a nonzero value as shown in Table 5.12.

<table>
<thead>
<tr>
<th></th>
<th>Initial Conditions</th>
<th>Final Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>φ</td>
<td>0°</td>
<td>0°</td>
</tr>
<tr>
<td>Θ</td>
<td>0°</td>
<td>0°</td>
</tr>
<tr>
<td>ψ</td>
<td>0°</td>
<td>0°</td>
</tr>
<tr>
<td>ω_x</td>
<td>1°/sec</td>
<td>0°/sec</td>
</tr>
<tr>
<td>ω_y</td>
<td>3°/sec</td>
<td>0°/sec</td>
</tr>
<tr>
<td>ω_z</td>
<td>2°/sec</td>
<td>0°/sec</td>
</tr>
</tbody>
</table>

Figure 5.15 shows that the satellite rotates until the angular velocity asymptotically approaches zero and the attitude returns to the desired configuration. Figure 5.16 and Figure 5.17 show the amount of desired input torque produced by the thrusters. Figure 5.16 assumes that the thrusters can be throttled, and Figure 5.17 shows the torque produced using the throttle filter.

Finally, a third simulation was run in which the initial and final attitudes and angular velocities were different as shown in Table 5.13. Figure 5.18 shows that the satellite rotates and the angular velocity goes to zero along all three directions when the thrusters are used to obtain the control torque for the equations of motion. Figure 5.19 and Figure 5.20 show the amount of desired input torque produced by the thrusters. Figure 5.19 assumes that the thrusters can be throttled, and Figure 5.20 shows the torque produced using the throttle filter.
Figure 5.15. No Attitude Position Change with Nonzero Initial Angular Velocity

Figure 5.16. Desired and Produced Torque (Throttling)
Figure 5.17. Desired and Produced Torque (No Throttling)

Figure 5.18. Attitude Position Change with Nonzero Initial Angular Velocity
Figure 5.19. Desired and Produced Torque (Throttling)

Figure 5.20. Desired and Produced Torque (No Throttling)
Table 5.13. Thruster Controller Verification Test 3

<table>
<thead>
<tr>
<th></th>
<th>Initial Conditions</th>
<th>Final Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>$0^\circ$</td>
<td>$45^\circ$</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>$0^\circ$</td>
<td>$45^\circ$</td>
</tr>
<tr>
<td>$\psi$</td>
<td>$0^\circ$</td>
<td>$45^\circ$</td>
</tr>
<tr>
<td>$\omega_X$</td>
<td>1'/sec</td>
<td>0'/sec</td>
</tr>
<tr>
<td>$\omega_Y$</td>
<td>3'/sec</td>
<td>0'/sec</td>
</tr>
<tr>
<td>$\omega_Z$</td>
<td>2'/sec</td>
<td>0'/sec</td>
</tr>
</tbody>
</table>

These three test simulation formats were run multiple (~10) times using different values for the current attitude and angular velocity, as well as the desired attitude. In each case, the thruster controller was able to deliver the necessary torque to successfully control the system.

5.4. PERFORMANCE RESULTS

Euler angles suffice for observing the difference between the current and desired attitude, but an attitude error angle has been defined using quaternion notation to provide the angle needed to rotate the satellite to the desired attitude from the current attitude. Because the attitude control requirement for UMR SAT is an error less than $5^\circ$ for MR SAT ($10^\circ$ for MRS SAT), defining one angle that defines the error is preferred to determining the total error from three Euler angles.

5.4.1. Thruster Controller Performance. Using the thruster controller simulation with the conditions shown in Table 5.13, the resulting attitude error angle can be seen in Figure 5.21. Figure 5.22 shows the number of thrusters active throughout the simulation. Notice that for any given time at most five thrusters are firing. Figure 5.23 shows the cumulative thruster firings during the five-minute simulation. It is seen that initially, when the attitude error is relatively large, more thruster firings are needed, but once the attitude motion becomes stable, fewer firings are needed to maintain the attitude. Note that the rate of thruster firings never reaches zero. This is because the thrusters cannot be throttled, which would fully eliminate the angular velocity. Further studies are
being conducted to determine methods for minimizing the amount of attitude maintenance required by the thruster control system.

Figure 5.21. Attitude Error Angle (Thruster Controller)

By introducing an attitude error, the control logic can easily decide whether a control torque is needed. As long as the attitude error is within the tolerance, no torque is needed. Figure 5.21 shows that within two minutes, the attitude error is driven to less than the maximum allowable error of 5°.
Figure 5.22. Active Thrusters During Control

Figure 5.23. Total Thruster Firings During Control
When the control logic is modified such that there are no control inputs (thrust) whenever the attitude error is less than 5°, the number of thrusts executed is reduced from 2006 thrusts to 899 (again using the initial conditions from Table 5.13). Figure 5.24 shows that the thruster controller performs to within the required specifications when this new attitude error constraint strategy is applied.

![Figure 5.24. Attitude Error Angle (Thruster Controller)](image)

**5.4.2. Magnetic Coil Controller Performance.** Using the coil controller simulation with the conditions shown in Table 5.13, the resulting attitude error angle can be seen in Figure 5.25. For this case, the attitude error goes below 10° after
approximately three hours and below 5° just past four hours. The coils require a longer period to stabilize the attitude due to their lower torque capabilities and their inability to produce a torque along the magnetic field direction. This extended control time is not expected to pose problems for the UMR SAT project because the coils will primarily be used during Detumble mode. Figure 5.26 shows the required current for each coil during the control simulation.

When the control logic is modified such that there are no control inputs (currents) whenever the attitude error is less than 5°, the average amp-hours used over the five hour simulation is increased from 194.4 amp-hours to 2869.5 (again using the initial conditions from Table 5.13). Figure 5.27 shows that this is because the magnetic coil controller does not perform as well when this method is applied because once the satellite is within the attitude requirement and no control is applied, the satellite begins to drift quickly. A new strategy is being formulated to help conserve power while not sacrificing controllability.
Figure 5.26. Current Needed For Control (Coil Controller)

Figure 5.27. Attitude Error Angle (Coil Controller)
6. CURRENT STATUS AND FUTURE DEVELOPMENT

6.1. CURRENT STATUS

The modeling of the control system for MR SAT has provided valuable information necessary for the future of the UMR SAT mission. Simulating the MR SAT EOMs in quaternion form has proven particularly useful. Quaternion notation avoids the trigonometric singularities that Euler angle notation typically encounters. Analyzing data in Euler angle notation is much easier to visualize; therefore, the data were transformed from Euler angle format to quaternion notation to execute simulations, and output data were transformed back to Euler angles for creating figures and performing analysis.

A guidance law was specifically designed to determine the necessary applied control moment, $N$, to correct the attitude from a current attitude, $q$, and angular velocity, $\omega$, to a desired attitude, $q_d$, and angular velocity of zero.

After the applied control torque is calculated, the control logic algorithm determines the method of control to produce the needed torque. This torque is produced using either the magnetic coils or eight cold gas thrusters aboard MR SAT. MRS SAT is able to create a torque using only magnetic coils (i.e., MRS SAT has no thruster capability onboard).

The guidance law and control methods were verified using three test simulations. These simulations were run for various initial conditions and final conditions. One test simulation assumed a current attitude with no angular velocity, and rotated the spacecraft to a desired position with no angular velocity. Another test simulation was initiated with a nonzero angular velocity, but the initial and desired attitudes were the same. The third verification test simulation was initiated with an angular velocity and differing initial and desired positions. An attitude error angle was introduced, enabling the control logic to determine when an attitude correction was required.

6.2. FUTURE DEVELOPMENT

With the guidance law and control logic algorithm verified, the next step is translating the algorithms to the C programming language and integrating them with the attitude determination software. Along with this, the updated (final) values for the
moments of inertia, mass, and other satellite parameters will need to be updated. The ADAC software will then be added to the orbit determination and control software. This will create a “black box” software package for the UMR SAT mission.

This software package will provide a test bed for demonstrating attitude determination and control as well as orbit determination and control using a high fidelity model with perturbations. Once this test bed is complete, a Monte Carlo analysis can analyze hundreds or thousands of possible tip-off error possibilities and ΔV corrections (for the Formation Flight phase) for various orbits. As details become available, specific mission corrections could also be analyzed.

One key development for future implementation is the construction of a robust hybrid control logic algorithm. As discussed in Section 4.2.3., such a controller has the potential of conserving time, power, and propellant, which is valuable for most any satellite mission. The hybrid model may consist of logic that uses the coils as much as possible to conserve propellant, and the thrusters for larger maneuvers and corrections that are time-depandant.

The θ-D controller is under development for use in the ADAC system to replace the guidance law. Unlike the guidance law, the θ-D controller uses optimal control theory for determining the required control torque. The θ-D controller is also better suited for determining the currents for the magnetic coils because it can easily account for the fact that the magnetic coils cannot produce a torque perpendicular to the magnetic field.
APPENDIX A

MATLAB SOURCE CODE
### A. MATLAB SOURCE CODE

<table>
<thead>
<tr>
<th>File Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ControlLawFixed.m</td>
<td>60</td>
</tr>
<tr>
<td>Rk4fixed.m</td>
<td>63</td>
</tr>
<tr>
<td>ControlEOMFixed.m</td>
<td>63</td>
</tr>
<tr>
<td>GuidanceLaw.m</td>
<td>65</td>
</tr>
<tr>
<td>Coils.m</td>
<td>65</td>
</tr>
<tr>
<td>GetThrottle.m</td>
<td>66</td>
</tr>
<tr>
<td>Thrusters.m</td>
<td>68</td>
</tr>
<tr>
<td>EulerToQuaternion.m</td>
<td>69</td>
</tr>
<tr>
<td>QuaternionToEuler.m</td>
<td>69</td>
</tr>
<tr>
<td>Quatmult.m</td>
<td>70</td>
</tr>
<tr>
<td>Magfld.m</td>
<td>70</td>
</tr>
<tr>
<td>RAGREN.m</td>
<td>75</td>
</tr>
<tr>
<td>Orbit.m</td>
<td>76</td>
</tr>
</tbody>
</table>
ControlLawFixed.m

% This will simulate the satellite for part of its mission and the result will be that the satellite needs to rotate some amount and the control system needs to apply the control using the thrusters, coils, or a combination of both.

close all; clc; clear all; format compact; format long g

% Moments of Inertia for the satellite
global I
I = [0.421 0 0; 0 0.421 0; 0 0 0.351]; % Units: kg m^2

global d2r
d2r = pi/180; % Converts Degrees to Radians

k_g = 10*[.02 .01]; % Weighting vector for GL
k_g_c = .05*[.02 .011]; % Weighting vector for coils
k_g_t = 5*[.02 .01]; % Weighting vector for thrusters

% Choose Control Method
global control_method
for control_method = 2
    control_method
if control_method==1; control_name = 'Guidance'; end
if control_method==2; control_name = 'Coils'; end
if control_method==3; control_name = 'Thrusters'; end
if control_method==4; control_name = 'Hybrid'; end

% Set up the system time parameters
T0 = 0; % Units: seconds Initial Time
Tf = 300*60; % Units: seconds Ending Time

% Chosen current and desired Euler Angles in Radians
phic=0*d2r; thetac=0*d2r; psic=0*d2r;
q = eulertoquaterion(phic, thetac, psic);
phid=45*d2r; thetad=45*d2r; psid=45*d2r;
qd = eulertoquaterion(phid, thetad, psid);

%omega0 = 0*[1;3;2]*d2r; % For Tip Off Error
omega0 = [1;3;2]*d2r; % For 0*10*rand(3,1)*d2r;
omega = 0*rand(3,1)*d2r; % Desired angular velocity should be zero

% Setting the initial position and velocity parameters
clear x; clear t
x0 = [q; omega0];
x(1,:) = x0; % Initial Position and Velocity
t(1,1) = T0; % Initial Time
xtmp(:,1) = x0; ttmp(1,1) = 0;

OPTIONS = odeset('RelTol', 1e-9, 'AbsTol', 1e-9);

% Timesteps
dt_g = 1; % Units: seconds Guidance Law
dt_c = 1; % Units: seconds Coils
dt_t = .1; % Units: seconds Thrusters
% Initial Condition for Throttle Filter
filter_hat = .5*ones(8,1); % Initial Estimate
k_t = 1; % Filter Counter
K_t = 1/4; % Weighting Factor (for throttle filter)

if control_method == 1 % Guidance Law
    dt = dt_g;
tspan = [t0:dt:tf]; % Simulation Timespan
for i = 2:size(tspan,2)
    a_err = acos(2*(x(end,1:4)*qd).^2-1)*180/pi; % Attitude Error Angle
    % Simulate Timestep
    [tmp, xtmp] = ode113(@controleomfixed,[0 dt],xtmp(end,:)',OPTIONS,qd,k_g); % Integration
    x(i,:) = xtmp(end,:);
    t(i,1) = tspan(i);
    tau(:,i) = guidancelaw(x(end,1:4)',qd,x(end,5:7)',k_g); % Units: N-m
    x(i,:) = rk4fixed(@controleomfixed,t(i-1,1),x(end,:)',dt,tau(:,i));
    t(i,1) = tspan(i);
tspan(i)
end
end

if control_method == 2 % Magnetic Coils
    dt = dt_c;
tspan = [t0:dt:tf]; % Simulation Timespan
for i = 2:size(tspan,2)
    a_err = acos(2*(x(end,1:4)*qd).^2-1)*180/pi; % Attitude Error Angle
    if a_err >= 5
        % Calculate torque needed for correction
        tau(:,i) = guidancelaw(x(end,1:4)',qd,x(end,5:7)',k_g_c); % Units: N-m
        current(:,i) = coils(tau(:,i),x(end,1:4)',qd,x(end,5:7)',t(i-1,1));
    else
        x(i,:) = rk4fixed(@controleomfixed,(i-1,1),x(end,:)',dt,zeros(3,1));
        t(i,1) = tspan(i);
tspan(i)
    end
end
end

total_current_used = sum(sum(abs(current)))
end

if control_method == 3 % Using Torque from Thrusters
    dt = dt_t;
tspan = [t0:dt:tf]; % Simulation Timespan
counter = 2;
for i = 2:size(tspan,2)
    a_err = acos(2*(xtmp(1:4)*qd).^2-1)*180/pi; % Attitude Error Angle
    if a_err >= 5
        % Calculate torque needed for correction
        tau(:,i) = guidancelaw(xtmp(1:4),qd,xtmp(5:7),k_g_t); % Units: N-m
        [filter(:,k_t) = GetThrottle([0;0],tau(:,i));

    end
end

% Calculate Throttle Settings
filter(:,k_t) = GetThrottle([0;0],tau(:,i));

% Thruster Logic
for j=1:8
    if filter(j,k_t) > filter_hat(j,k_t)
        u(j,k_t)=1; % Turns thruster on
    else
        u(j,k_t)=0; % Leaves thruster off
    end
end

% Simulate Timestep
xtmp = rk4fixed(@controleomfixed,ttmp(i-1,1),xtmp(:,dt,u(:,k_t));
tmp(i,1) = tspan(i);

if counter == 10
    x(i/10+1,:) = xtmp;
    t(i/10+1,1) = tspan(i);
    counter = 0;
end

% Update Filter Estimate
filter_hat(:,k_t+1) = filter_hat(:,k_t) + K_t*(u(:,k_t)-filter_hat(:,k_t));
k_t=k_t+1;
counter = counter + 1;
else
    xtmp = rk4fixed(@controleomfixed,ttmp(i-1,1),xtmp(:,dt,zeros(8,1));
tmp(i,1) = tspan(i);
    if counter == 10
        x(i/10+1,:) = xtmp;
        t(i/10+1,1) = tspan(i);
        counter = 0;
    end
    counter = counter + 1;
end
end

number_of_thrusts = sum(sum(u))
figure % Plot of thrusts by each thruster
    subplot(2,4,1);plot(tspan(1:end-1),filter(1,:))
    subplot(2,4,2);plot(tspan(1:end-1),filter(2,:))
    subplot(2,4,3);plot(tspan(1:end-1),filter(3,:))
    subplot(2,4,4);plot(tspan(1:end-1),filter(4,:))
    subplot(2,4,5);plot(tspan(1:end-1),filter(5,:))
    subplot(2,4,6);plot(tspan(1:end-1),filter(6,:))
    subplot(2,4,7);plot(tspan(1:end-1),filter(7,:))
    subplot(2,4,8);plot(tspan(1:end-1),filter(8,:))
xlabel('time in .1 seconds'),ylabel('filter throttle')
end

t = t/3600; % This is time in hours
    t = t*60; % This is time in minutes

% Changing the quaternions to Euler Angles
for i=1:1/dt:size(x,1)
    [phi(i) theta(i) psi(i)] = quaternion2euler(x(i,1:4));
end
phid(1:size(t)) = phid; thetad(1:size(t)) = thetad; psid(1:size(t)) = psid;

%%%%%%% ANALYSIS COMPLETE -FIGURES %%%%%%%%
figure %This figure is for the Euler Angles and omega
subplot(2,3,1);plot(t,phid/d2r,y,t(1:1/dt:end),phid(1:1/dt:end)/d2r,'b:');
grid on;xlabel('Time, (minutes)');ylabel('phi (circ)');
z = sprintf('%s',control_name);title(z)
subplot(2,3,2);plot(t,thetad/d2r,y,t(1:1/dt:end),thetad(1:1/dt:end)/d2r,'b:');
grid on;xlabel('Time, (minutes)');ylabel('theta (circ)');
z = sprintf('%s',control_name);title(z)
subplot(2,3,3);plot(t,psid/d2r,y,t(1:1/dt:end),psid(1:1/dt:end)/d2r,'b:');
z = sprintf('dt = %.4f',dt);title(z)
subplot(2,3,4);plot(t,x(:,5)/d2r,y,t(1:1/dt:end),x(:,5)(1:1/dt:end)/d2r,'b:');
grid on;xlabel('Time, (minutes)');ylabel('omega_x (circ/sec)');
subplot(2,3,5);plot(t,x(:,6)/d2r,y,t(1:1/dt:end),x(:,6)(1:1/dt:end)/d2r,'b:');
grid on;xlabel('Time, (minutes)');ylabel('omega_y (circ/sec)');
subplot(2,3,6);plot(t,x(:,7)/d2r,y,t(1:1/dt:end),x(:,7)(1:1/dt:end)/d2r,'b:');
grid on;xlabel('Time, (minutes)');ylabel('omega_z (circ/sec)');
figure %This is the figure for the attitude error angle
plot(t,acos(2*(x(:,1:4)*qd).^2-1)*180/pi)
grid on;xlabel('time (minutes)');ylabel('Attitude Error (circ)');

Rk4fixed.m
% Given to me by Mike Dancer
function Ynew = rk4fixed(EOMFile,t0,y0,dT,varargin)
    % Runge-Kutta 4th order constants
    a = [0 .5 .5 1];
    b = [0 .5 .5 1];
    w = [1 2 2 1]/6;
    k(:,1) = feval(EOMFile,t0,y0,varargin{:});
    for i = 2:4
        k(:,i) = feval(EOMFile,t0+a(i)*dT,y0+b(i)*dT*k(:,i-1),varargin{:});
    end
    Ynew = y0+dT*k*w;

ControlEOMFixed.m
% % M R SAT EQUATIONS OF MOTION% % % % % % % %
function xdot = controlemomfixed(t,x,u)
    global I;global control_method;global d2r
    xdot = 0*x;
    if control_method == 1
        % Torque from Guidance Law
        T = u;  % Units: N-m
    end
    if control_method == 2
% Number of turns for each coil
N = [240; 240; 240];

% Cross-Sectional area of each coil
A = [0.0064516; 0.0064516; 0.0064516]; % Units: m^2 from 10 in^2

% Unit Vector Normal Forces produced by each Coil (for the dipole)
Fn_1 = [cos(60*d2r) sin(60*d2r) 0]; % Coil on Panel 2
Fn_2 = [-cos(60*d2r) sin(60*d2r) 0]; % Coil on Panel 3
Fn_3 = [0 0 -1]; % Coil on Bottom Panel
Fn = [Fn_1; Fn_2; Fn_3];

% Finding the magnetic field vector in the inertial frame
[theta_dot, R, V, theta] = orbit(t);

% Conversion from Sec to days for times
times = 1.5253E9; % seconds (1.5253E9 since 1-1-06)

% Degree of accuracy (2-10) (Goddard Recommends 10)
mln = 10;

% Current Values (seconds)
leap_seconds = 22;

% Magfld values
[magfld_v, mag_magfld] = magfld(times, mln, R, leap_seconds);
magfld_v = magfld_v * 1e-7; % Units: Tesla (Inertial Frame)

% Transfer matrix from Inertial to BFF
q = x(1:4);
transfer = 2*(q(1)^2 + q(2)^2) - 1 2*(q(2)*q(3) + q(1)*q(4))...
2*(q(2)*q(4) - q(1)*q(3)) ;...
2*(q(2)*q(3) - q(1)*q(4)) 2*(q(1)*q(2) + q(3)*q(2) - 1 2*(q(3)*q(4) + q(1)*q(2)) ;
2*(q(2)*q(4) + q(1)*q(3)) 2*(q(3)*q(4) - q(1)*q(2)) 2*(q(1)*q(2) + q(4)*q(2) - 1];
magfld_v = transfer * magfld_v; % Units: Tesla (BFF)

V = [N(1)*A(1)*cross(Fn(1:3,1), magfld_v) ... % Coil 1
N(2)*A(2)*cross(Fn(1:3,2), magfld_v) ... % Coil 2
N(3)*A(2)*cross(Fn(1:3,3), magfld_v)]; % Units: Tesla - m^2

% Torque from magnetic coils
T = V*u; % Units: N-m
end

if control_method == 3
  % Torque from thrusters (var1 is u)
  T = thrusters(u); % Units: N-m
end

%%%%%%%%%%%%%%%%%%%%%%%%% THIS PART IS JUST INCASE THE CODE RUNS BUT %%%%%%%%%%%%%%%%%%%%
% DOESN'T REPRESENT THE HYBRID CONTROL %%%%%%%%%%%%%%%%%%%%%%%%%
if control_method == 4
  % Torque from Guidance Law
  T = u; % Units: N-m
end

%%%%%%%%%%%%%%%%%% EQUATIONS OF MOTION %%%%%%%%%%%%%%%%%%
xdot(1) = dot(x(2:4), x(5:7))/2;
xdot(2:4) = -(x(1)*x(5:7)+cross(x(5:7), x(2:4)))/2;
xdot(5) = (T(1) + x(6)*x(7))*(I(2,2)-I(3,3))/I(1,1);
xdot(6) = (T(2) + x(5)*x(7))*(I(3,3)-I(1,1))/I(2,2);
xdot(7) = (T(3) + x(5)*x(6))*(I(1,1)-I(2,2))/I(3,3);
**GuidanceLaw.m**

```matlab
%                       -----------------------
%                       - MR SAT Guidance Law -
%                       -----------------------
%Inputs will be the current attitude, desired attitude and angular rate
% current attitude --> q = \langle q_0, v_q \rangle
% desired attitude --> qd = \langle qd_0, v_{qd} \rangle
% angular rate     --> w = \langle w_0, v_w \rangle
%Output will be the torque needed to obtain the desired attitude
% needed torque    --> T = \langle 0, v_n \rangle and v_n = N_x, N_y, N_z
function T = guidancelaw(q,qd,w,k_g) %torque = f(desired, current, omega)

global I

%Error between the current and desired attitude
err = q-qd;

%This is the desired Angular Velocity %Units: rad/s
omega = k_g(1)*(cross(q(2:4),err(2:4)) - err(1)*q(2:4) + q(1)*err(2:4));

%Error between the current and desired angular velocity
err_omega = w - omega; %rad/s

%Needed Torque
T = cross(omega,I*omega) - k_g(2)*err_omega; %Units: N-m
```

**Coils.m**

```matlab
%                       -------------------------
%                       - MR SAT Magnetic Coils -
%                       -------------------------
%Inputs will be the needed torque, current attitude, desired attitude, angular rate, and time (for orbit calculation)
% needed torque    --> tau = [tau_x, tau_y, tau_z]
% current attitude --> q = \langle q_0, v_q \rangle
% desired attitude --> qd = \langle qd_0, v_{qd} \rangle
% angular rate     --> w = \langle w_0, v_w \rangle
% time in orbit     --> t = time in seconds
%Output will be the torque produced by the coils
% produced torque  --> T = [T_x, T_y, T_z]
function I = coils(tau,q,qd,w,t)

global d2r

%Number of turns for each coil
N = [240;240;240];

%Cross-Sectional area of each coil
A = [0.0064516;0.0064516;0.0064516]; %Units: m^2 from 10 in^2

%Unit Vector Normal Forces produced by each Coil (for the dipole)
Fn_1 = [cos(60*d2r) sin(60*d2r) 0]; %Coil on Panel 2
Fn_2 = [-cos(60*d2r) sin(60*d2r) 0]; %Coil on Panel 3
Fn_3 = [0 0 -1]; %Coil on Bottom Panel
```
Fn = [Fn_1; Fn_2; Fn_3];

% Finding the magnetic field vector in the inertial frame
[theta_dot, R, V, theta] = orbit(t);
times = 1.5253E9; % seconds (1.5253E9 since 1-1-06)
mln = 10; % degree of accuracy (2-10) (Goddard Recommends 10)
leap_seconds = 22; % Current Values (seconds)
[magfld_v, mag_magfld] = magfld(times, mln, R, leap_seconds);
magfld_v = magfld_v * 1e-7; % Units: Tesla (Inertial Frame)

% Transfer matrix from Inertial to BFF
transfer = 
\[
\begin{bmatrix}
2(q(1)^2 + q(2)^2) - 1 & 2(q(2)*q(3) + q(1)*q(4)) & 2(q(2)*q(4) - q(1)*q(3)) \\
2(q(2)*q(3) - q(1)*q(4)) & 2(q(1)^2 + q(3)^2) - 1 & 2(q(3)*q(4) + q(1)*q(2)) \\
2(q(2)*q(4) + q(1)*q(3)) & 2(q(3)*q(4) - q(1)*q(2)) & 2(q(1)^2 + q(4)^2) - 1
\end{bmatrix}
\]

magfld_v = transfer * magfld_v; % Units: Tesla (BFF)

% 'Known' Part of the Torque
% [x_component y_component z_component] for each coil
V = 
\[
\begin{bmatrix}
N(1)*A(1)*cross(Fn(1:3,1),magfld_v) & N(2)*A(2)*cross(Fn(1:3,2),magfld_v) & N(3)*A(2)*cross(Fn(1:3,3),magfld_v)
\end{bmatrix}
\]
Units: Tesla m^2

% Two vectors that are perp to magnetic field and orthogonal to each other
Dircos = acos(magfld_v / norm(magfld_v));
[m, i] = max(Dircos);
u = [0 0 0]';
u(i) = 1;

% Find matrix to cancel out the torque in the direction perp. to magnetic field. This is done b/c coils can't produce a torque in this direction
SF = 1e5; % Scaling Factor
R = SF * [v'; u'];

% This is [lambda; I] found from optimal control
answer = inv([(R*V)' eye(3) zeros(2,2) R*V]) *[zeros(3,1); R*tau];

% Current running through each coil
I = answer(3:5); % Units: Amps

% Scale current to hardware capabilities
if max(abs(I)) > 1; I = I / max(abs(I)); end % Units: Amps (Range -1 to 1)

T = V*I; % Tesla meter^2 Amp = N-m

GetThrottle.m

% Written with the help of Mike Dancer
function Throttle = GetThrottle(Fcmd, Tcmd)

% Thruster configuration
Tmax = .025; % Maximum thrust [N]
Rcm = [0 0 .300475/2];
b = .215;
a = b*sqrt(3)/2;
Pos = [a 0 .2703; ... % #1
   a .0829 .1486; ... % #2
   a 0 .0270; ... % #3
   a -.0829 .1486; ... % #4
   0 b .2703; ... % #5
   0 b .0270; ... % #6
   -a 0 .1486; ... % #7
   0 -b .1486'; ... % #8
Pos = Pos-Rcm*ones(1,8);
Dir = [-1 0 0; ... % #1
   -1 0 0; ... % #2
   -1 0 0; ... % #3
   -1 0 0; ... % #4
   0 -1 0; ... % #5
   0 -1 0; ... % #6
   1 0 0; ... % #7
   0 1 0]; % #8
Force = Tmax*Dir(1:2,:);
Torque = Tmax*cross(Pos,Dir);

% Force and Moment contraints
F = Force;
T = Torque;
for i = 1:2
    if Fcmd(i) < 0, F(i,:) = -F(i,:); end
    if Tcmd(i) < 0, T(i,:) = -T(i,:); end
end
if Tcmd(3) < 0, T(3,:) = -T(3,:); end

% Determine thruster throttle settings
M = 1e9; Sf = 1e-6;
A = [1 ones(1,8) zeros(1,8) M*ones(1,5) 0; ... % z row
     zeros(8,1) eye(8) zeros(8,5) ones(8,1); ... % eta1-eta8 rows
     zeros(2,1) F zeros(2,8) zeros(2,3) abs(Sf*Fcmd); ... % Force rows
     zeros(3,1) T zeros(3,8) zeros(3,2) eye(3) abs(Sf*Tcmd)]; % Torque rows
S = [1 10:22];
Ind = find(A(1,:) == M);
for i = 1:length(Ind)
    Col = Ind(i);
    Row = find(A(2:end,Col) == 1)+1;
    A(1,:) = A(1,:) - M*A(Row,:);
end
for k = 1:30
    % Locate pivot point
    [m,Col] = min(A(1,2:end-1));
    Col = Col+1;
    if m >= 0, break; end % Optimal solution found
    Ind = find(A(2:end,Col) > 0)+1;
    [m,Row] = min(A(Ind,end)./A(Ind,Col));
    Row = Ind(Row);
    % Pivot solution vector
    A(Row,:) = A(Row,:)/A(Row,Col);
    for i = 1:size(A,1)
        if i ~= Row, A(i,:) = A(i,:) - A(i,Col)*A(Row,:); end
    end
end
S(Row,1) = Col;
end
R = zeros(22,14);
for i = 1:size(S,1)
    R(S(i),i) = 1;
end
Throttle = R(2:9,:)*A(:,end);

% Adjust throttle settings
if max(Throttle/Sf) > 1, Throttle = Throttle/max(Throttle);
else Throttle = Throttle/Sf; end

Thrusters.m

% ------------------------
% - MR SAT Thrusters -
% ------------------------
%The input is the throttle setting [0 1]
% throttle         --> u  = [u_1,u_2,u_3,u_4,u_5,u_6,u_7,u_8]
%Output will be the torque produced by the coils
% produced torque  --> T  = [T_x,T_y,T_z]

function T = thrusters(u)

%Convert Degrees to Radians
d2r = pi/180;

%Magnitude of Thrust (This is a low estimate)
thrust_mag = 0.025; %Units: N

%Unit Vector Normal Force (Thrust) produced by each thruster
%[x direction;y direction;z direction]
Fn_1 = [-1;0;0]; %normal to Panel 1
Fn_2 = [-1;0;0]; %normal to Panel 1
Fn_3 = [-1;0;0]; %normal to Panel 1
Fn_4 = [-1;0;0]; %normal to Panel 1
Fn_5 = [0;-1;0]; %normal to plane between Panel 2 and Panel 3
Fn_6 = [0;-1;0]; %normal to plane between Panel 2 and Panel 3
Fn_7 = [1;0;0]; %normal to Panel 4
Fn_8 = [0;1;0]; %normal to plane between Panel 5 and Panel 6
Fn = [Fn_1 Fn_2 Fn_3 Fn_4 Fn_5 Fn_6 Fn_7 Fn_8];

%Location of the Center of Gravity (or mass) from origin
cm  = [0.0;0.0;0.300475/2]; %Units: m (Estimate for now)

%Distance to Panel 1 in x-direction
a = tan(60*pi/180)*.215/2; %Units: m

%Distance to plane between Panel 2 and Panel 3 in y-direction
b = .215/2*cos(60*pi/180); %Units: m

%Location of each thruster from origin
%[x direction y direction z direction]
r_t_o = [a 0.2703;... %1
a 0.0829 0.1486;... %2
a 0 0.0270;... %3
a -0.0829 0.1486;... %4
0 b 0.2703;... %5
0 b 0.0270;... %6
-a 0 0.1486;... %7
0 -b 0.1486'; %8 Units: m

% Location of each thruster from cm (This assumes the cm is the origin)
r_t_cm = r_t_o - cm*ones(1,8); %Units: m

% This is the torque produced by each thruster
% each column is a different thruster, row 1: x, row 2: y, row 3: z
torque = thrust_mag*cross(r_t_cm,Fn); % Units: N-m
T = (u'*torque'); % Units: N-m

**EulerToQuaternion.m**

% Rotation from Quaternion to Euler Angles

% The Euler Angles are: phi, theta, psi
% q = [q0;q1;q2;q3]
function [phi theta psi] = quaterniontoeuler(q)
m = [2*(q(1)^2+q(2)^2)-1 2*(q(2)*q(3)+q(1)*q(4)) 2*(q(2)*q(4)-q(1)*q(3));
     2*(q(2)*q(3)-q(1)*q(4)) 2*(q(1)^2+q(3)^2)-1 2*(q(3)*q(4)+q(1)*q(2));
     2*(q(2)*q(4)+q(1)*q(3)) 2*(q(3)*q(4)-q(1)*q(2)) 2*(q(1)^2+q(4)^2)-1];
theta = asin(-m(3,1));
phi = atan2(m(2,1),m(1,1));
while phi<0
    phi=phi+2*pi;
end
psi = atan2(m(3,2),m(3,3));
while psi<0
    psi=psi+2*pi;
end

**QuaternionToEuler.m**

% Rotation from Euler Angles to Quaternion

% The Euler Angles are: phi, theta, psi
function q = eulertoquaterion(phi,theta,psi)
cphi=cos(phi);ct=cos(theta);cpsi=cos(psi);
sphi=sin(phi);st=sin(theta);spsi=sin(psi);

% q=[q0;q1;q2;q3]
qu = [cos(phi/2)*cos(theta/2)*cos(psi/2)+sin(phi/2)*sin(theta/2)*sin(psi/2);... 
cos(phi/2)*cos(theta/2)*sin(psi/2)-sin(phi/2)*sin(theta/2)*cos(psi/2);...
cos(phi/2)*cos(theta/2)*sin(psi/2)+sin(phi/2)*cos(theta/2)*sin(psi/2);...
sin(phi/2)*cos(theta/2)*cos(psi/2)-cos(phi/2)*sin(theta/2)*sin(psi/2);]
q(2:4) = -q(2:4);
Quatmult.m

% Multiplication of two quaternions p*q
% where p = p_0 + p1*i + p2*j + p3*k
% and q = q_0 + q1*i + q2*j + q3*k
% function product = quatmult(p,q)

product = p(1)*q(1)-dot(p(2:4),q(2:4))...  
  + p(1)*q(2:4) + q(1)*p(2:4) + cross(p(2:4),q(2:4));

Magfld.m

%====================================================================
% magfld.m
%====================================================================
%**************************************************************************
% ***NOTE:  This function was donated to the University of Missouri-Rolla *
% Satellite Project by the NASA Goddard Space Flight Center             *
%**************************************************************************
% Function Name: magfld.m
%
% For local use:
% REMOVED GLOBALS and set run_time(5)=1 (input times must be in seconds).
% Does not need tcon40.
% Will load coef1990, coef1995, or coef2000.mat, as needed.
%
% Purpose: This function computes the reference Magnetic Field Vector in
% GCI Coordinates given time and spacecraft position.
%
% Invocation Method: [output_magfld_vectors, flag, magfld_magnitude] =
% magfld(times, mln, pos, leap_seconds);
%
% Argument           I/O  Description
% ------------------- --- ------------------------------------------
% times               (:,1) i  Array of times
% mln                 i   Degree of magnetic field model (2 - 10)
%                       (Goddard Recommends 10)
% pos                  (3,:) i  Spacecraft position vectors (km)
% leap_seconds         i   Number of leap seconds
% output_magfld_vectors (3,:) o  Output magnetic field vectors in GCI (mG)
% flag                 o   Output quality flag
%                       0 = ok
%                       1 = not enough input arguments
%                       2 = input time array is empty
%                       3 = Magnetic field order is invalid
%                       4 = input position array is empty
% magfld_magnitude     o   Magnitude of reference magnetic field (mG)
%
% External References:
% Function Name  Purpose
% ------------------- ------------------------------------------
% tcon40 (not used) Computes seconds since 9/1/57 from a calendar date
% coef1990.mat    Magnetic field coefficients and derivatives based
%                 on 1990 data (updated to definitive coeffs)
% coef1995.mat    Magnetic field coefficients and derivatives based
% on 1995 data (updated to definitive coeffs)
% coef2000.mat Magnetic field coefficients and derivatives based
% on 2000 data
%
% Global References:
% Parameters  Type  I/O  Description
% ----------  ----  ---  ----------------------------------------------
% run_time           i   (5,1) = Date/time indicator
%                               0 - Dates are in calendar format
%                               1 - Dates are in seconds since 9/1/57
% TCMAG              i   Epoch time for magnetic coefficients
%                        file(secs. from 9/1/57)
%                         900101.0 for time < 950101.0
%                         950101.0 for time < 20000101.0
%                         20000101.0 for all other times
% magcoef_name       i   Magnetic coefficients file name

% Development History:
% Name            Date      Description of Change
% --------------  --------  -------------------------------------------
% M. Nicholson    01/01/95  Created from the magfld FORTRAN code
% J. Deutschmann  07/01/95  Added in terms for model derivatives
% R. Harman       12/12/97  Added ability to take 1990 and 1995
%                             coefficients and the ability to take in
%                             multiple times and position vectors.
% R. Harman       01/09/98  Corrected epoch time problem.
% R. Luquette     03/31/98  Modified code to eliminate loops.
%                             Changed bounds of iloop
% A. Calder       07/13/98  Modified code to eliminate loops.
% D. Mucci        03/28/00  Added capability to process coef2000.mat and
%                             added the standard prolog
% M. Marchowsky   04/10/00  Made TCMAG a global and added
% magcoef_name

function [output_magfld_vectors, magfld_magnitude] = ...
    magfld( times, mln, pos, leap_seconds);

run_time(5,1) = 1;
if nargin >= 3 & ~isempty(times) & mln > 0 & mln < 11 & size(times,1) == size(pos,2)
    flag = 0;
    if run_time(5,1) == 0
        times = tcon40(times) - leap_seconds*ones(size(times));
    else
        times = times - leap_seconds * ones(size(times));
    end;

% -INITIALIZE PARAMETERS AT ENTRY

TG     = zeros(11,11);
TH     = zeros(11,11);
TGDOT  = zeros(11,11);
THDOT  = zeros(11,11);
DEGRAD = 57.2957795;
MAXN   = 10;
if times(1,1) < 1.178150400000000e+009  %19950101 in seconds since 9/1/57
load coef1990;
  TCMAG = 1.020384000000000e+009;  %Epoch 19900101.0
elseif times(1,1) < 1.335916800000000e+009  %20000101 in seconds since 9/1/57
load coef1995;
  TCMAG = 1.178150400000000e+009;  %Epoch 19950101.0
else
  load coef2000;
  TCMAG = 1.335916800000000e+009;  %Epoch 20000101.0
end

num_points = size(times,1);
output_magfld_vectors = zeros(3,num_points);
magfld_magnitude = zeros(1,num_points);

KNM = zeros(11,11);
SP = zeros(11);
CP = zeros(11);
G = zeros(11,11);
H = zeros(11,11);
SCH = zeros(11,11);
PNM = zeros(11,11);
DPNM = zeros(11,11);
PNM = zeros(11,11);
DPNM = zeros(11,11);

% -COMPUTE THE SCHMIDT COEFFICIENTS, EQ H-7 P781
SCH(1,1)=1.D0;
for N = 1:MAXN
  NN = N+1;
  SCH(NN,1) = SCH(NN-1,1)*(2.D0-1.D0/(N));
  XJ = 2;
  for M=1:N
    MM = M+1;
    SCH(NN,MM) = SCH(NN,MM-1)*sqrt((N-M+1.D0)*XJ/(N+M));
    XJ = 1.D0;
  end;
end;

% -NORMALIZE THE FIELD COEFFICIENTS AND DERIVATIVES, EQ H-6 P780
for N=0:MAXN
  for M=0:N
    NN = N+1;
    MM = M+1;
    TG(NN,MM) = TG(NN,MM)*SCH(NN,MM);
    TH(NN,MM) = TH(NN,MM)*SCH(NN,MM);
    TGDOT(NN,MM) = TGDOT(NN,MM)*SCH(NN,MM);
    THDOT(NN,MM) = THDOT(NN,MM)*SCH(NN,MM);
  end;
end;

% -COMPUTE THE KNM CONSTANST, EQ H-9 P781
%vvvvvvvvvvvvvvvvvvvvvvvvvvvvv GIL S.
KNM(2,1) = 0;
KNM(2,2) = 0;
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
for \( N = 2: \text{MAXN} \)
for \( M = 0:N \)
    \( NN = N+1; \)
    \( MM = M+1; \)
    \[
    \text{KNM}(NN,MM) = \frac{((N-1).0^2-M^2)}{(2.0*N-1.0)/(2.0*N-3.0)};
    \]
end;
end;

for \( \text{iloop} = 1: \text{num_points} \)

\% -COMPUTE COEFFICIENTS AT \text{REQTIM}
\text{DT} = (\text{times(iloop,1)}-\text{TCMAG})/3.15576e+07;
\text{G} = \text{TG+TGDOT.*DT};
\text{H} = \text{TH+THDOT.*DT};
\text{G} = \text{G(1:mln+1,1:mln+1)};
\text{H} = \text{H(1:mln+1,1:mln+1)};

\% the following is used when the position is input in
\% spherical/magnetic coordinates (rb)
%    \text{RMAG=}pos(1);
%    \text{COST=}cos(pos(2));
%    \text{SINT=}sin(pos(2));
%    \text{COSP=}cos(pos(3));
%    \text{SINP=}sin(pos(3));
\text{RMAG} = \text{norm(pos(:,iloop))};
\text{if abs(pos(3,iloop)/RMAG) == 1
    if pos(3,iloop) > 0
        \text{alp} = \pi/2;
    \text{end;}
    \text{if pos(3,iloop) < 0}
        \text{alp} = -\pi/2;
    \text{end;}
else
    \text{alp} = \text{atan2(pos(2,iloop),pos(1,iloop))};
\text{end;}
\text{COSA} = \text{cos(alp)};
\text{SINA} = \text{sin(alp)};
\text{DLONG} = \text{alp-ragren(times(iloop,1))};
\text{COST} = \text{pos(3,iloop)/RMAG};
\text{SINT} = \sqrt(1-\text{COST}^2);
\text{INT} = \text{ones(mln+1,1)}*\text{linspace}(0,\text{mln},\text{mln}+1);
\text{CP} = \text{cos(DLONGL.*INT)};
\text{SP} = \text{sin(DLONGL.*INT)};

\% -COMPUTE THE PNM, EQ H-8; AND THE DPNM, EQ H-10
\text{PNM} = \text{diag(SINT.^diag(INT))};
\text{PNM(2,1) = COST;}
\text{DPNM} = \text{zeros(mln+1,mln+1)};
\text{DPNM(2,2) = COST*PNM(1,1)};
\text{DPNM(2,1) = -SINT*PNM(1,1)};

for \( N = 3: \text{mln+1} \)
    \text{PNM(N,1:N-1) = COST*PNM(N-1,1:N-1)-KNM(N,1:N-1)*PNM(N-2,1:N-1);}
DPNM(N,N) = SINT*DPNM(N-1,N-1)+COST*PNM(N-1,N-1);
DPNM(N,1:N-1) = COST*DPNM(N-1,1:N-1) - SINT*PNM(N-1,1:N-1)-KNM(N, 1:N-1).*DPNM(N-2,1:N-1);
end;

% -COMPUTE FIELD UP TO ORDER MLN

AOR = 6371.20/RMAG;
brt_mat = G.*CP+H.*SP;
bp_mat = -G.*SP+H.*CP;

br_mat = brt_mat.*PNM;
b_t_mat = brt_mat.*DPNM;
bp_mat = bp_mat.*PNM;

aor_mat = (AOR.^(INT'+2));
br_mat = br_mat.*aor_mat.*(INT'+1);
b_t_mat = bt_mat.*aor_mat;
bp_mat = (bp_mat.*aor_mat).*INT;

BR = sum(sum(tril(br_mat)));
BT = -sum(sum(tril(bt_mat)));
BP = -sum(sum(tril(bp_mat)))/SINT;

BRB = ([BR BT BP]./100);

output_magfld_vectors(1,iloop) = ((BR*SINT+BT*COST)*COSA-BP*SINA)/100.0;
output_magfld_vectors(2,iloop) = ((BR*SINT+BT*COST)*SINA+BP*COSA)/100.0;
output_magfld_vectors(3,iloop) = (BR*COST-BT*SINT)/100.0;

% following is to extract b in inertial when the position is
% input in spherical/magnetic coordinate (rb)
% thets=pos(3)+ragren(times);
% thetb=pos(2);
% sinb=sin(thetb);
% cosb=cos(thetb);
% sins=sin(thets);
% coss=cos(thets);
% output_magfld_vectors(1)=(BR*sinb+BT*cosb)*coss-BP*sins;
% output_magfld_vectors(2)=(BR*sinb+BT*cosb)*sins+BP*coss;
% output_magfld_vectors(3)=BR*cosb-BT*sinb;
% output_magfld_vectors=output_magfld_vectors./100;

magfld_magnitude(1,iloop) = sqrt(BRB(1)*BRB(1)+BRB(2)*BRB(2)+BRB(3)*BRB(3));
end; % for iloop=1:num_points

else

if nargin < 3
flag = 1;
fprintf('NOT ENOUGH INPUT PARAMETERS FOR MAGFLD
');
elseif isempty(times)
flag = 2;
fprintf('NO INPUT TIMES FOR MAGFLD
');
elseif mln<0 | mln>10

end;
flag = 3;
fprintf('MAG FIELD MODEL ORDER IS INVALID FOR MAGFLD\n');
elseif size(times,1) ~= size(pos,2)
  flag = 4;
  fprintf('TIMES AND POSITION SIZES DO NOT MATCH UP IN MAGFLD\n');
end;
end; %if nargin>=3&~isempty(times)&mln>0&mln<11&size(times,1)==size(pos,2)

RAGREN.m

% function alpha = ragren(t)
% % RAGREN - Computes Greenwich hour angle (GHA) at a specified time
% % function alpha = ragren(t)
% % Input: t = 1-by-n array of civil seconds since 570901
% % Output: alpha = 1-by-n array of GHA values (radians)
% % External References: none
% % Special Note: Do not include leap seconds when computing the
% %     input t
%     % Adapted from ATTIT.ATTMAIN.UTIL77.PAN(RAGREN) M.Shear (CSC)
%     (alpha = 0 @ 720101.171808107)
%     % Constants: tref = transit time of 1st point of Aries
%     January 1, 1972, 17 hours, 18 minutes,
%     8.107 seconds, UT)
%     omega = Earth spin rate (deg/s)
%     % Original Version: 14 February 1995 - Don Chu
%     %% constants derived from American Ephemeris and Nautical Alminac
%     rtd = 180/pi;
%     tref = 452366288.107;
%     omega = 0.004178074622;
%     % accuracy better than 0.01 degrees for all times from 1900-2100
%     %% Greenwich Hour Angle
%     rot = (t-tref)*omega;
%     alfd = rem(rot,360);
%     alpha = alfd/rtd;
Orbit.m

%--------------------------------------------------------------
%Program Orbit
%--------------------------------------------------------------
%This program will find the following the location of MR SAT along the orbit
%This program was written by students of the University of Missouri–Rolla

function [theta_dot,R,V,theta]=orbit(t),a,e,i,w,RAAN,P)

%--------------------------------------------------------------
%***Define the Needed Constants
%--------------------------------------------------------------

pi=4*atan(1);  %3.141592.....
mu_earth=3.98699E5;  %Gravitational Constant of Earth
musun=1.327E11;   %Gravitational Constant of the Sun
R_D=180/pi;   %Radians to Degrees
D_R=pi/180;  %Degrees to Radians

%--------------------------------------------------------------
%***Define Constant Orbital Parameters
%--------------------------------------------------------------

a=6778;  %Semi-Major Axis (km)
i=51.6*D_R;  %Angle of Inclination (radians)
e=0.0;  %Eccentricity
w=0.0*D_R;  %Argument of Periapsis
RAAN=246*D_R;  %Right Ascension of the Ascending Node

P=2*pi*(a^3/mu_earth)^.5;  %period (sec)
pmin=P/60;  %period (min)
rp=a*(1-e);  %Perigee (km)
ra=a*(1+e);  %Apogee (km)
para=a*(1-e^2);  %Parameter (Semi-Latus Rectum)

n=(mu_earth/a^3)^.5;  %Mean Motion (rad/sec)
h=para*mu_earth^3)^.5;  %Specific Angular Momentum
specE=-mu_earth/(2*a);  %Specific Energy

%--------------------------------------------------------------
%***Determine the Location of the S/C wrt Inertial Frame
%--------------------------------------------------------------

M=n*t;  %Mean Anomaly (radians)

%Use Newton's Method to Find the Eccentric Anomaly

%First Define the Initial Eccentric Anomaly as the Mean Anomaly
E0=M;

%Newton's Method
tol=1;  %Tolerance
while tol>1E-10
E1=E0-((M-(E0-e*sin(E0)))/(e*cos(E0)-1));
E0=E1;

%--------------------------------------------------------------
tol=(M-(E0-e*sin(E0)));
end

%Calculate True Anomaly (in radians) using Eccentric Anomaly
nu=(atan(((1+e)/(1-e))^.5*tan(E1/2)))*2;
theta=nu+w;                     %radians

%Calculate the Radius
r=para/(1+e*cos(nu));          %Radius (km)
theta_dot=h/r^2;                %radians/second

%Use Direction Cosine Matrix (DCM) to Find x, y, and z
%components of the Spacecraft's Location using the radius
B_DCM=[cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0 1];
C_DCM=[1 0 0; 0 cos(i) -sin(i); 0 sin(i) cos(i)];
D_DCM=[cos(RAAN) -sin(RAAN) 0; sin(RAAN) cos(RAAN) 0; 0 0 1];
r_comp=[r; 0; 0];
A_DCM=D_DCM*C_DCM*B_DCM;
R=A_DCM*r_comp;

%============================================================================================================================================
%***Determine the Velocity of the S/C wrt Inertial Frame
%============================================================================================================================================

%First find the velocity components

%Velocity in the Radial Direction (km/s)
v_r=(mu_earth*e*sin(nu))/h;

%Velocity in the Tangential Direction (km/s)
v_theta=h/r;

%Put these components into the DCM
v_comp=[v_r; v_theta; 0];
V=A_DCM*v_comp;
APPENDIX B

MR SAT AND MRS SAT DIMENSIONS
B. MR SAT AND MRS SAT DIMENSIONS

Table B.1. MR SAT and MRS SAT Estimated Properties

<table>
<thead>
<tr>
<th>Spacecraft</th>
<th>Mass (kg)</th>
<th>$I_{XX}$ (kg-m$^2$)</th>
<th>$I_{YY}$ (kg-m$^2$)</th>
<th>$I_{ZZ}$ (kg-m$^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR SAT</td>
<td>28.25</td>
<td>1.57</td>
<td>1.53</td>
<td>0.79</td>
</tr>
<tr>
<td>MRS SAT</td>
<td>11.45</td>
<td>0.55</td>
<td>0.62</td>
<td>0.34</td>
</tr>
<tr>
<td>Coupled</td>
<td>39.70</td>
<td>3.75</td>
<td>3.68</td>
<td>1.11</td>
</tr>
</tbody>
</table>

These dimensions are from the UMR document Satellite Dimensions (01-011 Rev. B).

MR SAT dimensions

Side solar panels:
- (4) 19.7 x 44 x 0.5 cm
- (1) 19.5 x 44 x 0.5 cm
- (1) 16.65 x 44 x 0.5 cm

Honeycomb Al side panels: 21.5 x 45.5238 x 1.143 cm
Al isogrid side panels: 19.45 x 29.73 x 0.47625 cm
Al isogrid shelf: hexagon w/ 19.52375 cm sides 0.47625 cm thick
Al isogrid bottom panel: circle 41.275 cm diameter x 0.635 cm thick
Al isogrid top panel: hexagon w/ 20 cm sides 0.635 cm thick
Spacers: 0.65 cm outer diameter, 0.5 cm inner diameter, 1.55 cm long

Total dimensions of MR SAT:
- Isogrid: 31 cm tall
- Total: 46 cm tall

Note:
The side panels are placed on top of the bottom circular panel and the top panel is placed on top of the edges of the side panels.

Revised June 7, 2006:
- Bottom and top isogrid panels must be 0.635 cm instead of 0.47625 cm for stiffness and helicoil inserts
- Isogrid side panels must be 19.45 cm wide instead of 20 cm b/c they do not have angled edges anymore
- Isogrid side panels must be 29.73 cm tall instead of 30.0475 cm b/c bottom and top are thicker
MRS SAT dimensions

Side solar panels: 14 x 14.05 x 0.5 cm
Top solar panel: 20 x 13.4 x 0.5 cm
Honeycomb Al side panels: 15 x 15.4 x 0.889 cm
Honeycomb Al top panel: 22 x 15.4 x 0.889 cm
Al isogrid side panels: 16.7 x 15.5756 x 0.47625 cm
Al isogrid top/bottom panels: hexagon w/ 17.5 cm sides, 0.635 cm thick
Spacers: 0.65 cm outer diameter, 0.5 cm inner diameter, 1.27 cm long

Total dimensions of MRS SAT:
  Isogrid: 13.0865 cm tall
  Honeycomb: 15 cm tall

Note:
  The side panels are placed on top of the bottom hexagonal panel, and the top panel is placed on top of the edges of the side panels.

Total height of satellites with top solar panel: 51.2746 cm

Revised June 7, 2006:
- Bottom and top isogrid panels must be 0.635 cm instead of 0.47625 cm for stiffness and helicoil inserts
- Isogrid side panels must be 16.9 cm wide instead of 17.5 cm b/c they do not have angled edges anymore
- Isogrid side panels must be 11.817 cm tall instead of 12.1345 cm b/c bottom and top are thicker, and the total is the same

Revised October 26, 2006:
  MRS SAT side panels are now 15.5756 cm tall instead of 11.817 cm, added two rows of isogrid triangles to panels
APPENDIX C

LYAPUNOV STABILITY
C. LYAPUNOV STABILITY

The Lyapunov Stability principle was applied to demonstrate controller stability. To carry out a Lyapunov stability assessment, a positive definite Lyapunov function must have a negative definite derivative. If the derivative is found to be negative semi-definite, then the zero points must be equilibrium points to prove system stability. The ADAC system is defined using quaternion notation as

\[
q = \langle q_0, v_q \rangle, \text{ where } v_q = q_1 \hat{i} + q_2 \hat{j} + q_3 \hat{k} \tag{C1}
\]

\[
\omega = \langle \omega_b, v_\omega \rangle, \text{ where } v_\omega = \omega^b_1 \hat{i} + \omega^b_2 \hat{j} + \omega^b_3 \hat{k} \tag{C2}
\]

\[
\dot{q} = -\frac{1}{2} \omega q \tag{C3}
\]

\[
I\ddot{\omega} = N - v_\omega \times I v_\omega \tag{C4}
\]

where \(q\) is the attitude quaternion, \(\omega\) is the angular velocity quaternion, \(I\) is the satellite inertia tensor, and \(N\) is the applied control moment. To determine the required \(N\), a guidance law has been developed using an inner/outer loop design. The inner loop determines the desired angular velocity, \(\omega_d\), so that the current attitude quaternion, \(q\), will track the desired attitude quaternion, \(q_d\), using an error function, \(e_q = q - q_d\). This error function must quickly go to zero for the attitude correction to be valid. A positive definite Lyapunov function is selected as half of the square of the error quaternion magnitude.

\[
V = \frac{1}{2} \text{Re} \left( e^c e_q \right) = \frac{1}{2} |e_q|^2 \tag{C5}
\]

where \(e^c\) is the complex conjugate of \(e\), and \(\text{Re}: \mathbb{H} \to \mathbb{R}\) extracts the real part of the quaternion \(q \in \mathbb{H}\), with \(\mathbb{H}\) and \(\mathbb{R}\) representing the set of all quaternions and all real numbers, respectively. Knowing that

\[
e_q = q - q_d = \langle e_0, v_{e_q} \rangle \tag{C6a}\]
\[
\dot{\epsilon}_q = \dot{q} - \dot{q}_d = \left\langle \dot{\epsilon}_0, \dot{v}_q \right\rangle
\]

(C6b)

using Equation 3.28, it can be shown that

\[
\text{Re}\left(\epsilon_q^c \epsilon_q^c\right) = \epsilon_0^2 + v_q \cdot v_q
\]

(C7)

Furthermore,

\[
V = \frac{1}{2} (\epsilon_0^2 + v_q \cdot v_q)
\]

(C8)

Taking the derivative of Equation C8 gives,

\[
\dot{V} = (\epsilon_0 \dot{\epsilon}_0 + v_q \cdot \dot{v}_q) = \text{Re}(\epsilon_q^c \dot{\epsilon}_q)
\]

(C9)

which leads to

\[
\dot{V} = \text{Re}\left(\epsilon_q^c (\dot{q} - \dot{q}_d)\right)
\]

(C10)

The derivative of the desired state, \( \dot{q}_d \), is assumed to be zero, which leads to

\[
\dot{V} = \text{Re}(\epsilon_q^c \dot{q}) = \text{Re}\left(\epsilon_q^c \left( \frac{1}{2} \omega q \right) \right)
\]

(C11)

\[
= -\frac{1}{2} \text{Re}(\epsilon_q^c \omega q)
\]

Because \( \omega \in H_0 \), where \( H_0 \) is the set of all quaternions for which the real part is zero,
\( \omega_0 = 0 \) \hfill (C12)

The derivative of the Lyapunov function can be shown to be

\[
\dot{V} = -\frac{1}{2} \Re \left(e_0^T \left\langle -v_{\omega} \cdot v_{\omega} + q_0 v_{\omega} + v_{\omega} \times v_q \right\rangle \right)
\]

\[
= -\frac{1}{2} \Re \left( e_0^T \left\langle -v_{\omega} \cdot v_{\omega} + q_0 v_{\omega} + v_{\omega} \times v_q \right\rangle \right)
\]

\[
= -\frac{1}{2} \left[v_{\omega} \cdot (q_0 v_{\omega} + v_{\omega} \times v_q) - e_0^T (v_{\omega} \cdot v_q)\right]
\]

The derivative of the Lyapunov function, Equation C13, can be made negative definite by choosing \( \nu = \omega_0 \). The following vector matrix identities

\[
A \cdot (B + C) = A \cdot B + A \cdot C \hfill (C14a)
\]

\[
A \cdot (B \times A) = A \cdot (A \times B) = 0 \hfill (C14b)
\]

\[
(A \times B) \times C = (C \cdot A) B - (B \cdot C) A \hfill (C14c)
\]

are used in finding \( \omega_0 \). Suppose \( \omega_q = (v_{\omega} \times v_q) \). Then it can be shown that the equation for \( \dot{V} \) is
\[ \dot{V} = -\frac{1}{2} \left[ v_{q_e} \cdot \left( q_0 (v_{eq} \times v_q) + (v_{eq} \times v_q) \times v_q \right) - e_0 \left( (v_{eq} \times v_q) \cdot v_q \right) \right] \\
\quad = -\frac{1}{2} \left[ v_{q_e} \cdot \left( q_0 (v_{eq} \times v_q) + (v_{eq} \times v_q) \times v_q \right) - 0 \right] \\
\quad = -\frac{1}{2} \left[ q_0 v_{q_e} \cdot (v_{eq} \times v_q) + v_{q_e} \cdot \left( (v_{eq} \times v_q) \times v_q \right) \right] \\
\quad = -\frac{1}{2} \left[ 0 + v_{q_e} \cdot \left( (v_{eq} \times v_q) \times v_q \right) \right] \\
\quad = -\frac{1}{2} \left[ v_{q_e} \cdot \left( (v_{eq} \times v_q) \times v_q \right) \right] \\
\quad = -\frac{1}{2} \left[ v_{q_e} \cdot \left( (v_q \cdot v_{q_e}) v_q - (v_q \cdot v_q) v_{q_e} \right) \right] \\
\quad = -\frac{1}{2} \left[ v_{q_e} \cdot \left( (v_q \cdot v_{q_e}) v_q - v_{q_e} \cdot (v_q \cdot v_q) v_{q_e} \right) \right] \\
\quad = -\frac{1}{2} \left[ (v_q \cdot v_{q_e})^2 - v_q^2 \left| v_{q_e} \right|^2 \right] \\
\quad = -\frac{1}{2} \left[ v_q^2 \left| v_{q_e} \right|^2 \cos^2 \left( \theta_{v_q v_{q_e}} \right) - v_q^2 \left| v_{q_e} \right|^2 \right] \\
\quad = -\frac{1}{2} \left| v_q \right|^2 \left| v_{q_e} \right|^2 \left( \cos^2 \left( \theta_{v_q v_{q_e}} \right) - 1 \right) \]  

(C15)

The square of the magnitude of the vector parts of any quaternion are greater than zero. Therefore, the other elements in Equation C15 must be negative definite to prove that the system is stable. For a wide range of $\theta$, Equation C15 can be shown to be positive, which would not conclusively prove the controller’s stability, as illustrated by Figure C.1.
A new choice of $\omega_d = (v_q \times v_{\epsilon_q})$ shows that the equation for $\dot{V}$ becomes

\[
\dot{V} = -\frac{1}{2} \left[ v_{\epsilon_q} \cdot (q_0 (v_q \times v_{\epsilon_q}) + (v_q \times v_{\epsilon_q}) \times v_q) - e_0 \left( (v_q \times v_{\epsilon_q}) \cdot v_q \right) \right]
\]

\[
= -\frac{1}{2} \left[ v_{\epsilon_q} \cdot (q_0 (v_q \times v_{\epsilon_q}) + (v_q \times v_{\epsilon_q}) \times v_q) - 0 \right]
\]

\[
= -\frac{1}{2} \left[ q_0 v_{\epsilon_q} \cdot (v_q \times v_{\epsilon_q}) + v_{\epsilon_q} \cdot ((v_q \times v_{\epsilon_q}) \times v_q) \right]
\]

\[
= -\frac{1}{2} \left[ 0 + v_{\epsilon_q} \cdot ((v_q \times v_{\epsilon_q}) \times v_q) \right]
\]

\[
= -\frac{1}{2} \left[ v_{\epsilon_q} \cdot ((v_q \cdot v_q)v_{\epsilon_q} - (v_q \cdot v_{\epsilon_q})v_q) \right]
\]

\[
= -\frac{1}{2} \left[ v_{\epsilon_q} \cdot (v_q \cdot v_q)v_{\epsilon_q} - v_{\epsilon_q} \cdot (v_q \cdot v_{\epsilon_q})v_q \right]
\]

\[
= -\frac{1}{2} \left[ (v_q \cdot v_q)(v_{\epsilon_q} \cdot v_{\epsilon_q}) - (v_q \cdot v_{\epsilon_q})(v_{\epsilon_q} \cdot v_q) \right]
\]

\[
= -\frac{1}{2} \left[ v_q^2 \left| v_{\epsilon_q} \right|^2 - \left| v_q \right|^2 \left| v_{\epsilon_q} \right|^2 \cos^2 \left( \theta_{v_{\epsilon_q}v_q} \right) \right]
\]

\[
= -\frac{1}{2} \left| v_q \right|^2 \left| v_{\epsilon_q} \right|^2 \left[ 1 - \cos^2 \left( \theta_{v_{\epsilon_q}v_q} \right) \right]
\]

(C16)
The square of the magnitude of the vector parts of any quaternion are greater than zero. The elements in Equation C16, other than the square of the magnitude of the vector parts, can now be shown to be negative semi-definite, which in turn proves that \( \dot{V} \) is negative semi-definite. Figure C.2 illustrates that the chosen \( \omega_d \) gives a negative semi-definite \( V \), but not enough information is given to prove that the zero points are equilibrium points.

Additional control is needed to provide the stability; therefore, additional terms were added to the desired angular velocity.

\[
\omega_d = q_0 v_{e_q} - e_0 v_q - \left( v_{e_q} \times v_q \right)
\]  

(C17)

Using this desired angular velocity, the derivative of the Lyapunov function can now be shown to prove complete system stability.
\[
\dot{V} = -\frac{1}{2}\left[ v_{r_0} \cdot \left( q_0 \left( q_0v_{r_0} - e_0v_q - (v_{r_0} \times v_q)\right) + \left( q_0v_{r_0} - e_0v_q - (v_{r_0} \times v_q)\right) \times v_q \right) \right]
\]

\[
= -\frac{1}{2}\left[ -e_0 \left( q_0v_{r_t} - e_0v_q - (v_{r_t} \times v_q) \cdot v_q \right) \right]
\]

\[
= -\frac{1}{2}\left[ v_{r_0} \cdot \left( q_0q_0v_{r_0} - q_0e_0v_q - q_0 \left( v_{r_0} \times v_q \right) + q_0 \left( v_{r_0} \times v_q \right) - \left( v_{r_t} \times v_q \right) \right) \right]
\]

\[
= -\frac{1}{2}\left[ -e_0 \left( q_0v_{r_0} \cdot v_q - e_0 \left( v_{r_t} \cdot v_q \right) \right) \right]
\]

\[
= -\frac{1}{2}\left[ q_0^2 \left( v_{r_0} \cdot v_{r_0} \right) - 2q_0e_0 \left( v_{r_0} \cdot v_q \right) - \left( v_{r_0} \cdot \left( v_{r_0} \times v_q \right) \right) \right] + e_0^2 \left| v_q \right|^2
\]

\[
= -\frac{1}{2}\left[ q_0^2 \left| v_{r_0} \right|^2 - 2q_0e_0 \left| v_{r_0} \right| \left| v_q \right| \cos(\theta_{r_{r_0} q}) \right] + e_0^2 \left| v_q \right|^2 + e_0^2 \left| v_q \right|^2
\]

\[
= -\frac{1}{2}\left[ q_0^2 \left| v_{r_0} \right|^2 \right] - 2q_0e_0 \left| v_{r_0} \right| \left| v_q \right| \cos(\theta_{r_{r_0} q}) + e_0^2 \left| v_q \right|^2
\]

(C18)

The derivative still needs to be shown to be negative definite or semi-definite with the zero points being equilibrium points. The first portion of the equation has already been shown to give a negative semi-definite $V$. Considering only the last portion of the derivative equation, it can be shown that

\[
\left| q_0v_{r_0} - e_0v_q \right|^2 = \left| q_0v_q - (q_0 - q_{d_0})v_q \right|^2
\]

\[
= \left| q_0v_q - q_0v_{q_0} - q_0v_q + q_{d_0}v_q \right|^2
\]

\[
= \left| q_{d_0}v_q - q_0v_{q_0} \right|^2
\]

\[(C19)\]

Using Equation 3.25, it can be shown that
Therefore, \( q_0 v_{c_q} - e_0 v_q = 0 \) when \( q_{d_0} = \pm q_0 \), and \( v_q \) is collinear to \( v_{c_q} \). This occurs at the equilibrium points; therefore, the Lyapunov function still satisfies the necessary conditions to prove that the control is stable.

To ensure that the necessary \( \omega_l \) is obtained, a moment, \( N \), can be found such that \( v_\omega \) will be driven to \( \omega_l \). This is done by defining another error, \( e_\omega = v_\omega - \omega_l \). This error function must quickly go to zero for the inner/outer loop method to function effectively. When \( v_\omega = \omega_l \), the system is stable and \( q \to q_{d_0} \). Again, a selected definite Lyapunov function and its derivative are

\[
V = \frac{1}{2} e_\omega^T J e_\omega \quad (C21)
\]
\[
\dot{V} = \frac{1}{2} \dot{\epsilon}_\omega^T I \epsilon_\omega + \frac{1}{2} \dot{\epsilon}_\omega^T \dot{\epsilon}_\omega \\
= \frac{1}{2} \epsilon_\omega^T I \dot{\epsilon}_\omega + \frac{1}{2} \epsilon_\omega^T \dot{\epsilon}_\omega \\
= \epsilon_\omega^T \dot{\epsilon}_\omega \\
= \epsilon_\omega^T I (\dot{\epsilon}_\omega - \dot{\alpha}_\omega) \\
= \epsilon_\omega^T I (\dot{\epsilon}_\omega - 0) \\
= \epsilon_\omega^T I (\dot{\epsilon}_\omega) \\
= \epsilon_\omega^T (N - \nu_\omega \times I \nu_\omega) \\
\]

\( \text{(C22)} \)

From this, the necessary control moment, \( N \), which ensures that the derivative is negative definite and that \( \nu_\omega \) will approach \( \alpha_\omega \) is

\[
N = \alpha_\omega \times I \alpha_\omega - \epsilon_\omega \\
\text{(C23)}
\]
APPENDIX D

THROTTLE FILTER EXAMPLE
D. THROTTLE FILTER EXAMPLE

This section is intended to show an example of how the throttle filter operates. This example uses data from the thrust controller simulation (with initial conditions from Table 5.13). Again, the filter works by comparing a filter estimate of the current throttle setting, $\hat{\theta}$, with the desired throttle setting, $\theta$, calculated from the linear programming thruster solution. For this example, only thruster two is viewed. The time interval of 300 to 310 seconds is modeled here. If the estimated throttle setting is higher than the desired throttle setting, then the thruster should not fire during the next $\Delta t$ (0.1 second) time interval. However, if the estimated throttle setting is lower than the desired throttle setting, then the thruster should be fired during the next $\Delta t$. Thus, the thruster logic is given by

$$u = \begin{cases} 
0 \text{ (thruster off)} & \theta < \hat{\theta} \\
1 \text{ (thruster on)} & \theta > \hat{\theta} 
\end{cases} \quad (D.1)$$

Once the $u$ is determined for each thruster, the throttle estimates are updated using the following filter equation

$$\hat{\theta}_{k+1} = \hat{\theta}_k + K \left( u - \hat{\theta}_k \right) \quad (D.2)$$

where $K$ is a filter gain of 0.25 and $k$ indicates the time interval. Using the discrete on/off values for each thruster, the appropriate thrusters are turned on during the next time interval, $\Delta t$, after which Equation D.1 is used to determine which thrusters are activated during the next time interval. Table D.1 shows the values for each of the throttle filter variables.
<table>
<thead>
<tr>
<th>k</th>
<th>$\theta$</th>
<th>$\hat{\theta}$</th>
<th>$u$</th>
<th>$\hat{\theta}_{k+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>0.03765</td>
<td>0.0481005</td>
<td>0</td>
<td>0.036075375</td>
</tr>
<tr>
<td>301</td>
<td>0.03996</td>
<td>0.03607538</td>
<td>1</td>
<td>0.277056531</td>
</tr>
<tr>
<td>302</td>
<td>0.04228</td>
<td>0.27705653</td>
<td>0</td>
<td>0.207792398</td>
</tr>
<tr>
<td>303</td>
<td>0.03032</td>
<td>0.2077924</td>
<td>0</td>
<td>0.155844299</td>
</tr>
<tr>
<td>304</td>
<td>0.03258</td>
<td>0.1558443</td>
<td>0</td>
<td>0.116883224</td>
</tr>
<tr>
<td>305</td>
<td>0.03484</td>
<td>0.11688322</td>
<td>0</td>
<td>0.087662418</td>
</tr>
<tr>
<td>306</td>
<td>0.03712</td>
<td>0.08766242</td>
<td>0</td>
<td>0.065746814</td>
</tr>
<tr>
<td>307</td>
<td>0.03941</td>
<td>0.06574681</td>
<td>0</td>
<td>0.04931011</td>
</tr>
<tr>
<td>308</td>
<td>0.04171</td>
<td>0.04931011</td>
<td>0</td>
<td>0.036982583</td>
</tr>
<tr>
<td>309</td>
<td>0.04401</td>
<td>0.04931011</td>
<td>0</td>
<td>0.036982583</td>
</tr>
<tr>
<td>310</td>
<td>0.04631</td>
<td>0.03698258</td>
<td>1</td>
<td>0.277736937</td>
</tr>
</tbody>
</table>

Notice that the initial throttle estimate was 0.048, and the desired throttle was 0.03765. For this reason, over the time interval, the thruster fired twice, and the final throttle estimate was 0.0369, which was much closer to 0.03765.
BIBLIOGRAPHY


VITA

David Randall Walker was born on June 29, 1983, in Sikeston, Mo. to James and Gina Walker. On Feb. 10, 1987 his sister, Amy, was born in Clinton, Mo. He graduated from Jefferson City High School in Jefferson City, Mo. in May of 2001. From there he went to the University of Missouri–Rolla to pursue a bachelor’s degree in Aerospace Engineering. In May of 2005, he obtained his goal and went on to receive his master’s degree in Aerospace Engineering in May 2007. While attending a wedding in May of 2005, David was introduced to Blair Gamel, whom he married the following May. While attending UMR, David was a member of AIAA, vice president and president of Sigma Gamma Tau, Residential Assistant at the Christian Campus House and worship leader for Christian Campus Fellowship, an avid member of the UMR Ultimate Frisbee Club Team, and the MR SAT Program Manager.