3-2021

Hybridized Spacecraft Attitude Control via Reinforcement Learning using Control Moment Gyroscope Arrays

Cecily C. Agu

Hybridized Spacecraft Attitude Control via Reinforcement Learning using Control Moment Gyroscope Arrays

THESIS

Cecily C Agu, First Lieutenant, USAF

AFIT/ENY/MS/21-M-328

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/ENY/MS/21-M-328

Hybridized Spacecraft Attitude Control via Reinforcement Learning using Control Moment Gyroscope Arrays

THESIS

Presented to the Faculty

Department of

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Astronautical Engineering

Cecily C Agu, B.S.A.E.

First Lieutenant, USAF

March 25, 2021

AFIT/ENY/MS/21-M-328

Hybridized Spacecraft Attitude Control via Reinforcement Learning using Control Moment Gyroscope Arrays

THESIS

Cecily C Agu, B.S.A.E.
First Lieutenant, USAF

Committee Membership:

Major Joshuah A. Hess, Ph.D
Chair

Major Costantinos Zagaris, Ph.D
Member

Dr. Richard G. Cobb
Member

Abstract

Next-generation spacecraft have an increased requirement for precise pointing accuracy and rotational agility to accomplish a variety of missions. Current attitude determination and control systems (ADCS) employ multiple types of actuators such as reaction wheels, control moment gyroscopes (CMG), and thrusters to maneuver spacecraft for minimal time slews between orientations. There is practical importance in designing spacecraft to achieve rapid retargeting and CMGs are an effective option. Traditional methods to solve the general attitude control problem and conduct control allocation utilize open-loop optimal control algorithms such as dynamic programming to optimize overall system performance, combined with state feedback-based control to track the optimal solution. A limitation of these current control methods is to process realistic system constraints such as rate constraints and provide solutions that contain guaranteed stability and accuracy. Machine learning techniques in the form of reinforcement learning (RL) can solve these complex nonlinear problems by serving as a potential controller option for the ADCS system. The current study examines the use of an RL agent in an ADCS for performance enhancement of a spacecraft conducting multiple slewing maneuvers to gather ground target information. Three CMG arrays were implemented in two simulated spacecraft environments using an RL controller. The performance of the controllers was evaluated using target profiles from traditional control law implementations, singularity measure, and two sets of initial state values. The current research demonstrates that while RL techniques can be implemented, further exploration is needed to investigate the operational efficacy of an approach such as hybridized control for producing comparable performance attributes with respect to traditional control laws.

Table of Contents

Page

List of Figures

List of Tables

Hybridized Spacecraft Attitude Control via Reinforcement Learning using Control Moment Gyroscope Arrays

I. Introduction

1.1 Motivation

The rapid progression and expansion of technology in modern society is attributable to increased automation, programming, and computational capacity. In this modern industrial society, the wide adoption of computers and microprocessors to largely automate work and increase specialization of the overall technological infrastructure has improved common activities such as financial transactions, instant communications, and teleworking [12]. One such influential technological invention is the personal computer, which in the past few decades has seen a staggering increase in computational capacity, thanks in part to the modern semiconductor. The concept and projection of these electronic devices is attributed to Gordon Moore, who in 1965, gave a prediction on the future of the semiconductor components industry over the next ten years [13]. What came to be defined as Moore's law, or the general trend that the number of transistors in a dense integrated circuit doubles about every two years, has driven hardware growth, increased machine complexity, and reliance on autonomous computation. Requirements in industrial, commercial, and military sectors alike rely on rapid, autonomous decision-making for projected outputs for large-scale operations and customer bases. Examples include off-shore oil production [14], distributed cooperative planning within large commercial satellite constellations such as Starlink [15], and autonomous military vehicles.

The symbiosis between computers and semiconductors has made possible factory-scale model assembly, new kinds of firm-customer relationships in a sector such as online commerce, and unmanned aerial systems for improved surveillance, reconnaissance, and humanitarian support. This steadily increasing need for computers to not only perform these tasks but learn and improve autonomously has also expanded the field of computer science. The limitations of the natural intelligence displayed by human beings and animals has prompted the need for duplication and expansion of this intelligence for machines. The concise definition of what is considered artificial intelligence (AI) is the effort to automate intellectual tasks normally performed by humans [16]. The early AI of the mid-20th century was symbolic in nature, requiring programmers to craft large sets of explicit rules for manipulating knowledge. Each symbolic manipulation represents theories or laws to be implemented within a system. As technology progressed, so did the requirements for data manipulation by machines and a new approach called machine learning came to replace symbolic AI and proved to be more useful by statistically learning patterns or regularities from data [17]. Machine learning as a subset of artificial intelligence has become an important aspect of the rapidly progressing technology, data storage, and computational task parallelization of modern society.

The use of algorithms and data representation assimilation to improve themselves through experience automatically has caused the development of several applications of machine learning for tasks such as email filtering, computer vision, and game-playing [18]. The syntactic analysis within machine learning and the informed decision based on what is learned from provided data fulfills and automates tasks too complex and time consuming for standard human time frames or time-dependent astrodynamic processes such as orbits. Two types of machine learning called supervised (SL) [16] and reinforcement learning (RL) [7] are adaptive tools used for

solving problems in real-time scenarios. There exists significant application potential for these types of machine learning to similarly improve a wide variety of space systems with complex dynamic models and approximations. Examples include attitude determination and control systems on board spacecraft, ground sensors, and real-time rendezvous and proximity operations. Any spacecraft system can incorporate artificial intelligence, but notable examples of current interest include satellite pose estimation using a convolutional neural network [19] and spacecraft on-board autonomy [20]. Another notable example of AI incorporation is the work of George and Hess on the application of artificial neural networks (ANN) to conduct translational spacecraft control in proximity operations, evaluating neurcontroller performance as a viable solution for complex controls problems [21].

When applying reinforcement learning to spacecraft systems, this study is focused on the attitude control and determination system (ADCS), specifically the control allocation therein. The general spacecraft attitude control problem involves three-axis orientation using momentum exchange actuator devices such as reaction wheels and control momentum gyroscopes [3]. Control allocation is defined as the combination of which and how much of an actuator to use in order to avoid system constraints. The controller of the system is linked to the actuators and serves as the control allocation authority. Certain space missions may require the agile satellite capabilities of minimal time slewing to maximizing the amount of information to be collected, while other missions may require precise pointing [3]. The attitude control system actuators commonly utilized to accomplish this slew and collect process are reaction wheel assemblies and control moment gyroscopes. Precise pointing is therefore required when conducting the collect mode for specific mission targeting and a robust attitude control system makes this attainable. Traditional methods to solve the general attitude control problem and conduct control allocation utilize open-loop optimal control

algorithms such as dynamic programming to optimize overall system performance followed by state feedback-based control to perform the attitude control maneuvers [22]. These current control methods process realistic system constraints (e.g. input saturation and rate constraints) and provide solutions that contain guaranteed stability and accuracy. Additional control allocation constraints can be realized as secondary objectives that will be explored later on in this study [14]. Reinforcement learning has the ability to solve this complex problem and in the context of on-orbit operations, is a potential option for this environment. This study seeks to examine the benefits of using machine learning agents to solve control allocation problems in attitude determination system of a spacecraft and the following section provides detail on the individual aspects of the study.

1.2 Problem Statement

The purpose of this research is to develop an algorithmic environment for a spacecraft attitude control system, specifically utilizing reinforcement learning as a control allocation authority. Such an environment requires the spacecraft state information for real-time control for on-orbit operations. Changes to the structure of the spacecraft throughout these operations will produce subsequent changes in the moments of inertia. These changes necessitate a reactive attitude control system to accurately estimate rotational dynamics which can be accomplished using a reinforcement learning algorithm. Traditional control allocation schemes within ADCS are established control methods that achieve desired performance outcomes but can still encounter allocation challenges based on the type of actuator system. RL has recently shown promise in solving difficult numerical problems and has been discovered to generate non-intuitive solutions to existing problems, making it an ideal candidate for handling the nonlinear dynamics surrounding the use of control moment gyroscopes. There is

a need to examine the computational efficiency, attitude characterization, and control strategy of an RL controller and the potential benefits it brings a spacecraft ADCS. The use of RL in a spacecraft ADCS can provide additional control allocation authority that may outperform conventional control methods. The process for this RL model to communicate with a simulated spacecraft is described in greater detail in Section 3.3 of this thesis.

1.3 Objectives, Tasks, and Scope

This research is intended to provide a simulated basis for spacecraft control allocation, defined as the controller output to the actuator array, performed with machine learning for eventual experimental implementation. The objectives, corresponding tasks, assumptions, and research scope are outlined in this section.

1.3.1 Objectives

Through modeling hybridized estimation techniques, this research seeks to accomplish the following objectives:

(a) Can machine learning techniques be applied to traditional methods of spacecraft attitude control and control allocation?

(b) Are there certain attitude parameterizations that will produce better ML techniques for spacecraft attitude control? What performance enhancement and error reduction can machine learning techniques provide for an attitude control system?

(c) Do results produced by the machine learning methods resemble results produced by traditional methods for attitude control?

1.3.2 Tasks

The following tasks require accomplishment in order to properly address the proposed research objectives:

1. Develop a reinforcement learning model for integration with a spacecraft control system;

2. Implement an algorithm for the reinforcement learning environment that when integrated with an attitude control system, function as controllers in a feedback-loop with the simulation;

3. Expand the complexity of the model by adding constraints and noise to simulate spacecraft hardware limitations and vary spacecraft physical properties and actuator types to evaluate overall system performance, error, and singularity avoidance;

4. Validate the attitude control performance values generated by the machine learning models by comparing to traditional attitude control techniques in a baseline model.

1.3.3 Assumptions and Scope

The orbital regime of this work is with Earth-centered orbits and projected for use with on-board realization and implementation. These orbits are circular and no other classical orbital elements and perturbations are defined. The simulation is a modified scenario from the multiple target collection from [2] and is outlined in further detail in the methodology section of this presentation. Ground targets have pre-determined attitudes and controller saturation is conducted. There are three actuator arrays and hybrid control (RL agent inside a control loop) executed within these simulations. In addition to the RL generated solution or torque for the actuator array of choice, off-line solutions for parameters such as initial gimbal angles and saturation limits are calculated outside the main RL environment and spacecraft simulation with the

use of numerical optimization techniques such as sequential quadratic programming. Spacecraft are rigid bodies with constant moment of inertia properties and reflect full state information availability to the system controller. This study is purely focused on exploring the possibility of using machine learning techniques for spacecraft attitude control and not for developing new reinforcement and supervised learning techniques.

1.4 Research Overview

This thesis contains five chapters to include this initial chapter. Chapter II provides a contextual background and relevant literature for the various aspects of this research in order to provide clarity and information for the reader, and to assess the current state-of-the-art. Chapter III chronicles the methodology of the study and details the proposed implementation of the reinforcement learning model. Chapter IV presents an analysis of the implemented methodology and results. The final chapter, Chapter V, concludes the thesis with a discussion of the study's findings and applicable future work. The following subsections identify and list the distinct research objectives and tasks of this study in addition to a few assumptions and limitations.

II. Background

Attitude control of aerospace systems has undergone intensive study, improvement, and development throughout the history of spaceflight [23]. Described as the process of controlling the orientation and angular rate of a spacecraft in relation to a chosen reference frame, spacecraft attitude control research was first published in 1952, with additional studies on gravitational torque, Earth magnetic field external effects, and early spin stabilization studies performed throughout the 1950s [23]. The problems of the rotating rigid body and gyroscopic effects are classical representations of rotational kinematics and dynamics.

The incorporation of multi-body systems and increasing system complexity drove the need for algorithms to accurately perform attitude control of spacecraft. Several algorithms or control laws described in Section 2.1 address the challenges and functional requirements of spacecraft to either keep one axis in an inertial or quasi-inertial direction, three-axis inertial stabilization, and three-axis aligned with a rotating reference frame [3]. Steering laws and singularity avoidance techniques are covered in Section 2.2.

Due to the real-time requirement of spacecraft flight control, new techniques have sought to improve the control authority and robustness of such systems and an increasingly popular method is through the use of machine learning [15]. The subsets of supervised and reinforcement learning are of particular interest in that such methodologies are shown to augment spacecraft attitude control systems to help accomplish a variety of missions and are covered in Section 2.3. The applications and benefits of machine learning will also be explored in Section 2.3 with respect to spacecraft control authority. Section 2.4 details the training, software, and experimental attributes of this study to conclude the background section.

2.1 Control Theory

This section introduces classical control concepts that will be later integrated with machine learning techniques. Control theory involves the control of dynamical systems in a multitude of processes. The objective is to implement a control model that exacts the required amount of control of such a system to achieve a desired performance outcome. The design of the system controller requires a mathematical representation of the plant, with such models taking on many forms to include differential equations, transfer functions, block diagrams, and state space [24]. A controller determines corrective behavior by monitoring the reference behavior or desired values and the actual produced values. Automatic control is obtained by applying the error signal, or difference between the actual and desired values, as feedback to allow the controller to correct or limit deviation of the measured value from a desired value. The following subsections discuss various attributes of control systems as applied to spacecraft attitude.

2.1.1 Equations of Motion

The equations of motion (EOM) for the actuator systems to be described in the subsequent subsection are detailed here and draw from similar notation utilized by Doupe [2]. In order to describe the EOM for attitude control of a spacecraft, Euler's equation is the starting point. Euler's rotational EOM are the basis for such descriptions, with the time derivative of the angular momentum $\vec{H}$ with respect to an inertial reference frame equal to the sum of the external moments $\vec{M}$ [3]. This relationship is found in Eq. (1) that expresses terms with respect to inertial N and body-fixed B frames

$$\dot{\vec{H}} \equiv \left\{ \frac{d\vec{H}}{dt} \right\}_N = \left\{ \frac{d\vec{H}}{dt} \right\}_B + \vec{\omega}^{B/N} \times \vec{H} \tag{1}$$

where after resolving J and ω in the body frame, $\vec{H} = J\vec{\omega}^{B/N}$ can be produced, J is the moment of inertia matrix of the body and $\vec{\omega}$ is the angular velocity vector. Subscript B denotes the spacecraft body reference frame and subscript N represents the inertial frame. Time derivatives are notated with a dot. The brackets around the second and third terms of Eq. (1) signify column vector notation describing the respective location or reference frame a term is from. This column vector notation also describes the vector head and reference point of the letters B and N in the angular velocity term, respectively. This relationship can be furthered developed through and written in an unresolved vector form as

$$\vec{M}_B = \hat{J}_B\dot{\vec{\omega}}_{BN} + \vec{\omega}_{BN} \times \hat{J}_B \cdot \omega_{BN} \tag{2}$$

where $\vec{M}_B$ is a vector of external torques applied to the body or spacecraft (s/c) and $\hat{J}_B$ is a tensor. The term $\hat{J}_B\dot{\vec{\omega}}_{BN}$ is the entire spacecraft angular momentum $\vec{h}_B$. The version of Euler's equation in Eq. (2) includes the assumptions of a rigid body spacecraft and the actuators effects are added through the use of angular momentum term separation, seen in Eq. (3)

$$\vec{h}_{tot} = \vec{h}_{s/c} + \vec{h}_{act} = J\omega_{BN} + \vec{h}_{act}. \tag{3}$$

Substituting Eq. (3) into Eq. (2) yields a version of Euler's equation that when used with an expression of an actuator angular momentum in the body frame, can be used to describe the development of equations for a single CMG.

$$\vec{M}_B = \left\{ \frac{d}{dt}\vec{h}_{tot} \right\}_N = J_B\dot{\vec{\omega}}_{BN} + \dot{\vec{h}}_{act} + \vec{\omega}_{BN} \times (J_B\vec{\omega}_{BN} + \vec{h}_{act}) \tag{4}$$

Control moment gyroscopes (CMGs) are a type of momentum exchange device found in modern spacecraft and are spinning wheels that gimbal about an axis fixed

on the spacecraft body. Input torque along the gimbal axis changes the direction of the wheel's angular momentum vector, thus producing a torque on the spacecraft in the opposite direction. Control is achieved from this gimbal action and the magnitude of the angular momentum vector is typically fixed due to the constant wheel spin rate. The inclination of the of the wheel is denoted by the angle β. There are three main types of CMGs as seen in Fig. 1: Single-gimbal CMG (SGCMG), Double-Gimbal CMG (DG-CMG), Variable-Speed CMG (VSCMG) [3].

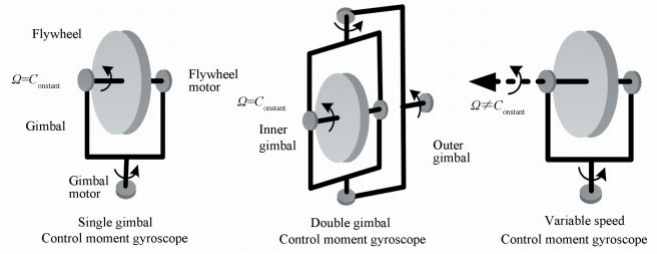


Figure 1: Various CMG arrays [1].

This research uses the term Constant Speed CMG (CSCMG) in place of SGCMG for comparison purposes to other CMG array types. For the purposes of this study, control allocation will be examined for the CSCMG, VSCMG, and Reaction Wheel CMG (RWCMG) arrays. A SGCMG has a spinning rotor constrained to rotate on a circle in a plane normal to the gimbal axis. A VSCMG has a variable rotor and an extra degree of freedom compared to conventional SGCMG/CSCMG arrays. A RWCMG array consists of three RWs and 4 CMGs. For a Single-Gimbal Control Moment Gyroscope (SGCMG) system, several additional terms must be defined and the following developments are from Doupe's research [2].

The mass moment of inertia of the CMG in the rotor frame is shown in Eq. (5) where A_r is the inertia about the non-spin axes and C_r is the rotor's inertia about the spin axis, $\hat{r_i}$. CMGs can to control motion about all 3 principal axes.

$$(J_r)_{\hat{r}} = \begin{bmatrix} A_r & 0 & 0 \\ 0 & A_r & 0 \\ 0 & 0 & C_r \end{bmatrix} \tag{5}$$

A coordinate transformation from the rotor frame to the body frame is performed using Eq. (6)

$$(J_r)_B = R_{B\hat{r}}(J_r)_{\hat{r}}R_{\hat{r}B} \tag{6}$$

where $R_{B\hat{r}}$ is the rotation matrix from the rotor wheel to the body reference frame of B. This rotation matrix is comprised of a rotation from the rotor frame to the gimbal frame by an angle of $-\psi_i$ and a rotation from the gimbal frame to the body frame by an angle of $-\delta_i$ [2]. The variables ψ_i and δ_i are the rotor and gimbal angles, respectively. The spacecraft angular velocity vector with respect to the inertial frame is defined with

$$\vec{\omega}_{RN} = \vec{\omega}_{\hat{r}B} + \vec{\omega}_{BN}. \tag{7}$$

The angular velocity of the CMG rotor with respect to the body is then

$$\vec{\omega}_{\hat{r}B} = \dot{\psi}_i\hat{r}_{r3} + \dot{\delta}_i\hat{g}_{i2} = \dot{\psi}_i R_{B\hat{g}}\hat{g}_{i3} + \dot{\delta}_i R_{B\hat{g}}\hat{g}_{i2} \tag{8}$$

where $\hat{g}_i$ are the rotor axis and the angular velocity of the body with respect to the inertial frame is $\vec{\omega}_{BN} = \omega_{i1}B_{i1}+\omega_{i2}B_{i2}+\omega_{i3}B_{i3}$. Typical assumptions that accompany CMG rotors that aid simplification of angular momentum-related expressions are detailed next. Assuming the rotor is an oblate flywheel, $C_r \approx 2A_r$. With typical CMG rotors spinning tens of thousands of revolutions per minute to provide higher torque-to-power ratios for agile maneuvering, these rotation rates are much larger in value compared to single-digit gimbal spin rates. This relationship is captured as $\dot{\psi} >> \omega_i$ and $\dot{\psi} >> \dot{\delta}_i$, where $\dot{\delta}_i$ are the gimbal slew rates and ω_i are the spacecraft

slew rates [2]. The angular momentum equation of the actuators, or in this case the single SGCMG, simplifies to Eq. (9) and excludes the minuscule angular momentum of the gimbal assembly.

$$\vec{h}_{act} = R_{B\hat{g}i} \begin{bmatrix} 0 \\ 0 \\ C_{ri}\dot{\psi}_i \end{bmatrix} \tag{9}$$

Transitioning from one SGCMG to multiples of any type of CMG as portrayed in Fig. 2 requires the $\vec{h}_{act}$ term in Euler's equation to contain the sum of each individual SGCMG [2]. This sum is represented as

$$\sum_{i=1}^{N} \left(R_{B\hat{g}i} \begin{bmatrix} 0 & 0 & C_{ri}\dot{\psi}_i \end{bmatrix}^T \right) \tag{10}$$

where there are N CMGs. The rotation matrices must also translate the gimbal orientation within the body frame, with each spacecraft CMG array requiring a unique set of rotation matrices [2].

A second type of momentum exchange device is the reaction wheel. These devices are comprised of a powered motor that produces torque on a flywheel, increasing its angular momentum and consequently causing an equal and opposite torque and angular momentum on the spacecraft. This flywheel has a fixed axis of rotation in the body frame and the angular momentum of the reaction wheel is

$$\vec{h}_i = J_{rw}\vec{\psi}_i \tag{11}$$

where $\vec{\psi}_i$ is the angular rate vector of a reaction wheel and J_{rw} is the reaction wheel moment of inertia tensor [2]. To conduct full three-axis control, a minimum of three non-coplanar reaction wheels are required. Making the assumption of an orthogonal set of three reaction wheels that are aligned with the spacecraft principal axes, the

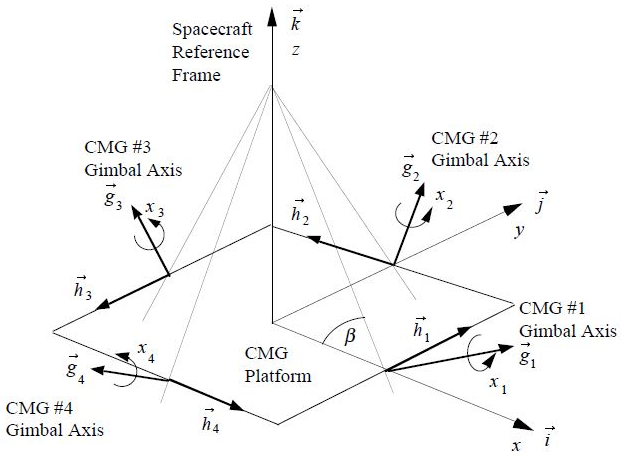


Figure 2: Pyramid mounting arrangement of four single-gimbal CMGs [2].

net angular momentum equation for the reaction wheel array can be formulated by also making use of $\vec{h}_{body} = J_B\vec{\omega}_{BN}$. The variable $\vec{h}_{act}$ is substituted for the reaction wheel array.

$$\vec{H}_{net} = J_B\vec{\omega}_{BN} + \begin{bmatrix} J_{rw}\psi_1 \\ J_{rw}\psi_2 \\ J_{rw}\psi_3 \end{bmatrix} \tag{12}$$

The internal torque applied by the reaction wheel motor is defined using Eq. (2),

$$M_{rw} = J_{rw}\dot{\psi}_{rw} \tag{13}$$

where M_{rw} is the internal torque applied by the reaction wheel rotor and $\dot{\psi}_{rw}$ is the reaction wheel acceleration. Control is performed through the adjustment of this angular acceleration in order to change the spacecraft's angular rotational acceleration $\dot{\omega}$. With CMG and RW devices described, the discussion turns to an examination of combining these two devices into a single array for the purposes of improving performance as explored in Doupe's research [2]. Such an array introduces complexities when considering the control allocation of the CMG δ_i and RW $\dot{\psi}_i$ terms.

In developing RWCMG equations of motion, VSCMG arrays are crucial to this process. The equations for VSCMG arrays can be adapted to traditional SGCMG systems by keeping the CMG rotor spin rates constant and to reaction wheel-only systems by keeping gimbal rates constant. The VSCMG equations can be further adapted to a RWCMG array by treating each actuator as a VSCMG and holding the rotor spin and gimbal rates constant. These VSCMG array equations of motion, control steering, null motion, and singularity avoidance were derived by Schuab [25] and have a differing geometric viewpoint than the equations of Section 2.1.1. In order to utilize the VSCMG arrays and RWCMG, the topic of null motion will be discussed next. The Schaub equations for VSCMG array involves null motion and directly relate to the RWCMG array. While the concept of null motion is a facet of this research, it is not further described and utilized and instead limited contextual information is provided here.

Singularities of CMG arrays have been identified and studied as impediments to their successful implementation. Some singularity avoidance algorithms use null motion to keep the Jacobian matrix from becoming singular. Null motion is described as the non-orthogonal configuration of three of more actuators that generate the ability to relay no resultant torque on the spacecraft through gimbal angle trajectories [25]. VSCMG arrays can be used to avoid singularities with its use of RW modes to always produce the required torque of a chosen control law. The motor torques of the RW have power requirements that drive the use of the VSCMG null space to avoid the singular CMG situation and complete required control [26]. The null space mapping of desired rotor accelerations and gimbal rates into the required torque of the maneuver attribute of VSCMG steering laws was used in the derivation by Schaub [25]. While the Schaub null motion VSCMG array equations are not applied to multiple target slews, the current study focuses on an RL controller ADCS system with CMG

would be incomplete without the implementation of null motion during target collection modes. These target collection modes involve some travel of the CMG gimbals along null motion trajectories toward preferred angles to prepare for the next slew. A further mission description is provided in Section 3.1. The VSCMG null space allows for the creation of combined attitude and energy storage devices and this use of such an array allows for target collection without issues brought on from singular configurations. This study seeks to examine the impact the use of an RL controller will have on the ability of a singularity avoidance techniques such as the Moore-Penrose pseudo-inverse and usage of VSCMG arrays. A RL control law has different stability properties than traditional control laws used in conjunction with chosen singularity avoidance method and resulting pseudo-inverse equations and performance comparison is one of the primary objectives of this thesis. Rotor accelerations and gimbal rates in the trajectory of the Lyapunov-based equation of an *NxN* VSCMG for null motion in Eq. (14) theoretically imparts no torque because of their existence in the null space of the weighted pseudo-inverse of Eq. (45) [25]. This equation takes on the form of

$$\begin{bmatrix} \ddot{\vec{\psi}} \\ \dot{\vec{\delta}} \end{bmatrix} = \dot{\vec{Y}} = k(WA^T(AWA^T)^{-1}A - I_{NxN})T(\Delta Y) \tag{14}$$

where $\ddot{\vec{\psi}}$ is the CMG rotor acceleration, $\dot{\delta}$ is the gimbal rate, k is a positive gain, W is the weighting matrix, I is an *NxN* identity matrix, T is a *2Nx2N* matrix of ones or zeros, and ΔY is the difference between the current and preferred states [2]. With the outline of the VSCMG array, the RWCMG array depicted in Fig. 3 first involves the use of a Jacobian matrix that comprises four CSCMGs, three RWs, and the partial derivatives of the angular momentum vector with respect to each actuator type.

The net angular momentum from a RWCMG array follows the same equation convention as Eq. (12) but the torque vector or derivative of the angular momentum

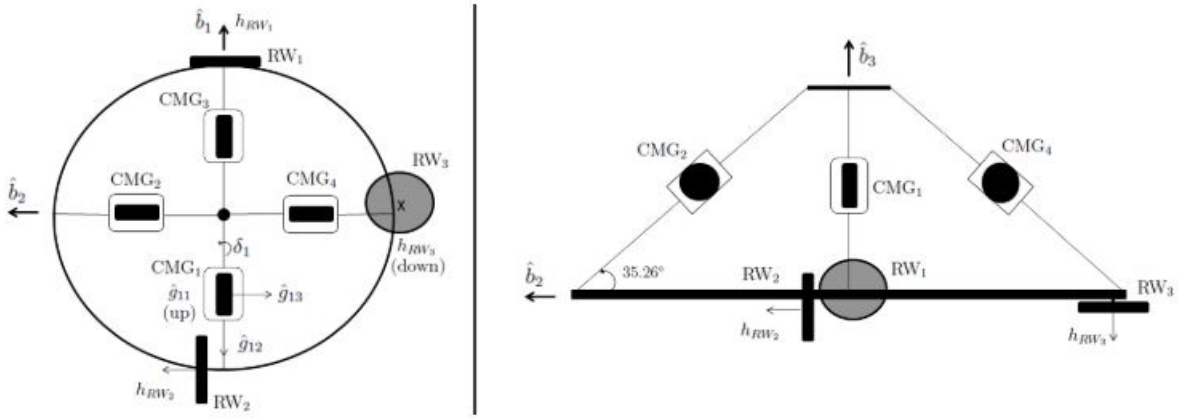


Figure 3: Simulated CMG Pyramid Geometry with RWA Layout, Top-View (left), Side-View (right) [2]

requires special calculation. With multiples of these terms, the angular momentum and the calculation of its derivative require the use of a Jacobian matrix. The components of the Jacobian matrix are found by taking the partial derivatives of the angular momentum vector with respect to each CMG δ_i and RWA $\dot{\psi}_i$. The matrix has the dimension 3xN where N is the total number of actuators in the array. For the typical pyramid of SGCMGs as in Fig. 2, the Jacobian dimension is 3x4 while for the RWCMG system the Jacobian dimension is 3x7, containing the SGCMGs and three RWs [2]. Fixing the reaction wheel momentum vector in place with respect to the spacecraft body is accomplished by holding $\dot{\delta}_i$ as zero and only considering the $\dot{\psi}$ terms. The RWCMG array Jacobian matrix takes on the following form

$$A(\dot{\vec{\psi}}, \vec{\delta}) = \begin{bmatrix} \frac{\partial \vec{h}(1)}{\partial \dot{\psi}_{RW_1}} & \frac{\partial \vec{h}(1)}{\partial \dot{\psi}_{RW_2}} & \frac{\partial \vec{h}(1)}{\partial \dot{\psi}_{RW_3}} & \frac{\partial \vec{h}(1)}{\partial \dot{\psi}_{CMG_1}} & \frac{\partial \vec{h}(1)}{\partial \dot{\psi}_{CMG_2}} & \frac{\partial \vec{h}(1)}{\partial \dot{\psi}_{CMG_3}} & \frac{\partial \vec{h}(1)}{\partial \dot{\psi}_{CMG_4}} \\ \frac{\partial \vec{h}(2)}{\partial \dot{\psi}_{RW_1}} & \frac{\partial \vec{h}(2)}{\partial \dot{\psi}_{RW_2}} & \frac{\partial \vec{h}(2)}{\partial \dot{\psi}_{RW_3}} & \frac{\partial \vec{h}(2)}{\partial \dot{\psi}_{CMG_1}} & \frac{\partial \vec{h}(2)}{\partial \dot{\psi}_{CMG_2}} & \frac{\partial \vec{h}(2)}{\partial \dot{\psi}_{CMG_3}} & \frac{\partial \vec{h}(2)}{\partial \dot{\psi}_{CMG_4}} \\ \frac{\partial \vec{h}(3)}{\partial \dot{\psi}_{RW_1}} & \frac{\partial \vec{h}(3)}{\partial \dot{\psi}_{RW_2}} & \frac{\partial \vec{h}(3)}{\partial \dot{\psi}_{RW_3}} & \frac{\partial \vec{h}(3)}{\partial \dot{\psi}_{CMG_1}} & \frac{\partial \vec{h}(3)}{\partial \dot{\psi}_{CMG_2}} & \frac{\partial \vec{h}(3)}{\partial \dot{\psi}_{CMG_3}} & \frac{\partial \vec{h}(3)}{\partial \dot{\psi}_{CMG_4}} \end{bmatrix} \tag{15}$$

where $\dot{\psi}$ is the RWA spin rate and δ_i are the CMG gimbal angles [2]. Using a pyramid of four SGCMG and three reaction wheels allows for the angular momentum to be

calculated as

$$\vec{h}_{act} = R^{bg}_{RW_1}\begin{bmatrix}0\\0\\C_{r_{RW1}}\dot{\psi}_{RW1}\end{bmatrix} + R^{bg}_{RW_2}\begin{bmatrix}0\\0\\C_{r_{RW2}}\dot{\psi}_{RW2}\end{bmatrix} + R^{bg}_{RW_3}\begin{bmatrix}0\\0\\C_{r_{RW3}}\dot{\psi}_{RW3}\end{bmatrix} \tag{16}$$
$$+ R^{bg}_{CMG_1}\begin{bmatrix}0\\0\\h_0\end{bmatrix} + R^{bg}_{CMG_2}\begin{bmatrix}0\\0\\h_0\end{bmatrix} + R^{bg}_{CMG_3}\begin{bmatrix}0\\0\\h_0\end{bmatrix} + R^{bg}_{CMG_4}\begin{bmatrix}0\\0\\h_0\end{bmatrix}$$

Computing the value of the Jacobian for Eq. (16) and assuming identical spin axis inertia C_{RW} yields

$$A(\dot{\vec{\psi}}, \vec{\delta}) = \begin{bmatrix} C_{RW} & 0 & 0 & -h_0 c\beta c\delta_1 & h_0 s\delta_2 & h_0 c\beta c\delta_3 & -h_0 s\delta_4 \\ 0 & C_{RW} & 0 & h_0 s\delta_1 & h_0 c\beta c\delta_2 & -h_0 s\delta_3 & -h_0 c\beta c\delta_4 \\ 0 & 0 & -C_{RW} & h_0 s\beta c\delta_1 & h_0 s\beta c\delta_2 & h_0 s\beta c\delta_3 & h_0 s\beta c\delta_4 \end{bmatrix} \tag{17}$$

where $s\beta$ and $c\beta$ are the sine and cosine functions of the β angle, while $s\delta_i$ and $c\delta_i$ are the respective gimbal angle functions [2]. The full spacecraft torque equation with an RWCMG array is

$$\dot{h}_{act} = A(\dot{\vec{\psi}}, \vec{\delta}) \begin{bmatrix} \ddot{\psi}_{RW1} \\ \ddot{\psi}_{RW2} \\ \ddot{\psi}_{RW3} \\ \dot{\delta}_{CMG1} \\ \dot{\delta}_{CMG2} \\ \dot{\delta}_{CMG3} \\ \dot{\delta}_{CMG4} \end{bmatrix} \tag{18}$$

where $A(\dot{\vec{\psi}}, \vec{\delta})$ is the Jacobian matrix. The angular momentum exchange that occurs between the RW and CMGs is determined by weighting matrices, which will be de-

scribed in 2.1.2 to include the control inputs from the aforementioned control devices. Kinematic relationships are an additional required set of equations for describing satellite attitude control. A commonly utilized parameterization of spacecraft orientation are Euler Parameters, or unit quaternions. Invented when researching for hypercomplex numbers that could be represented by points in three-dimensional space, the unit quaternion is formulated from Euler's eigenaxis rotation theorem [3]. A unit quaternion is a set of four parameters, a three-component vector portion and a scalar part, are a mathemtical representation of space orientations and rotations in three dimensions. This notation is ideal for use in spacecraft attitude parameterization and large rotations, avoiding singularities present in the Euler angle parameterization. Euler's theorem states that any displacement of a rigid body such that a point on the rigid body remains fixed, is equivalent to a single rotation about a fixed axis that runs through the fixed point [27]. Euler's theorem implies any two reference frames can be related through the use of a unit vector which defines the rotation axis and an angle. The rotation axis, also known as the Euler axis, is denoted by the unit vector $\hat{e}$ and rotation about this axis is denoted by the Euler angle α. A singularity occurs when relating reference frames where α equals zero or 90°, as the Euler axis is undefined. Quaternions avoid such singularities and are defined as

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} e_1 sin(\frac{\alpha}{2}) \\ e_2 sin(\frac{\alpha}{2}) \\ e_3 sin(\frac{\alpha}{2}) \\ cos(\frac{\alpha}{2}) \end{bmatrix} \tag{19}$$

where e_i are the components of $\hat{e}$ along the x,y, and z-axes. The relationship between the change in orientation expressed in quaternions and the spacecraft's angular

velocity $\vec{\omega}$ is given as

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} . \tag{20}$$

The above descriptions of equations of motion for the momentum exchange devices that form the RWCMG array forms a foundation for upcoming discussions, to include discussion of torque and controller interactions with said devices.

2.1.2 Active Control

With the evolution of sophisticated Guidance, Navigation, and Control (GNC) architectures and specific mission requirements, agile spacecraft have gained importance. Current space imagery missions require the agile satellite capabilities of minimal time slewing to maximize the amount of information collected. The attitude control system actuators commonly utilized to accomplish this slew and collect functionality are reaction wheel assemblies (RWA) and CMGs. As alluded to in the previous section and described by Wie [3], a CMG-based attitude control system takes on the following form in Fig. 4. In order to accomplish any number of objectives such

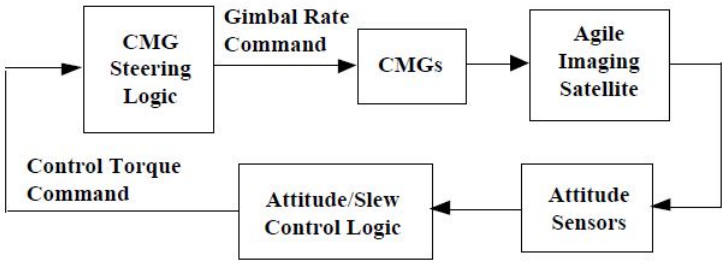


Figure 4: A CMG-based attitude control system of agile imaging satellites [3]

as complete information capture by on-board sensors, docking and close-proximity

operations, or stabilization to communicate with ground antennas a spacecraft must accomplish precise attitudes. An example of the importance of the control capabilities of an ADCS is with the International Space Station (ISS). Four 4760 Nms (35 ft-lbf-s) Double Gimbal Control Moment Gyroscopes (DGCMG) with unlimited axial gimbal freedom produce continuous attitude control for the ISS in addition to a thruster attitude control system for momentum desaturation [28]. The ISS has flown in three different orbital attitudes, each effectuating actions such as altitude re-boosts, vehicle dockings, debris-avoidance, and optimal solar ray capture. Although thrusters accomplished attitude control prior to the activation of the CMGs, such attitudes in addition to the micro-gravity environment required for science payloads would not be possible without the use of the DGCMGs. The importance and capabilities of CMGs will be further explored in this research. The application of machine learning creates challenges in an already complex system but this study employs the use of a simplified model.

Control allocation is concerned with the exacted control authority and is motivated by the desire to efficiently command a system. This research seeks to examine the control allocation for an over-actuated mechanical system, or one that contains more actuators than required to meet motion control objectives. Spacecraft control systems usually require multiple redundant actuators capable of achieving particular objectives for one or more control loops and stages under consideration. This system design may be favorable to meet fault tolerance and system configuration requirements. The task which links the control law with the particular commands to be sent to each individual actuator in the configuration is called control allocation. Control allocation algorithms are tasked with dynamically allocating the expected control from control laws among redundant efforts to obtain desired performance. Their primary objective is to compute a control input that ensures the commanded

control is produced jointly by all effectors, or actuators for all time t [14]. RW and CMG actuators take control command inputs from a controller and have corresponding limitations based on their design. RWs are unable to produce large amounts of torque. Utilizing the angular momentum contribution of a SGCMG array as

$$h^B = \sum_i^N h_i = \begin{bmatrix} \hat{a}_1(\delta_1) & \hat{a}_2(\delta_1) & ... & \hat{a}_N(\delta_N) \end{bmatrix} \begin{bmatrix} h_1 \\ h2 \\ \vdots \\ h_N \end{bmatrix} = Z(\delta) \begin{bmatrix} J_{\omega 1}\omega_1 \\ J_{\omega 2}\omega_2 \\ \vdots \\ J_{\omega N}\omega_N \end{bmatrix} \tag{21}$$

where $Z(\delta)$ contains the unit vectors in the direction of the angular momentum and δ is the gimbal angle of each CMG. The number of columns in the Z matrix equals the number of CMGs in the array. Because the CMG is spinning at a constant rate, $\dot{h} = 0$ and assuming all CMGs in the array have the same angular momentum magnitude, Eq. (22) be used to find the required torque from the system controller

$$\vec{\tau}_\omega^B = -(A(\delta)\dot{\delta}J_\omega + [\omega_{BN}^\times]Z(\delta)\vec{h}] = \vec{\tau}_{req} \tag{22}$$

where $A(\delta)$ is $\frac{\partial Z}{\partial \delta}$, $\omega_{BN}^\times$ is the skew symmetric matrix of the angular velocity, $\dot{\delta}$ is the control input to the CMGs, and $J_\omega\omega$ is the angular momentum of each wheel. The Z and A matrices are outlined below in their entirety, with β the skew-angle of the

CMGs as in Fig. 2.

$$Z(\delta) = \begin{bmatrix} -\cos\beta\sin\delta_1 & -\cos\delta_2 & \cos\beta\sin\delta_3 & \cos\delta_4 \\ \cos\delta_1 & -\cos\beta\sin\delta_2 & -\cos\delta_3 & \cos\beta\sin\delta_4 \\ \sin\beta\sin\delta_1 & \sin\beta\sin\delta_2 & \sin\beta\sin\delta_3 & \sin\beta\sin\delta_4 \end{bmatrix} \tag{23}$$

$$A(\delta) = \begin{bmatrix} -\cos\beta\cos\delta_1 & \sin\delta_2 & \cos\beta\cos\delta_3 & -\sin\delta_4 \\ -\sin\delta_1 & -\cos\beta\cos\delta_2 & \sin\delta_3 & \cos\beta\cos\delta_4 \\ \sin\beta\cos\delta_1 & \sin\beta\cos\delta_2 & \sin\beta\cos\delta_3 & \sin\beta\cos\delta_4 \end{bmatrix} \tag{24}$$

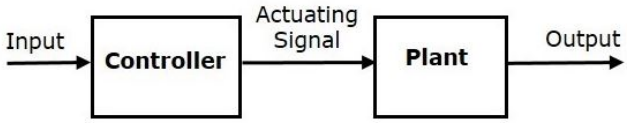
The required torque of RWA begins with the angular momentum contribution of the cluster as modeled by Eq. (18). The $\hat{a}_i$ unit vectors are fixed to the body, leading to $\dot{Z}$ equalling zero. This transforms the required torque equation derived from the general dynamic equation for a spacecraft with momentum exchange devices into Eq. (25).

$$\vec{\tau}_{\omega}^{B} = -(Z(\dot{h}) + [\omega_{BN}^{\times}]Z\boldsymbol{h}) = -\left(Z \begin{bmatrix} J_{\omega 1}\dot{\omega}_1 \\ J_{\omega 2}\dot{\omega}_2 \\ \vdots \\ J_{\omega N}\dot{\omega}_N \end{bmatrix} + [\omega_{BN}^{\times}]Z \begin{bmatrix} J_{\omega 1}\omega_1 \\ J_{\omega 2}\omega_2 \\ \vdots \\ J_{\omega N}\omega_N \end{bmatrix} \right) \tag{25}$$

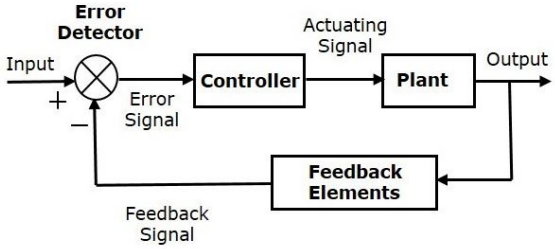
The above discussion outlines the required angular momentum by RW and CMG arrays to counteract disturbances (or for reorientation) affecting a spacecraft's attitude. Computing the control input to these arrays to produce the corresponding steering laws and CMG singularity avoidance are detailed in Section 2.2. The control laws of the system controllers tested with multiple actuator systems as performed in [2] were used so their results could be compared to a reinforcement learning agent functioning as a machine learning controller as outlined in the next subsection.

2.1.3 Control Laws

A control system can be categorized as open-loop (OL) and closed-loop (CL) as depicted in Fig. 5, where controllers are either required to accomplish feedback or non-feedback control [29]. A closed-loop system feeds the actuating error to the controller as to reduce error and produce desired values. In contrast, an open-loop system does not utilize the output to impact the control actions and accomplishment of desired outputs relies on the known input-output relationship in the absence of internal or external disturbances. In order to determine the desired excitation based on the error signal, controllers (or control laws) are used.



(a) Open-Loop System



(b) Closed-Loop System

Figure 5: Block diagrams of notional open and closed-loop systems

Such control law algorithms are used by a controller to process and determine an output amount to be sent to the system plant, or the relation between the input and output signal [30]. A control law of interest and used for comparison purposes to the RL controller is the quaternion feedback control law. Wie details the use of quaternion feedback controllers for spacecraft performing minimum-time slew maneuvers and by adding slew rate constraints, cascade saturation control logic was formulated. Wie [31] presented a three-axis quaternion feedback (QF) controller with slew rate and

control saturation limits, testing and executing two successive large maneuvers to demonstrate robustness. The controllers of the simulations used a cascade-saturation logic where the quaternion states do not appear in the angular rate state equations. Equations (26) through (28), an adaptation from Wie's formulation, will be used in this research

$$\dot{\underline{q}} = -\frac{1}{2}\vec{\omega} \times \underline{q} \pm \frac{1}{2}\sqrt{1 - ||\underline{q}||^2}\vec{\omega} \tag{26}$$

$$\dot{q}_4 = -\frac{1}{2}\vec{\omega}^T \underline{q} \tag{27}$$

$$\dot{\vec{\omega}} = J^{-1}\left(-\vec{\omega} \times J\vec{\omega} + \vec{u}\right) \tag{28}$$

where $\underline{q}$ is a vector consisting of q_1, q_2, and q_3. In addition to the standard QF controller, a second controller Wie [3] derived is the saturated QF controller given as

$$\vec{u} = -sat[Ksat(P\vec{q_e}) + C\vec{\omega}] \tag{29}$$

where K, P, and C are gains that can be defined based on the desired damping ratio and natural frequency of the response [31]. Wie demonstrated without loss of generality, several closed-loop nonlinear systems with these gain selections and control logic combinations which are globally asymptotically stable. The relationships between these gains and response parameters are outlined in Wie's following methods for the quaternion feedback controller, where k and c are positive scalar constants, J is the moment of inertia matrix, and ω_n is the nautral frequency [3].

$$\begin{array}{llll} C = 2\xi\omega_n J & c = mean(diag(\frac{C}{I})) & P = k\frac{K}{J} & KP = 2\omega_n^2 J \\ k = mean(diag(\frac{KP}{J})) & k_i = c\frac{|q_i(0)|}{||\vec{q}(0)||}\omega_{max} & K = diag(k_1, k_2, k_3)J & \end{array} \tag{30}$$

Wie used a saturation function defined as the following to define a control input $\vec{u}$

[31]

$$\underset{L_i}{sat}(\vec{x}) = \begin{bmatrix} \underset{L_1}{sat}(x_1) \\ \underset{L_2}{sat}(x_2) \\ \vdots \\ \underset{L_n}{sat}(x_n) \end{bmatrix} \tag{31}$$

where $\underset{L_i}{sat}(x_i)$ is defined as

$$\underset{L_i}{sat}(x_i) = \begin{cases} L_i & if \quad x_i > L_i \\ x_i & if \quad -L_i \leq x_i \leq L_i \\ -L_i & if \quad x_i < L_i \end{cases} \tag{32}$$

The cascade saturation control logic uses the slew rate limit L_i computed from $(c/2k)min(\sqrt{4a_i|q_{ei}|}, |\omega_i|_{max}$, where a_i is the maximum control acceleration limit. This logic is detailed as

$$\vec{\tau} = -J^{-1}(2k\underset{L_i}{sat}\left(\vec{q}_e + \frac{1}{T}\int \vec{q}_e\right) + c\vec{\omega}) \tag{33}$$

$$\vec{u} = \underset{U}{sat}(\vec{\tau}) \tag{34}$$

2.2 Singularity Avoidance

Reaction wheel and control moment gyroscopes are commanded by a system control law but are subject to physical and design limitations. The maximum available torque of a reaction wheel is regulated by the torque generated by the motor. The use of a larger motor bypasses this limitation but imposes increased electrical and weight requirements. The steering law that commands a RWA must therefore be selected in conjunction with saturation logic that allows for the generation of effective torque. A

CMG array can overcome the size, weight, and power limitations of a RWA. While the torque envelope is larger, a performance difficulty this momentum exchange device encounters are singularities. A singularity occurs when the gimbals of a CMG array are respectively orientated so that each of the torque vectors produced by moving the gimbals is parallel to each other. In this instance, no usable output torque is produced. The calculations for CMG steering laws require the use of the pseudo-inverse to avoid these singular states, although these approaches do not guarantee singularity avoidance. A discussion of RWA and CMG steering laws, singularity avoidance methods, and machine learning applications will take place in this section.

2.2.1 Singularity Types

There exist several types of singularities for SGCMGs as exemplified by Fig. 6. The CMG arrays of this research will be encountering elliptic singularities, or null solutions to the gimbal angles that do not exist for a specific point of CMG angular momentum space. These singularities can be escaped by perturbing the angular momentum to induce torque error in the system.

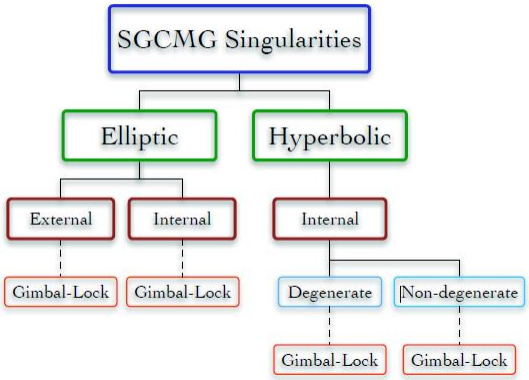
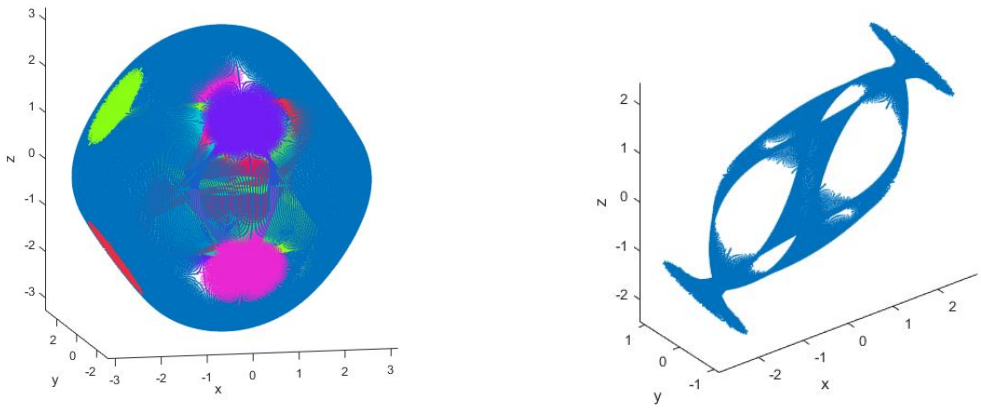


Figure 6: Singularities for SGCMGs [4]

When the angular momentum vectors of Fig. 2 all lie in the two-dimensional plane of the CMG platform, an external singularity and gimbal lock occurs. With all gimbal

transverse axes lying in the same plane as the desired commanded torque, gimbal lock is an undesired condition that can be avoided by over designing rotor sizes, gimbal β angles, and using the physical properties of the spacecraft. Different arrangements of CMGs to control motion about all three principal axes for redundancy and failure accommodation purposes, with the most commonly studied being the four CMGs in a pyramid configuration. The CMG configuration also plays a role in singularity avoidance through the selected steering logic employed to avoid the geometric singularity problem inherent to CMGs. The gimbal array angular momentum capability is depicted as a three dimensional surface. Saturation singularities or external singularities are elliptic in nature and occur on the surface of the angular momentum envelope, exemplified in Fig. 7. Desired angular momentum values exist on the outer surface of the external surface map while the unachievable angular momentum values are captured as the holes around the gimbal axes. Gimbal lock is represented in the internal surface maps as a hole. It is of the interest for the research herein to observe an RL controller's ability to avoid singularities. To the author's knowledge, no such research has been conducted.



(a) External Angular Momentum Map

(b) Internal Angular Momentum

Figure 7: Maps of Pyramidal Four-CMG configuration Unit Angular Momentum Space [5]

2.2.2 Steering Laws

With the dynamics and controller equations defined, the steering logic and singularity avoidance will now be discussed. Reaction wheel steering laws are formulated by first taking the required torque represented by Eq. (22) and solving for the control input $\dot{h}$

$$\dot{h} = Z^{-1}(-\tau_{req} - [\omega_{BN}^{\times}]Zh) \tag{35}$$

which requires Z to be full rank or requires use of the pseudo-inverse. The traditional Moore-Penrose pseudo-inverse of Eq. (36) is one such method, where solving Eq. 14 for reaction wheel accelerations and gimbal rates from this equation produces numerical singularities,

$$\dot{\vec{Y}} = A^T(AA^T)^{-1}\dot{\vec{h}}_{act} \tag{36}$$

The steering logic for SGCMGs will be outlined to serve as a basis for CMG arrays. The control input $\dot{\delta}$ is computed from the commanded torque from the controller

$$\tau_{\omega}^{B} = -(A(\delta)\dot{\delta}J_{\omega}\omega + [\omega_{BN}^{\times}]Z(\delta)h) \tag{37}$$

where $\dot{Z}(\delta) = \frac{\partial Z}{\partial \delta}\dot{\delta} = A(\delta)\dot{\delta}$ and assuming all CMGs in the array have the same angular momentum magnitude and that the wheels are spinning at constant rates. The matrix $A(\delta)$ requires inversion and the pseudo-inverse as with the RWA and several methods are outlined in Section 2.2. A notable example is the singularity robust inverse. The singularity condition or singularity measure m for this is $\sqrt{det(AA^T)}$ and singularity avoidance is not guaranteed using pseudo-inverse-based approaches. Several papers detail better forms of the inverse calculation and singularity avoidance for SGCMGs [32] and VSCMGs ([33],[34]) and the next subsection continues the discussion of the

$\dot{\vec{Y}}$ and $A^{\dagger}$ calculations. The resulting CMG steering law is given in Eq. (38)

$$A(\delta)\dot{\delta} = \frac{1}{J_{\omega}\omega}(-\tau_{req} - [\omega_{BN}^{\times}]Z(\delta)h)] \tag{38}$$

2.2.3 Techniques and Methods

With an introduction to steering laws, developing the RWCMG approach requires additional descriptions of other singularity avoidance methods and an overview of VSCMGs. Control algorithms treat internal CMG singularities that numerically occur when solving for the variables of $\dot{\vec{Y}}$. These methods constrain the angular momentum and/or gimbal angles envelope visualised in Fig. 8 or apply null motion, to avoid/escape singularities. Using only null motion cannot avoid internal or elliptical singularities. The momentum of the CMGs is constrained to these surfaces with the envelope representing the available values. When the CMG rotors are gimballed to achieve maximum values for angular momentum while reaching the edge of this envelope, saturation occurs and external torque must be provided to return the CMG array to a fully controllable state. Several singularity methods are outlined next and are taken from the survey and research performed by Leve [4]. Figure 8 summarizes these methods and a few will be described from the sections of null motion, singularity escape, and singularity avoidance and escape algorithms.

As mentioned previously, a singularity measure $m = \sqrt{det(AA^T)}$ can be used to detect the distance from singularity. The use of such a measure falls within the null motion algorithm type of local gradient (LG). These methods use null motion to keep the Jacobian matrix from becoming singular by using the null vector d of Eq. (39) to maximize an objective function $f = 1/m$. Minimizing this objective function

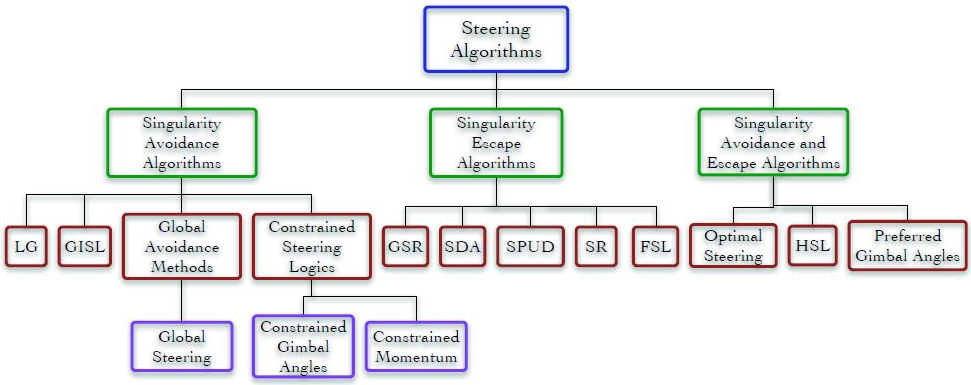


Figure 8: Steering Algorithms used when encountering singularities [4]

maximizes the singularity measure.

$$d = \nabla_\delta f = \frac{\partial f}{\partial m}\left(\frac{\partial m}{\partial \delta}\right)^T = \frac{-1}{m^2}\left(\frac{\partial m}{\partial \delta}\right)^T \tag{39}$$

Another null motion algorithm is the Generalized Inverse Steering Law (GISL), which is a variation of the Moore-Penrose pseudo-inverse. This methods uses another Jacobian matrix B which has each of its columns orthogonal to the associated column of the original Jacobian matrix A. The B matrix does not eliminate internal singularities and only serves to add null motion. With a four-CMG pyramid arrangement, the A and B matrices are

$$A = \begin{bmatrix} -c\theta c\delta_1 & s\delta_2 & c\theta c\delta_3 - s\delta_4 & \\ -s\delta_1 & -c\theta c\delta_2 & s\delta_3 & c\theta c\delta_4 \\ s\theta c\delta_1 & s\theta c\delta_2 s\theta c\delta_3 & s\theta c\delta_4 & \end{bmatrix} \tag{40}$$

$$B = \begin{bmatrix} -c\theta s\delta_1 & -c\delta_2 & c\theta s\delta_3 c\delta_4 & \\ c\delta_1 & -c\theta s\delta_2 & -c\delta_3 & c\theta s\delta_4 \\ s\theta s\delta_1 & s\theta s\delta_2 s\theta s\delta_3 & s\theta s\delta_4 & \end{bmatrix} \tag{41}$$

where s and c denote sine and cosine of the pyramid skew angle θ and δ_i are the

gimbal angles. The resulting pseudo-inverse containing these matrices is

$$A^{GISL} = (A + B)^T (A(A + B)^T)^{-1}. \tag{42}$$

The singularity robust inverse (SR) is another variation of the Moore-Penrose pseudo-inverse and a type of singularity escape algorithm which adds torque error to pass through or escape an internal singularity [4]. It contains a positive definite matrix λI that is an identity matrix scaled by the singularity parameter $\lambda = \lambda_0 exp^{-\mu m^2}$ that is typically on the order of 10^{-2} and the positive semi-definite matrix AA^T. If the rank of the A matrix or equivalent rank AA^T is less than 3 for a set of gimbal angles, the pseudo-inverse does not exist. This inverse has the form

$$A^{SR} = A^T (AA^T + \lambda I)^{-1} \tag{43}$$

and is able to escape both hyperbolic and elliptical singularities. In the area of singularity avoidance and escape algorithms that avoid singularities through null motion and use of torque error, preferred gimbal angles are one such method. These angles refer to the set of initial SGCMG gimbal angles that can be reached by null motion, with the word *preferred* denoting how the maneuvers originating from the angles avoid a singular configuration and therefore these angles are preferred. The attitude EOM and backwards integration of the SR inverse are used to find the preferred gimbal angles and the method cannot avoid singularities if the initial set is $\delta_0 = [45 \quad -45 \quad 45 \quad -45]^T$ deg [2]. With the null space projection matrix undefined at a singularity, the SR inverse is used in place of the Moore-Penrose pseudo-inverse of A and the system instead adds torque error at the singularity. Eq. (44) describes this relationship

$$\dot{\delta}_n = [1 - A^{SR} A]d \tag{44}$$

and the knowledge that traveling from point to point in gimbal space is not possible through standalone null motion due to absence of n dimensions of the null space.

The EOM for VSCMGs found in [2] are directly applicable to the RWCMG array. The Schaub [25] derived equations along with steering logic and singularity avoidance will be referenced for the remaining discussion on RWCMGs. The VSCMG array behaves as a SGCMG by holding a CMG rotor rate $\dot{\psi}_i$ constant while by conversely holding a VSCMG gimbal rate $\dot{\delta}_i$ constant, the VSCMG will behave as a reaction wheel. In addition to these EOM, [8] outlined a weighted Moore-Penrose inverse solution where the square diagonal matrix of weights W determine which actuator set (CMG versus reaction wheel) is active per operation time step. This inverse solution is

$$\begin{bmatrix} \ddot{\vec{\psi}} \\ \dot{\vec{\delta}} \end{bmatrix} = WA^T(AWA^T)^{-1}\dot{\vec{h}}_{act} \tag{45}$$

where the number of terms in W equals the number of terms in $\dot{\vec{Y}}$. Setting a weight to zero turns the affected mode off and this inverse solution allows the CMG to avoid singularities caused by rank deficiencies in the A matrix. To determine the closeness of the system to a singularity, the weights are made dependent on the proximity to a SGCMG singularity

$$\nu = det\frac{1}{h_0^2}(A(\vec{\delta})A(\vec{\delta})^T) \tag{46}$$

where $A(\delta)$ corresponding to the N SGCMGs of the N columns of the A matrix and h_0 is a nominal CMG angular momentum that is the same for all CMGs. Schaub developed the following form of the weighting matrix for the VSCMG array in Eq.

(48) that is adjusted by the singularity nearness metric ν in Equation (46)

$$W = \begin{bmatrix} W_{RW_1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & W_{RW_2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & W_{RW_3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{47}$$

where $W^0_{RW_i}$ and μ are positive scalars chosen by the system designer and

$$W_{RW_i} = W^0_{RW_i} e^{-\mu\nu}. \tag{48}$$

This weighting matrix is adapted for use in the RWCMG system for purposes of this study and as established in [2]. Approaching a singularity will cause the RW mode usage to increase to reduce the possibility of excessive gimbal rate values and that the VSCMG produces the required control torque. Varying the rotation rates increases the null space dimension from one to five dimensions which proves to be advantageous in having extra degrees of freedom in which to maneuver the gimbal angle trajectories. Traditional CSCMGs contain only one degree of freedom in which to conduct null motion. The arrays used in this research constitute over-actuated or over determined systems as four gimbal rates are used to control three axes. With Eq. (45) through (47) and traditional singularity avoidance methods outlined, the RWCMG system has a basis for controlling the spacecraft and tracking singularities. The purpose of this study is to examine improvements to known methods and the next subsection begins this discussion.

2.3 Machine Learning Overview

Machine and deep learning are useful techniques that rapidly incorporate learned data representations to accomplish multifarious tasks human beings are unable to complete in standard and useful time frames. These subsets of artificial intelligence are inspired by biological intelligence information processing, which maintains a type of internal model while incrementally processing information and constantly updating the model as new information enters.

2.3.1 Supervised Learning and Neural Networks

Several approaches can be taken by computers to discover and learn how to perform tasks without explicit programming instructions. These approaches primarily take the form of four techniques that depend on the nature of the input and fed-back output available to the learning system. Unsupervised, semi-supervised, and reinforcement learning serve as three algorithms. A fourth method of supervised learning is described in this section and for reference to this study. The goal of supervised learning is for a computer to learn to map input data to known targets given a set of samples [16]. Examples of supervised learning problems include object detection, sequence generation as for predicting a caption to describe a picture, and syntax tree prediction (or predicting a sentence's decomposition into a syntax tree). A knowledgeable external supervisor, or the designer, provides supervised learning architectures with a set of labeled examples for training and eventual testing purposes. A diagram of this process is provided in Fig. 9,

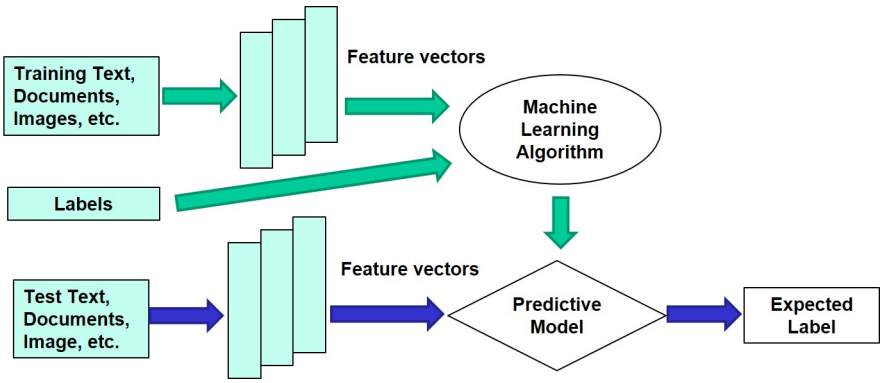


Figure 9: Supervised Learning Model [6],

where the designer is responsible for selecting and providing the training data with matching labels while the machine selects which features to glean from this data set in order to learn the appropriate relationships. The algorithm is tasked with extrapolating its responses in order to perform the desired response in situations not present in the training set, which is assessed by using a test data set. Data preparation and preprocessing is a key attribute to ensuring the algorithm is able to learn data representations in the presence of noise and/or missing feature values. After data selection, input feature representation, and the structure of the learned function, the designer selects a learning algorithm to best solve the problem at hand. Types of supervised learning algorithms include classification [35], scalar regression [36], and active learning [37]. The designer is responsible for utilizing the appropriate algorithm, as each has its own strengths and weaknesses. More minute examples and descriptions of their intricacies include logic (symbolic) learning methods such as decision tress and rule-based classifiers, statistical learning algorithms such as naive Bayes classifiers and Bayesian networks, and support vector machines [38].

The designer then runs the learning algorithm on the gathered training data set, adjusts certain control parameters to optimize performance, and evaluates the accuracy of the learned function. The steps outlined here describe the classical statistical machine learning approach and require many designer inputs and computational

steps. Such systems encountered many problems with data acquisition and representation, logical and symbolic representations of complex problems, and difficulty with large-scale implementation. These problems caused what is considered an AI winter, a period in the 1970s and 1980s in which reduced funding and interest in artificial intelligence research was prominent [39]. Practical neural network technology was realizable starting in the 1980s due to the development of a metal-oxide-semi-conductor (MOS) very-large-scale integration (VLSI) technology [40]. This technology allowed scientists to find a way to efficiently train large networks through the use of the Back Propagation algorithm, which uses gradient-descent optimization to train chains of parameteric operations. Neural networks and artificial intelligence encountered a second AI winter in the 1990s and emerged in the 2010s with important breakthroughs and have become important algorithms for solving difficult large-scale problems such as those in the fields of computer vision and natural-language processing [16]. An artificial neural network (ANN), loosely modeling the neurons in a mammalian brain, contains several connected nodes called artificial neurons. An initial attempt to exploit this biological architecture, ANN's evolved into networks capable of improving various performance results and performing tasks conventional algorithms had little success achieving. The artificial neurons of an ANN are connected to each other in various patterns, forming a directed, weighted graph. An example of such a architecture is illustrated in Fig. 10, where input, hidden, and output layers are categorized.

Several components comprise an ANN and its basic unit is an artificial neuron. Various inputs to the network, contained in the input layer, are multiplied by a connection weight. These products are then summed, fed through a transfer function inside the neuron to generate a result, and then output. An example of a transfer function is a sigmoid transfer function, which takes the value from the summation function and turns it into a value between zero and one. Multiple neurons form

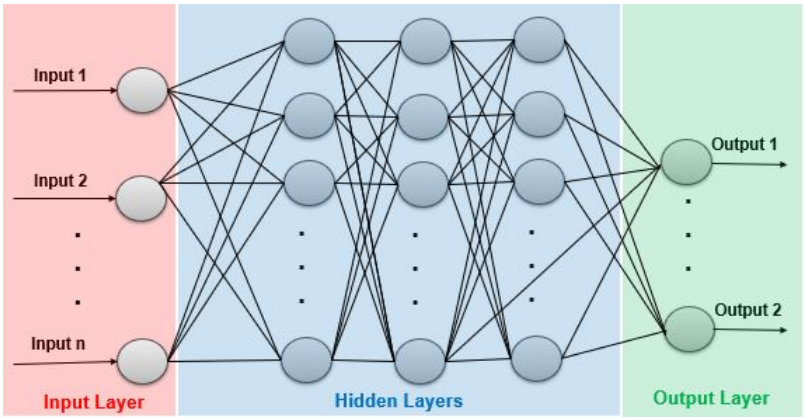


Figure 10: Artificial neural network architecture [7]

individual layers and the first layer is known as the input layer [41]. The input layer receives data from files, electronic sensors, or other type of source. The hidden, or internal layers exist between the input and output layers and serve to produce an output to a desired result using mathematical functions. The output layer sends the cumulative ANN information to an outside process or device. The connections between the neurons and layers are important aspects of neural networks and dictate the operation, performance, and problem solving nature of these structures. Changing the parameters concerning these connections is the responsibility of the designer to use an ANN to generate applicable solutions. While supervised learning is applicable to many kinds of problems, it is an inadequate solution in situations where problems require learning from interaction. It is often impractical to collect examples of desired behavior that represents all possible situations in interactive problems, leading to the need for the machine to learn from its own experience and introduces the concept of reinforcement learning.

2.3.2 Reinforcement Learning

Reinforcement learning (RL) refers to how to map circumstances to actions as to maximize some total reward, or having a goal-oriented agent interacting with an

environment. This agent is not specifically told which actions to take, but must learn which actions yield the most reward in the environment. At each iterative step, the agent gets evaluative feedback about the performance and its actions and is therefore reinforced, either positively of negatively, to improve the performance of subsequent actions. RL stems from dynamical systems theory and the optimal control of finite-state and action, discrete-time Markov decision process (MDP). These processes include the three aspects of a sensation, action, and goal and several RL methods have been developed to learn optimal policies for MDPs [7]. Figure 7 contains the MDP structure. As classical formalizations of sequential decision making, MDPs serve as mathematically idealized forms of reinforcement learning problems. A reinforcement learning agent has the objective of maximizing the cumulative reward for a task goal through interactions with an environment, taking actions and updating states in order to do so. Most reinforcement learning algorithms estimate value

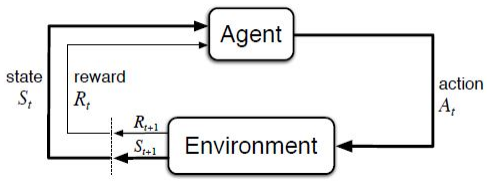


Figure 11: The agent-environment interaction in a Markov decision process [7]

functions, or functions of states or state-action parts that estimate how well the state is for the agent. The evaluation of "how good" the state is determines the future rewards that the agent can expect and the value function represents the reward/penalty accumulated by the agent in the long-term. Value functions are defined with respect to a policy, which is a mapping from states to probabilities of selecting each possible action [7]. For deterministic MDPs, the state-value function is formally defined as

$$v_\pi \doteq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right], \forall\ s \in\ S \tag{49}$$

where $\mathbb{E}[\cdot]$ is the expected value of a random variable given the agent follows the policy π for multiple timesteps t, k is the number of steps, R is the reward, γ is the discount rate, and S_t is the current state. While an agent is guided by a policy, it must have some way in which to be reinforced. This generates the need and utilization of a reward. This type of signal, passed from the environment to the agent, describes how the agent should behave in order to maximize the total amount of received reward. A formal description of this signal is contained within what is considered the reward hypothesis [42], which states that the maximization of the expected value of the cumulative sum of a received scalar signal (called reward). The cumulative reward, or return $R(\tau)$, is defined as the sum of rewards obtained by the agent at each time step

$$r_t = R(s_t, a_t, s_{t+1}) \tag{50}$$

$$R(\tau) = \sum_{t=0}^{T} r_t \tag{51}$$

where the reward at a timestep r_t is comprised of the current state s_t, the action just taken a_t, s_{t+1} [43]. There are four different formulations of return functions, combinations of discounted or undiscounted, finite or infinite horizon. One example formulation is the finite-horizon discounted return of Eq. (51), which is the obtained sum of rewards in a fixed window of steps. Another formulation is the infinite-horizon discounted return, which is a discounted sum of all rewards ever obtained by the agent and theoretically to infinity. This discount factor γ allows for the maximization of the return summed to infinity by preventing the return from reaching infinity through varying importance placed on immediate and future rewards [43]. The value of the discount factor is $\gamma \in [0, 1]$, where a small discount factor closer to 0 implies placing greater importance to immediate rewards than to future rewards and a large

value closer to 1 implying greater importance to future rewards. The infinite-horizon discounted return is

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t. \tag{52}$$

The value of taking action a in state s under a policy π, expressed as $q_\pi(s, a)$, is the expected return starting from a state and taking an action. The action-value function for a policy π is as follows [7]:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t | S_t = S, A_t = a] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right] \tag{53}$$

The above value functions are estimated from the agent's experienced environment interactions in order to reach the objective of determining a policy which maximises the value function. The value function v_π serves as a solution to Bellman's equation that averages over all possibilities of a state value and the values of its successor states [7]. Such an equation contains the relationships between a state and future possibilities, where each probability has an associated weight. The expression of v_π using Bellman's equation is

$$v_\pi = \sum_a \pi_{(a|s)} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_{pi}(s')\right], \forall\ s\ \in\ S. \tag{54}$$

where p is the probability. It is implicit that the actions a are taken from an action set, the next states $s^{'}$ are taken from a set of states, and that the rewards are taken from a reward set. The sum of the above equation is over all values of $a, s^{'}$, and r, as well as their probabilities and weights in order to generate an expected value [7]. The optimal value functions for the state and actions can be expressed using the Bellman optimality condition, which is defined as the value of a state under and optimal policy must equal the expected return for the best action from that state. The equations of

v_* and q_* are state and action value functions for a policy, respectively [7]:

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \tag{55}$$

$$q_*(s, a) = \sum_{s',a} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')]. \tag{56}$$

Finite MDPs have a unique solution for the v_*. The Bellman optimality equation of Eq. (55) is a system of equations, where each equation corresponds to a state. The known dynamics of the environment can be solved as a system of equations for v_* and then for q_*. An optimal policy can then be determined with this calculated information. The Bellman relations of Eqs. (55) and (56) form the basis of all RL methods [7]. These methods fall into two categories: model-based and model-free. Model-based methods rely on planning and are therefore offline algorithms that require complete environmental knowledge. Model-free methods rely on the component of learning and are based on temporal difference (TD). Temporal-difference methods are online and do not use an explicit system model, relying on process data to adjust the utility of temporally different successive state estimates [44]. Algorithms that can be used to compute optimal policies given a correct environmental model as a MDP define the term dynamic programming (DP). This method is not a learning method but instead a computational method used for determining optimal behavior [42]. DP serves as a theoretical foundation, albeit of limited utility, for reinforcement learning. DP's main principle is the use of value functions to aid in the search for good policies. Turning the Bellman equations into assignments allows for obtaining DP algorithms that will improve approximations of the desired value functions. A non-exhaustive taxonomy of RL methods compiled by Open AI [8] is provided below and a few examples of popular methods will be discussed. Two such methods include DP policy optimization and Q-learning.

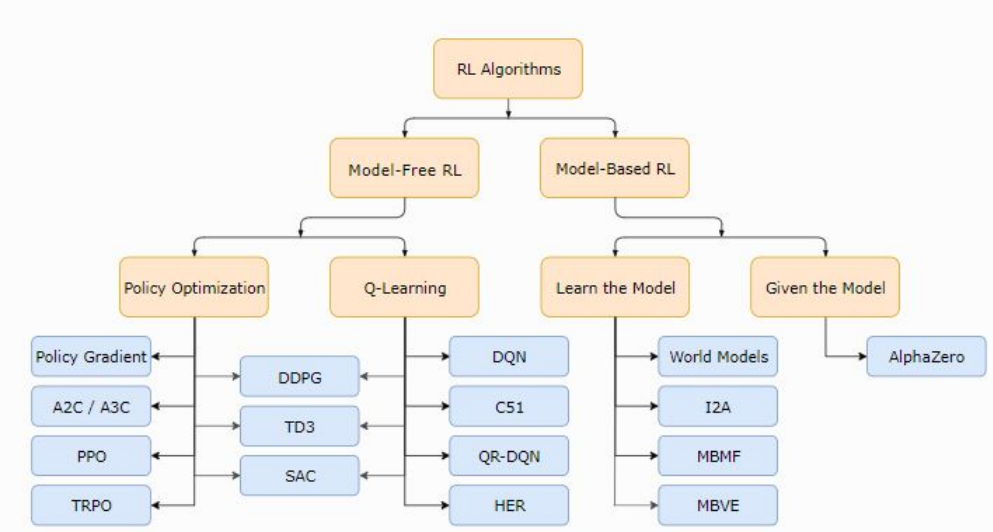


Figure 12: Modern RL Algorithms [8]

An example of a model and DP-based algorithm is policy iteration, which alternates between policy improvement and policy evaluation [45]. A resulting sequence of uniformly improving policies and value functions can be obtained by first starting with an initial policy, estimates the value function (policy evaluation) by looping through each state, and then improves the policy using each state estimate to determine the best value function estimate. Due to the utilization of a greedy search, each policy is guaranteed to be a strict improvement over the previous one, unless the previous policy is already optimal [7]. The sequence and Bellman expectation equation updates to converge to the value function are provided in Eq. (57), where $\overset{\text{E}}{\rightarrow}$ denotes a policy evaluation and $\overset{\text{I}}{\rightarrow}$ a policy improvement.

$$\pi_0 \overset{\text{E}}{\rightarrow} v_{\pi 0} \overset{\text{I}}{\rightarrow} \pi_1 \overset{\text{E}}{\rightarrow} v_{\pi 1} \overset{\text{I}}{\rightarrow} \pi_2 \overset{\text{E}}{\rightarrow} \ldots \overset{\text{I}}{\rightarrow} \pi_* \overset{\text{E}}{\rightarrow} v_* \tag{57}$$

Q-learning algorithms utilize an approximator $Q(s, a)$, or Q-factor that contains state-action pairs in place of the state value function $v(s)$. This Q-iteration algorithm

is an off-policy TD control algorithm [46] defined by

$$Q(S_t, A_t) \longleftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \tag{58}$$

where α is the step size and the learned action-value function, Q, directly approximates q_* independently of the followed policy (off-policy). The optimal action-value function of q_* is found after performing a greedy search stemming from an action corresponding to a state. Such a process simplifies the analysis of the algorithm and Q has been shown to converge with a probability of 1 to q_* [47].

This research utilizes a policy gradient algorithm to solve the RL problem associated with the general attitude control problem. Policy gradient algorithms such as Deep Deterministic Policy Gradient (DDPG) [48] and REINFORCE [49] are commonly used algorithms in solving problems with continuous action spaces. They function by representing the policy as a parameteric probability distribution $\pi_\theta = \mathbb{P}[a|s;\theta]$ that stochascially selects some action a in state s according to parametric vector θ [50]. A policy defines the behavior of the RL agent in an environment by telling the agent what action to perform per state. Policies can be classified into two types: stochastic and deterministic. Stochastic policies maps states to a probability distribution while deterministic policies $a = \mu_\theta(s)$ direct the agent to perform one particular action in a state [50]. For solving the problem of the general spacecraft attitude control problem, a deterministic policy is a more appropriate choice of policy than a stochastic one as all information is available to the ADCS and no sources of randomness are introduced into the system. The deterministic policy gradient theorem, modified from the stochastic gradient theorem, considers a deterministic policy $\mu_\theta : \mathcal{S} \longrightarrow \mathcal{A}$ with parameter vector $\theta \in \mathbb{R}^n$ [50]. The right arrow describes a state corresponding with an action. The performance objective is $J(\mu_\theta) = \mathbb{E}[r_1^\gamma|\mu]$, the probability distribution is $p(s \longrightarrow s'$,t, $\mu)$ and the discounted state distribution

is $\rho^{\pi}(s') := \int_{\mathcal{S}} \sum_{t=1}^{\infty} \gamma^{t-1} p_1(s) p(s \longrightarrow s', t, \pi) ds$ [50]. The performance objective is written as an expectation in Equation (59).

$$J(\mu_\theta) = \int_{\mathcal{S}} \rho^{\mu}(s) r(s, \mu_\theta(s)) ds = \mathbb{E}_{s\ \rho^{\mu}}[r(s, \mu_\theta(s))] \tag{59}$$

The final form of the deterministic policy gradient theorem is in Eq. (60) and implies that the $\nabla_\theta \mu_{theta}(s)$, $\nabla_a Q^{\mu}(s, a)$, and the deterministic policy gradient exists [50].

$$\begin{aligned} \nabla_\theta J(\mu_\theta) &= \int_{\mathcal{S}} \rho^{\mu}(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu}(s, a)\Big|_{a=\mu_\theta(s)} ds \\ &= \mathbb{E}_{s\ \rho^{\mu}}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu}(s, a)\Big|_{a=\mu_\theta(s)}\right] \end{aligned} \tag{60}$$

As stated previously, policy gradient algorithms are comprised of a policy model and value function. It is advantageous to learn both the value function and policy and simultaneous learning of these components is captured by the actor-critic method. The actor model updates the policy parameters θ for π_θ(a—s) in the direction suggested by the critic (policy learning) while the critic updates the value function parameters according to the policy algorithm (value function learning) [7]. As a TD method, the actor-critic method has the actor select actions and the critic criticise the actions made by the actor. This form of on-policy learning has the critique represented by a TD error, or scalar signal that drives the learning in both actor and critic. Fig. 13 captures the actor-critic architecture. This research will utilize an actor-critic architecture to solve a problem with continuous-valued actions to save on computation time and handle several domain-specific constraints. Reinforcement learning can be compared to the objective of an optimal controller, which similarly aims to optimize a long-term performance criterion. This similarity lends itself to the connection and application of RL to optimal control problems. How reinforcement learning can solve optimal control problems will be explored in the next section as well as implementa-

tion, which motivate the research performed in this study.

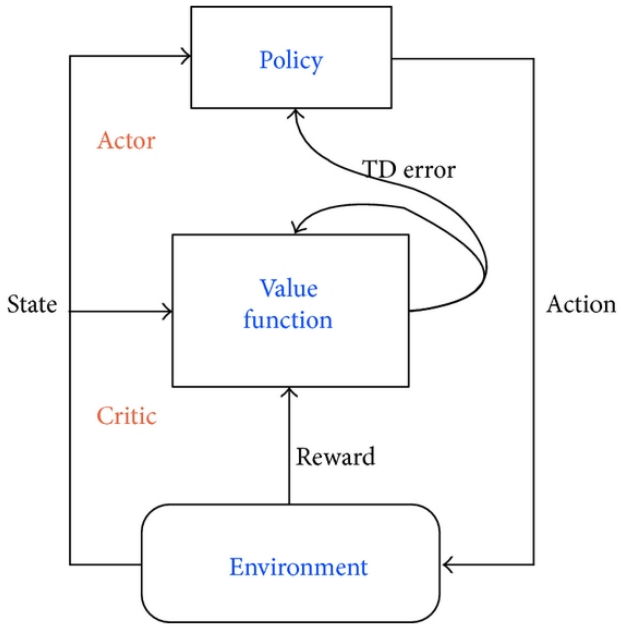


Figure 13: Actor-Critic Architecture [9]

2.3.3 Reinforcement Learning and Optimal Control

There exist several commonalities between reinforcement learning and optimal control and the shared elements between both disciplines will be outlined in this section. The design of a controller to minimize or maximize a measure of a dynamical system's behavior over time [7] can be addressed by RL and optimal control. Several methods have been developed to obtain optimal control to accomplish minimization of a system performance measure criterion. Two such methods are the use of Pontraygin's Minimization Principle and R.E. Bellman's dynamic programming (DP) [51]. The method of dynamic programming or exact DP, is described as a process that combines the decision-making involved in the principle of optimality with sequences of decisions. This forms a combined definition of an optimal policy and trajectory. The principle of optimality is a property of an optimal policy described by Bellman

as whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision [52]. This concept is mathematically described below as tail subproblems that starts at x_k^* at time k and minimizes over the optimal control sequence the "cost-to-go" from k to N

$$g_k(x_k^*, u_k) + \sum_{m=k+1}^{N-1} g_m(x_m, u_m) + g_N(x_N) \tag{61}$$

where $u_0^*, \cdots, u_{N-1}^*$ is an optimal control sequence belonging to set $U_0(x_0)$ with corresponding state sequence $x_1^*, \cdots, x_N^*$ [53]. The optimal control sequence is constructed with

$$u_0^* \in \arg \min_{u_0 \in U_0(x_0)} [g_0(x_0, u_0) + J_1^*(f_0(x_0, u_0))] \tag{62}$$

$$x_1^* = f_0(x_0, u_0^*) \tag{63}$$

and the sequentially increasing series of $k = 1, 2, \cdots, N-1$ to the final form of

$$u_k^* \in \arg \min_{u_k \in U_k(x_k^*)} \left[g_k(x_k^*, u_k) + J_{k+1}^*(f_k(x_k^*, u_k)) \right], \;\; x_{k+1}^* = f_k(x_K^*, u_k^*). \tag{64}$$

The generality of the x_k is not only a great strength of DP, but when combined with the tail subproblem that starts at x_k^* in the principle of optimality, the tail of an optimal sequence is optimal for the tail subproblem. This principle serves as the basis for the DP algorithm and allows for the backwards piecemeal construction of an optimal cost function. The DP algorithm can be summarized as sequentially solving all the tail subproblems of a given time length and using the solution of the tail subproblems of a shorter time length [53]. Further application of the DP algorithm to optimal control generates the following complete definition, where the DP algorithm

produces the optimal costs $J_k^*(x_K)$ of the x_k-tail subproblems

$$J_N^*(x_N) = g_N(x_N), \forall\, x_n \tag{65}$$

and for $k = 0, \cdots, N-1$ let

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} \left[g_k(x_k, u_k) + J_{k+1}^*(f_k(x_k, u_k)) \right], \forall\, x_k \tag{66}$$

The DP algorithm is sequentially constructing functions backwards at stage k, starting from J_N^*. For every initial state x_0, the optimal cost $J^*(x_0)$ is obtained at the last step of $J_k^*(x_k)$ and is equal to the number $J_0^*(x_0)$. Exact DP makes optimal decision in states and contains deterministic state transitions. This algorithm can be applied to a broad range of optimization problems ranging from deterministic to stochastic [54], combinatorial optimization and infinite state/control spaces optimal control [55], and neural network-based environments [56]. Two shortfalls of exact DP are the curse of dimensionality, which is the inflation of computation as the number of states increases, and the need for a mathematical model to fully describe the environment. Finding the exact solution to optimal control methods using DP is therefore often infeasible. In order to overcome these difficulties, suboptimal solution methods can be utilized and are known by several equivalent names such as approximate dynamic programming, neuro-dynamic programming, and reinforcement learning [53]. For the purposes of this study, the term reinforcement learning will be utilized and subsequently discussed as a solution to solving optimal control problems. As inferred by the equivalent name of approximate dynamic programming, RL uses an approximate cost denoted by $\tilde{J}$ instead of J^*. There are two types of approximation for this DP-based suboptimal control. The first is approximation in value space, where the optimal cost function of a given policy is approximated. The optimal control sequence

is transformed into the following form

$$\tilde{u}_0 \in arg \min_{u_0 \in U_0(x_0)} \left[g_0(x_0, u_0) + \tilde{J}_1(f_0(x_0, u_0)) \right] \tag{67}$$

$$\tilde{x}_1 = f_0(x_0, \tilde{u}_0) \tag{68}$$

$$\tilde{u}_k \in arg \min_{u_k \in U_k(\tilde{x}_k)} \left[g_k(\tilde{x}_k, u_k) + \tilde{J}_{k+1}(f_k(\tilde{x}_k, u_k)) \right], \quad \tilde{x}_{k+1} = f_k(\tilde{x}_K, \tilde{u}_k). \tag{69}$$

The second type of approximation is called approximation in policy space, where optimization is applied to select the policy over an appropriate family of policies. This parametric family of policies is

$$\mu_k(x_k, r_k), \quad k = 0, \cdots, N-1 \tag{70}$$

where r_k is a parameter and that can be estimated using an optimization method. These approximation methods have many variations as applied to a variety of analytical and computational problems and serve as the foundational description to how RL can be used to solve some, but not all DP problems. With a description of the common boundary between artificial intelligence and optimal control, specifically using reinforcement learning, the next subsection contains applications of these concepts to the general spacecraft attitude control problem.

2.3.4 Spacecraft Control Authority with Machine Learning

Novel machine learning controllers have been successfully implemented in a variety of mission scenarios such as robust hovering in the unknown dynamical environment of an asteroid [57], finding low-thrust trajectories in cis-lunar space using reinforcement learning [58], and learning guidance strategies instead of hand-crafted solutions [59]. For the purposes of this section, examples will be provided that use RL for spacecraft attitude control as either a controller or elsewhere inside the closed-loop

control system to directly impact the attitude control system.

Various examples and results of RL-based controllers for use in spacecraft attitude control have been published recently. Many of the following examples will utilize policy iteration methods to solve the general and continuous attitude control problem and apply various methodologies to train and optimize the RL agents. Researchers studying the use of a policy iteration method of Proximal Policy Optimization (PPO) [60] to satisfy the dual-objectives of obtaining a desired angular velocity and orientation at a desired time. It was determined that the inter-dependence of the angular velocity and rigid body attitude hinders solution convergence and to circumvent this, initial conditions were set close to the target states [61]. While such a learning-based method may be viewed as trivial, a strength to the RL agent is the ability to learn simpler problems first in order to solve complex problems later. The RL-based method yielded a 25 percent improvement versus a conventional controller method. Application of an RL Direct Policy Search algorithm (DPS) [62] for spacecraft maneuver near small bodies was implemented to accomplish inertial-frame hovering in low-gravity and unstable environments of small celestial objects [63]. Traditional control methods are shown to require full environment knowledge, which is procured with the use of many controlled trajectories around the body that are costly in both time and use of propellant. RL-based control was found to outperform a previously developed RL-based controller by Gaudet [64] and a another RL algorithm in terms of Steady State Mean Error (SSME) or spacecraft's average offset to the target hovering position. In a survey on artificial intelligence trends in spacecraft guidance dynamics and control, it was observed that success has been already achieved in the domains of orbital prediction, planetary landing, spacecraft guidance, and interplanetary trajectory optimisation and low-thrust propulsion [15]. The general attitude control problem is crucial to each domain type and the results of this research increase the development

and use of deep learning and RL-based methods in spacecraft systems.

2.4 Training and Software Optimization

The approximate cost term $\tilde{J}$ introduced in the previous section can be expanded to $\tilde{J}_k(x_k, r_k)$, which represents a class of functions called an approximation architecture. The process of choose the parameter vectors r_k is commonly defined as training or tuning the architecture [53]. Several training approaches can be applied to solve for the appropriate tunable parameters, and a common systematic approach through applied numerical optimization. As introduced in Section 2.3.1, feature vectors are involved within a supervised learning model architecture. This same term as applied to the parametric approximation of RL is expanded to define a type of cost approximation called feature extraction, where the state x_k is mapped into the feature vector $\phi_k(x_k)$. The scalar component features of the feature vector are based on human intelligence and features of an environment, often leading to complex nonlinear mappings. This drives the need for reducing the complexity into linear mappings. A good approximation to the cost-to-go that linearly weights the features coupled with a feature vector yields

$$\tilde{J}_k(x_k, r_k) = \hat{J}_k(\phi_k(x_k), r_k) = \sum_{\ell=1}^{m_k} r_{\ell,k}\phi_{\ell,k}(x_k) = r_k^{'}\phi_k(x_k) \tag{71}$$

where $r_{\ell,k}$ are weights, and $r_k^{'}\phi_k(x_k)$ is the inner product of r_k and $\phi_k(x_k)$ [53]. This linear feature-based architecture requires simpler training and parameter tuning and found within with neural networks and RL environments alike. ANNs are utilized within approximate DP problems and their properties are important considerations when training RL agents.

One important property of ANNs is universal approximation, which is described

as how any neural network with at least 1 hidden layer approximate any piecewise continuous function for inputs within a specific range and a sufficiently large number of nonlinear units [65]. The universal approximation property of ANNs and in turn RL environments does verify the adequacy of the neural network architecture but does not guarantee "good" performance. Increasing "good" performance depends on the training of linear and nonlinear architectures and several software tools integrated with languages such as Python and MATLAB exist to ease user-defined training processes. Training specific to the RL agent utilized in this research are discussed in Section 3.3 but with all RL methods that contain neural networks, training is conducted with the intent to change parameters to minimize some predefined loss function. A loss function records the error between the target and network-predicted output. One common type of loss function is least squares optimization, or least squares regression, where the generally nonconvex cost function of the training problem can be linearized to produce a closed-form solution. This process poses challenges in the field of RL as neither the network inputs nor correct outputs are given in advance and are instead obtained by agent-environment interaction [66]. A second type of function critical to neural networks are activation functions which describe the output of a selected node given certain inputs and allow access to a richer hypothesis space. Activation functions are nonlinear and by using smaller numbers of nodes paired with deep representations, complex problems can be computed [16]. Several types of activation functions are utilized in deep learning, with the most popular being $ReLU$, and descriptions of these can be found within [67].

Agent training for RL algorithms contains many tunable variables called hyperparameters designed to change the agent's ability to learn the policy and/or values of its environment. In addition to timesteps of the simulation, the agent interacts with the environment in batches of these timesteps in what are called episodes. The

episode details the agent's interaction from an initial state to a final state. The seed value of the simulation in a RL environmental context initializes randomization. This enables the generation of the same pattern of numbers per simulation run. As an RL agent trains, a common problem that arises is finding a balance between exploration (search the sample space in order to discover new features) and exploitation (using the found information during exploration) of the environment. Finding optimal actions depends on the using exploration to discover new paths to the goal and exploiting its learned knowledge [68]. One method of ensuring consistent exploration is to add adaptive noise to an agent's parameters in the form of exploration noise and policy noise. With the exploration size or the number of training steps to run, the exploration noise can lead to a more consistent exploration and bountiful set of behaviors. The policy noise also allows the agent to learn more readily from perturbations concerning the policy rather than the action space. Another hyperparameter that can have a large impact to the training efficiency is the batch size, which refers to the number of training samples used in one episode. A large batch size increases the accuracy of the gradient at a cost to increased memory requirements and computation time. A smaller batch size will require less memory and speed training. A final type of hyper parameter utilized by RL methods is called the discount factor. By determining how much importance immediate and future rewards are to the RL agent, the impact to the agent learning actions can be quantified. Initial values for hyperparameters can be determined from similar problem types and RL algorithms. These variables will be detailed with exact values in Chapter 3.1. With the provided attitude control, machine learning, and control theory, attention now turns to the research methodology. The SimSat simulator properties used by Doupe [2], specific RL architecture properties, and general mission description are outlined in Chapter III. The means of assessing RL inter-loop performance are also introduced.

III. Methodology

This chapter summarizes the environmental development, control architecture, and machine learning attributes of this research. The formulation of the methodology presented in this section serves to connect the concepts of the previous chapter with the experimental setup. A representative spacecraft and mission description are first defined, followed by the simulation and reinforcement learning schemes. An additional discussion topic is the evaluation of the reinforcement learning analytical results compared to traditional optimized solutions. These topics are formally summarized into two problem areas that will have their results presented in the concluding chapter of this study.

3.1 Spacecraft Mission Description

This section describes the simulated spacecraft environment and provides experimental setup details. As described in the previous chapter, this study references the simulations performed by Doupe [2] utilizing the AFIT SimSat and some details are reiterated here.

3.1.1 Mission Environment

The simulated spacecraft of this research are tasked with performing the primary objective of imaging ground targets with an RL commanded ADCS. This primary objective is expanded into two problem areas of interest, with the environmental details and parameters consistent between each problem area but differences noted in Section 3.4. The spacecraft is assumed to be in a circular orbit at an altitude of 770 km, constant ground velocity of 6.663 km/s, and maintains no greater than 30° for maximum off-nadir angle. The spacecraft has a body reference frame that is parallel

to a 2-D ground surface. Ground targets are assumed to be stationary in a 2-D flat field and the orbiting spacecraft must perform pre-determined attitudes corresponding to an on-board boresight or sensor instrument reading of a ground target. The coordinates of the desired ground target are (300, -250) km and these coordinates were chosen in order to require the spacecraft to perform large angle maneuvers for better performance evaluation of the RL controller. Known assumptions before the mission include where this target is and the required viewing dwell time. The representative target observation field mirrors the methodology from Doupe's research but only the first two targets are retained. A graphic of this environment is captured in Fig. 14, where the red star denotes the starting position of the spacecraft after completing a collection of that target (completed outside of this research's simulation) and the blue star the desired and next target.

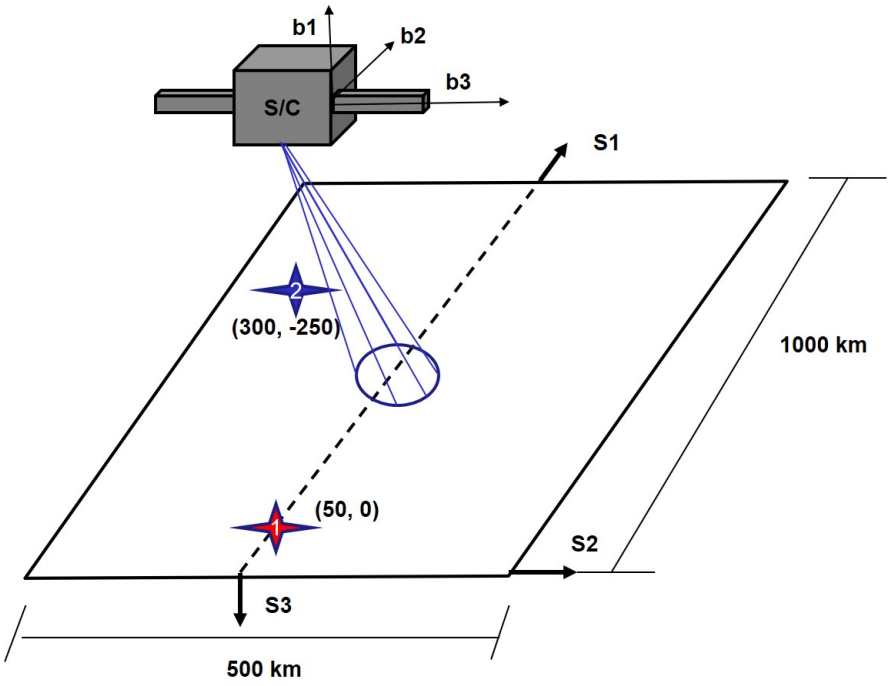


Figure 14: Graphic of target environment

In order to accomplish this mission, the primary objective of the spacecraft aided by a controller is to achieve a desired angular velocity and attitude simultaneously.

This objective is sought to be accomplished by the RL commanded ADCS, the details of which are contained within Section 3.3. This research seeks to compare an RL-based controller to traditional control methods and will utilize three different selections of CMG arrays that are tasked with slewing and precise attitude control. Such a control scheme comparison has not been previously investigated and tested and is an important aspect of this machine-learning based research. In order to evaluate the controller performance and overall attitude control authority, multiple actuator arrays are utilized to establish a family of attitude control capabilities for various mission types requiring agility. The general attitude control problem is challenging in spacecraft missions where capabilities change throughout the mission duration and this research seeks to evaluate the use of reinforcement learning within the control loop to solve optimal control problems. The environmental parameters of Section 3.1.2 are referenced throughout this chapter and the RL agent is tasked with interacting with these attributes.

3.1.2 Simulated Spacecraft Parameters

The properties shown in Table 1 and Eqs. (72) and (73), serve as the central simulated spacecraft to evaluate the RL agent performance, reiterated from Doupe's dissertation [2] and usage of SimSat. These values are later modified to those found in Appendix C for the simulations and are explained in Section 3.2.3. The SimSat spacecraft experiment platform is an AFIT owned and student designed hardware apparatus ([69], [70], [71]). From its original 1999 configuration, several iterations and upgrades have been performed. Its present configuration consists of a pyramid of four SGCMGs at a beta angle of 54.74°, three pairs of fans to simulate thrusters, an automated mass balance system, and three orthogonal reaction wheels. SimSat's has a CMG angular momentum storage of 0.45 N$\cdot m\cdot$s. This study seeks to utilize four

CMGs and starting with SimSat's SGCMG properties, will utilize three other ADCS configurations of CSCMG, VSCMG, and RWCMG. These first two configurations are meant to emulate an array of CMG actuators typically utilized by agile spacecraft [72], while VSCMG and RWCMG arrays are currently not implemented in hardware [2]. As described in Chapter II, CMGs conduct attitude control by tilting a spinning rotor to change angular momentum and cause a gyroscopic torque to rotate the spacecraft. These devices provide greater vehicle acceleration than a reaction wheel system of the same power. SimSat's physical parameters are:

$$\text{Spacecraft Inertia } (kg\text{-}m^2) = \begin{bmatrix} 6.454 & -0.197 & -0.175 \\ -0.197 & 9.716 & -0.142 \\ -0.175 & -0.142 & 12.848 \end{bmatrix} \tag{72}$$

$$\text{Single CMG Rotor Inertia} = \begin{bmatrix} 9.5E^{-4} & 0 & 0 \\ 0 & 9.5E^{-4} & 0 \\ 0 & 0 & 1.6E^{-3} \end{bmatrix} \tag{73}$$

Table 1: Spacecraft Initial States for CSCMG, VSCMG, and RWCMG arrays

State	Value
Quaternions ($\vec{q}$)	$[0;0;0;1]^T$
Angular Rate ($\vec{\omega}$)	$[0;0;0]^T$ rad/s
Gimbal Angles ($\vec{\delta}$)	([45°;-45°;45°;-45°]) or SQP values
Gimbal Rates ($\dot{\vec{\delta}}$)	0 rad/s each
Gimbal Accelerations ($\ddot{\vec{\delta}}$)	0.1 rad/s each
Rotor Rates ($\dot{\psi}_{CMG}$)	2600 rpm each
Rotor Accelerations ($\ddot{\psi}_{CMG}$)	0 rad/s^2 each
RW Rate ($\dot{\psi}_{RW}$)	0 rpm each
RW Acceleration ($\ddot{\psi}_{RW}$)	0 rad/s^2 each
Control vector (requested torque)	$\vec{u}$ = [0.0075;0.0061;-0.5492] N·m
CMG torque vector (commanded actuator torque)	$\dot{\vec{h}}_r$ = [-0.0075;-0.0061;0.5492] N·m

While SimSat serves as the reference spacecraft in a simulated environment, and

this research not performing hardware implementation, there is an additional spacecraft employed for purposes of this research. In order to understand performance range, the on-orbit imaging satellite of Pleiades High-Resolution (PHR) [11] was selected in addition to SimSat. This study is concerned with evaluating spacecraft control allocation of next-generation spacecraft that will require precise pointing accuracy and rotational agility. The PHR constellation of spacecraft is an example of the practical importance of designing spacecraft to achieve rapid retargeting with the use of CMGs. The actuator system of PHR employs a pyramidal cluster of four SGCMGs with a skew angle of 30°, each CMG with an angular momentum of 15 N-m-s to orient the 1000-kg spacecraft and accomplish a 60° roll (cross-track) slew maneuver in less than 25 seconds [11]. PHR has an approximately 800 kg-m^2 roll/pitch inertia. Situated at an altitude of 695 km, the CMG array of PHR has allowed for rapid slewing maneuvers while minimizing the micro-vibrations impacts to visual imagery collection. Table 2 contains the ADCS limitations of SimSat and Pleiades-HR for comparison, where each parameter corresponds to each actuator type. The values outlined in the tables of this subsection will be utilized in the simulation and among all three actuator configurations. Also, PHR gimbal acceleration limits were not found in literature and uses SimSat's value. The VSCMG rotor rates and accelerations for PHR will use SimSat's values for consistency. Because PHR does not employ RWs, the RW rates and accelerations of Sentinel-2 [73] will be used in order to conduct the RWCMG array simulation portion. The 4 RW array providing 18 N-m-s of angular momentum storage has a rate of 4000 rpm [74]. The RW acceleration was selected to be 136 rad/sec^2, mirroring the 1.5x magnitude increase between SimSat and Sentinel-2's RW rate values. While Sentinel-2 has a moderate slewing rate requirement of less than 0.5°, it is of a similar size, mission, and class of satellite to PHR. With the simulation parameters outlined, a background of the hardware

and software utilized for the simulation will be detailed next and followed by the simulation scheme.

Table 2: ADCS CMG Limitations of SimSat [2] and Pleiades-HR [11]

Parameter	SimSat	Pleiades-HR
Max Torque (N·m)	0.25	45 (avg: 20)
Angular Rate (deg/s)	8	4
RW Rate (rpm)	2650	N/A
RW Acceleration rad/s^2	10	N/A
Gimbal Angles (deg)	[-360°,-360°]	[-360°,-360°]
Gimbal Rates (rad/s)	2.5	3
Gimbal Acceleration (rad/s^2)	4.75	N/A

3.1.3 Hardware and Software

Simulation efforts were performed using a desktop computer with an Intel(R) Core™ 9700KF Processor, with seven cores at 3.60GHZ. The desktop uses a single graphics card in the form of a GeForce(R) GTX 1660 Ti 6GB GDDR6 (Turing) chip and has a x86_64 bit architecture with 16.0 GB of RAM. The desktop operating system was Windows 10 Pro with code written in the software programs of Python 3.8 and MATLAB. As an open-source interpreted, high-level and general-purpose programming language, Python is a powerful and useful tool in the realm of RL packages and environments. This study takes advantage of this attribute that would be less practical in other language libraries. Another language that was used in this study was MATLAB, a numerical computing environment that was used to conduct validation of Python generated results. MATLAB's matrix manipulation and symbolic computing abilities make it an ideal candidate for verifying the results generated from Python. The Python toolkit used for the RL algorithm implementation of this research is OpenAIGym [75], a toolkit for developing and comparing RL algorithms. The OpenAIGym library or Gym module provides a suite of RL algorithms that are compatible with existing computational libraries such as the TensorFlow [76] library

for machine learning and Theano [77], a library and optimizing compiler for data manipulation. While the Gym module serves to create the RL environment, the open source machine learning library PyTorch [78] is used to formulate the deep neural networks in the RL architecture. One of PyTorch's main advantages is accelerated tensor computation with the use of graphical processing units (GPU), which is an important tool in this research used to speed simulation run times. As alluded to, several machine learning libraries and frameworks exist for a variety of applications such as computer vision, natural language processing, and deep learning. A compiled list of deep reinforcement learning libraries to include the ones in this study can be found at the following Github location [79] for more information.

3.2 Simulation Model Closed-Loop Control Scheme

With outlined spacecraft mission environmental parameters, this section describes the general closed-loop scheme integrated with the RL controller. The diagram of Fig. 15 will serve as a reference for the following subsections, where relationship signal flow is quantified by arrows. This diagram contains the items used to describe the process flow of the control loop. The initialization section contains the simulation parameters and limits, passing this information to the calculation portion of the dynamic equations which are then utilized by the RL controller and environment. The RL control block is described in Section 3.3.1. For simplicity, the applicable simulation parameters introduced in Section 3.1.2 and defined in the initialization block have been visually omitted from the figure. The following subsection expands on these applicable parameters by integrating them with the RL environment.

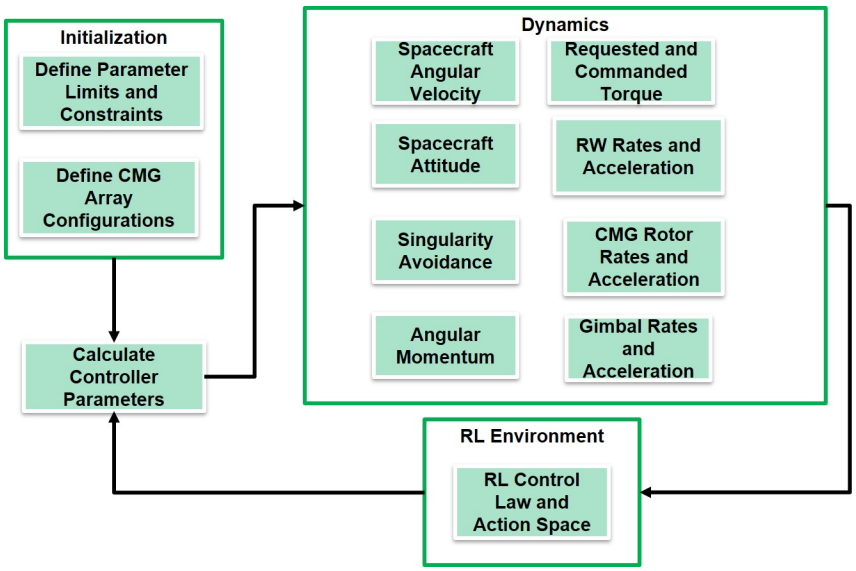


Figure 15: General Closed-Loop Control Scheme

3.2.1 Observation and Action Spaces

The observation space (O) of the simulation contains the states the RL agent will use when structuring the policy function. It must know information about its environment before it can begin to learn and the type of provided observations dictates which type of policy can be implemented. The observation space contains the environment states. From the simulation description and parameters, the observation space will contain continuous values about the spacecraft and the ADCS system and these values are captured in Eq. (74) for SimSat and PHR. The initial values for these states for both spacecraft and all CMG arrays are in tables in Appendix C, the initial time starts from zero seconds, and the angular momentum calculated for each array conducted in Section 2.1. The initial values found in Appendix C are the values the environment is reset to per episode and were generated from the baseline simulation. The angular momentum is constant for the CSCMG as seen by Eq. (16). The VSCMG array has the additional states of the CMG rotor rate ($\dot{\psi}_{CMG}$) and acceleration ($\ddot{\psi}_{CMG}$). The RWCMG array has the additional states of RW rate ($\dot{\psi}_{RW}$)

and acceleration ($\ddot{\psi}_{RW}$). The simulation run time and time step are held constant between each spacecraft and ADCS configuration.

$$\mathrm{O} = \begin{bmatrix} \text{Quaternion } (q) \\ \text{Angular Velocity } (\omega) \\ \text{Gimbal Angle } (\delta) \\ \text{Gimbal Rate } (\dot{\delta}) \\ \text{Gimbal Acceleration } (\ddot{\delta}) \\ \text{Controller Torque } (u) \\ \text{CMG Torque } (\dot{h}_r) \\ \text{Singularity Measure } (m) \\ \text{Current time } (t) \\ \text{Sim time step } (dt) \\ \text{Angular momentum } (h) \end{bmatrix} \tag{74}$$

It was determined providing numerous states was more advantageous to the agent's ability to converge and required computation time. The spacecraft specific minimum and maximum values of these completely observable states are contained within Tables 4 and 8 of Section 3.4. These values are based on the hardware limitations of both spacecraft and the designer and user of simulation has the option to change these values. Some values are restated in Tables 5, 6, 9 and 10 from earlier discussions. The vector components for each parameter type have the same minimum and maximum value. For example, any component of the quaternion unit vector ranges from -1 to 1, inclusive. OpenAIGym environments have 6 different types of spaces and the Box space was utilized to contain the bounded and continuous state values. The action space consists of all actions the agent can perform in the environment. It can be considered the control exacted by the agent to change the observations or

states to bring about desired goal conditions. The action space of this simulation is continuous but discretized by the Box space of OpenAIGym environments. The agent is responsible for controlling the torque input to the CMG arrays. The initial gimbal angles for each simulation run are predefined as the alternating Vadali look up values. This initial gimbal angle selection approach differs from the preferred gimbal angle optimization methodology found in Doupe's research [2] and this study seeks to determine the ability of the agent in having such environmental interaction.

3.2.2 Dynamic Equations

The initialization section of the control diagram describes calculation of all parameters and associated limits and constraints inside the closed-loop. The observed parameters of this system are contained within the initialization and dynamics sections and are utilized by the RL environment. The spacecraft mission is categorized by the use of three ADCS configurations outlined in the previous section and for organizational purposes, these arrays will be outlined in a similar order here. The parameters utilized by the RL controller first depend on the calculation of the Euler angles $\vec{\theta}_c$ and quaternion attitude characterizations. These commanded or desired parameters are determined per time step to orient the spacecraft at the desired target but for the purposes of this research will reflect a single time step. The commanded 3-2-1 Euler angles can be found from a function of simulation time, target coordinates, and simulation time [2] and the desired attitude and angular rate were selected 17 seconds into the simulation. The desired angles and rate for the intended target are presented in Eq. (75) and (76).

$$\theta_{c1} = [0,\ 15,\ 15.944]^\circ \tag{75}$$

$$\vec{\omega} = [0.0013,\ 6.6038\text{E}{-}05,\ -0.0049]^\circ/sec \tag{76}$$

The spacecraft is initially commanded from rest $\vec{q} = [0, 0, 0, 1]$ and $\vec{\omega} = \vec{0}$ and no perturbation or gravitational torques are applied. The spacecraft is not required to accomplish a rest maneuver after accomplishing the objectives but this can be programmed if so desired. The goal of achieving the commanded attitude and desired velocity can be encapsulated by the following cost functional

$$min \;\; J = \int_{t_0}^{t_f} -m + \alpha_1 \vec{q_e} + \alpha_2 \vec{\omega_e} \;\; dt \tag{77}$$

where the RL agent is tasked with minimizing the time spent to reach these desired states and maximizing the singularity measure m, denoted with a negative sign. Weights are attached to the attitude and angular weight using the variable α_i. This cost functional was not implemented in the simulation environment and instead serves as an optimal control analogy. Minimizing the error is captured in the second and third terms as limiting the absolute error of the quaternion attitude ($\dot{\vec{q}_e}$) and angular velocity ($\vec{\omega_e}$). The units of all three cost function components are dimensionless. The RL agent has a finite amount of time to perform the slewing maneuvers associated with each target. While the cost functional of Eq. (77) can be discretized, it is assumed that for each rest-to-rest maneuver performed per target has an individual cost. The costs will not be summed into a cumulative value. The cost functional is subject to the dynamics of

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, t) \tag{78}$$

where

$$\boldsymbol{x} = \begin{bmatrix} q \\ \omega \\ m \end{bmatrix}. \tag{79}$$

To avoid confusion and provide clarification, the observation space of the previous subsection contains all states the RL environment must know in order to solve the optimization problem. The states of interest to said problem are contained in Eq. (75). The control $\boldsymbol{u}(t)$ are the gimbal angles δ and controller torque calculated from Eq. (37). The quaternion attitude, singularity measure, and angular rate are constrained by the initial state $\boldsymbol{x}(t_0) = [q(t_0), \omega(t_0), m(t_0)]^T$, where $q(t_0) = [0, 0, 0, 1]^T$, $\omega(t_0) = [0, 0, 0]^T rad/s$, and $m(t_0) = 0.3$. The singularity measure cannot have an initial value of zero as this equates to the CMG array starting in an undesirable singular state and the value of 0.3 is similar to low-range values from [2]. The path constraints, state, and control bounds are provided by the Observation and Action space Section 3.2.2. The spacecraft can only follow a solution trajectory within ranges specified in this section. The RL controller does not require conditioning with the use of limit enforcement algorithms but controller saturation is conducted by clipping the controller values. Tables 5 and 9 for each array type contains the spacecraft specific values. These values are taken from the limit enforcement section saturation function calculations conducted by Doupe. With the array equations and concepts outlined, attention turns to the required equations for propagating values such as the gimbal rates and rotor accelerations per evaluated time step.

The propagation methods used in the simulation are Euler's method and the Runge-Kutta method. These methods are needed in order to propagate the states of the environment over several timesteps by solving ordinary differential equations given initial times. The first-order Euler method is a numerical integration procedure that approximates the actual $y(t)$ function curve using local linearity and joining of multiple line segments [10]. This local linearity defines local error, or error per step, as proportional to the square of the step size while the global error is proportional to the step size [10]. The Euler method algorithm is illustrated in Fig. 16.

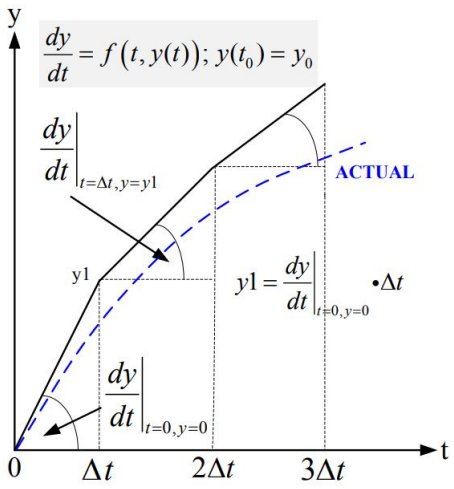


Figure 16: Euler's Method Algorithm [10]

The fourth-order Runge-Kutta (RK4) integration method is used to solve the numerical ordinary differential equation $y'(x) = f(x, y)$ with initial condition $y(w_0) = y_0$. It is the most widely known method of the Runge-Kutta family. The classical representation of this algorithm is shown in Eq. (80), where K_1, K_2, K_3 and K_4 represent slopes used to generate a weighted average of increments taken throughout the approximation process and h is the step size [80].

$$w_{k+1} = w_k + \frac{1}{6}(K_1 + K_2 + K_3 + K_4)h \tag{80}$$

$$K_1 = f(x_k, w_k) \tag{81}$$

$$K_2 = f(x_k + \frac{1}{2}h, w_k + \frac{1}{2}K_1 h) \tag{82}$$

$$K_3 = f(x_k + \frac{1}{2}h, w_k + \frac{1}{2}K_2 h) \tag{83}$$

$$K_4 = f(x_k + h, w_k + K_3 h) \tag{84}$$

The dynamic equations contained within $f(\boldsymbol{x}, \boldsymbol{u}, t)$ will now be described with

respect to the methods of integration employed by this study. The vectors of

$$\boldsymbol{x}_1 = \left[q, \omega, \delta, \dot{\delta}\right]^T, \quad \boldsymbol{x}_2 = \left[q, \omega, \psi, \dot{\psi}, \delta, \dot{\delta}\right]^T \tag{85}$$

contains the states integrated by the simulation and used by the RL environment to update all state information. The vector $\boldsymbol{x}_1$ corresponds to the CSCMG array and $\boldsymbol{x}_2$ the VSCMG and RWCMG arrays. The Euler and RK4 implementation equations for the CSCMG array is in Eq. (86) through (91), and the VSCMG and RWCMG arrays additionally using Eq. (92) and (93). The $[^\times]$ operation refers to a skew symmetric matrix. The quaternion is normalized before and after the integration process to ensure the unit norm constraint is satisfied. These propagation methods are also utilized in the validation simulation of Section 3.3, albeit RK4 with a variable step size.

$$\dot{q}_i = \frac{1}{2}\begin{bmatrix}[\bar{q}^\times] + q_4 J \\ -\bar{q}^T\end{bmatrix}\vec{\omega} \tag{86}$$

$$\dot{\vec{\omega}}_i = J^{-1}(\tau - [\vec{\omega}^\times]J\vec{\omega}) \tag{87}$$

$$q_f = q_i + \dot{q}_i dt \tag{88}$$

$$\omega_f = \omega_i + \dot{\omega}_i dt \tag{89}$$

$$\delta_f = \delta_i + \dot{\delta}_i dt \tag{90}$$

$$\dot{\delta}_f = \dot{\delta}_i + \ddot{\delta}_i dt \tag{91}$$

$$\psi_f = \psi_i + \dot{\psi}_i dt \tag{92}$$

$$\dot{\psi}_f = \dot{\psi}_i + \ddot{\psi}_i dt \tag{93}$$

3.2.3 Singularity Avoidance

As described in Section 2.2, singularities can occur when the gimbal orientation of a CMG array produces parallel torque vectors, leading to undesirable torque outputs. Several steering laws and logic were next outlined to illustrate the methods to avoid and/or escape such situations. The purpose of this section is to both describe the steering laws and proposed methodology for singularity avoidance analysis for the RL-controlled system. The methodology of this subsection is driven by the following outline of singularities in Fig. 6, where despite the configuration type all CMG arrays have the possibility of encountering singularities [4]. The singularity robust inverse was utilized as the singularity escape algorithm for the simulation along with singularity measure of Eq. (96). The generalized from taken from [2] and [3] is

$$\dot{\vec{\delta}} = A^T(AA^T + \lambda I)^{-1}\dot{\vec{h}}_{act} \tag{94}$$

$$\lambda = \begin{cases} \lambda_0(1 - m_i/m_0)^2, & for\ m_i < m_0 \\ 0, & for\ m_i \geq m_0 \end{cases} \tag{95}$$

$$m_i = \sqrt{\det(AA^T)} \tag{96}$$

The constants λ_0 and m_0 were 0.05 each and adjustable parameters. The value for m_i approaches zero when singularities occur. To force the gimbal rate to steer around a singularity, the piece-wise relationship of λ either has λ be zero or nonzero when m_i drops below the desired m_0 threshold. Adjusting this λ_0 parameter will reduce the unwanted amount of torque error that results from the singularity avoidance. With the usage of a RWCMG array, a weighting matrix W is introduced that is derived from the Schaub weighted pseudo-inverse of Eq. (14). The total gimbal rates and reaction wheel accelerations will determine the dimensions of this diagonal matrix and in this case is 7×7. This matrix is used as a switch between the active actuator

set being used to steer the spacecraft. The initial weighting matrices for the CSCMG, VSCMG, and RWCMG were as follows

$$W_{CSCMG} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{97}$$

$$W_{VSCMG} = \begin{bmatrix} 0.099 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.099 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.099 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.099 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{98}$$

$$W_{RWCMG} = \begin{bmatrix} 0.099 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.099 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.099 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{99}$$

3.2.4 Target Collection

The simulation description is further elaborated in this subsection. With the target deck representation of Fig. 5, the spacecraft is tasked with conducting a slew and collection maneuver for a singular target. The dwell time on each target bounds the

operation space of the RL agent to achieve the desired attitude, but the agent is not tasked with conducting the slew in minimum time. Such an optimization problem is beyond the scope of this research and would entail an alternative problem formulation. To simplify this observation space, it was determined that the RL agent must achieve the desired attitude in a maximum amount of time, where the collection time is incorporated into the total time amount. These time allocations were determined from the simulation run times per CMG array as in [2]. Per time step the agent spends outside of this given time, less rewards are given. The simulation begins with a slew period of time that allows the controller to orient the spacecraft. The required orientation for the spacecraft for the collection period is a difference between the commanded (θ_c) and measured (θ_m) Euler angles to be less than or equal to 2°. This difference is captured in a three-axis pointing error defined in [2] and provided in Eq. (103) , where N is the number of time steps

$$\epsilon_1(k) = |\theta_{1c}(k) - \theta_{1m}(k)| \tag{100}$$

$$\epsilon_2(k) = |\theta_{2c}(k) - \theta_{2m}(k)| \tag{101}$$

$$\epsilon_3(k) = |\theta_{3c}(k) - \theta_{3m}(k)| \tag{102}$$

$$\epsilon = \sum_{k=1}^{N_{ts}} |||[\epsilon_1(k)\epsilon_2(k)\epsilon_3(k)]||_2 \tag{103}$$

Before the collection, a check is performed to examine if the spacecraft is in collect mode and if all targets have been collected already. Once the collection begins, the spacecraft has the given amount of time per target as shown in Table 2 to collect during its maximum time limit. The spacecraft must be stabilized and have limited angular velocity during this mode. The initial or preferred gimbal angles are set to the Vadali angles [81] of [45°, -45°, 45°, -45°] for each gimbal for the first target and the RL controller is responsible for determining these angles for each subsequent target.

3.3 Reinforcement Learning Control Structure

The general simulation model closed-loop control scheme of the previous section contains an deep RL controller and environment. The RL methodology outlined in this section will cover the RL environment and model, observation and action spaces as to how they relate to applicable parameters of Section 3.2.2, and the model evaluation and training processes.

3.3.1 Implementation and Model

This section details each component of the RL control loop as it functions inside of the general control scheme outlined in Section 3.2. The OpenAI Gym module and PyTorch libraries were utilized software packages in this implementation and have specified configurations. This modular environmental structure can be seen in Fig. 17. The structure is characterized by a driver file that serves as the main file for the simulation. The arrows flow from the direction of the originating block to the recieving block. Driver_File.py conducts the policy training and evaluation once the TD3 agent and replay buffer have been generated. The specifics of the TD3 algorithms is provided in Section 3.3.2 and this learning algorithm utilizes two actor-critic networks. The use of such deep neural networks means deep RL is used in this study to conduct robust nonlinear function approximation. The driver file also imports the environment that includes all equation calculations and integration schemes from Calc_Params.py and reward function inputs from Reward.py. As the agent learns the policy, state data is output to the Plotting.py function for user interpretation. Finally, the best model is saved from the driver file for later usage and to skip time required for re-training.

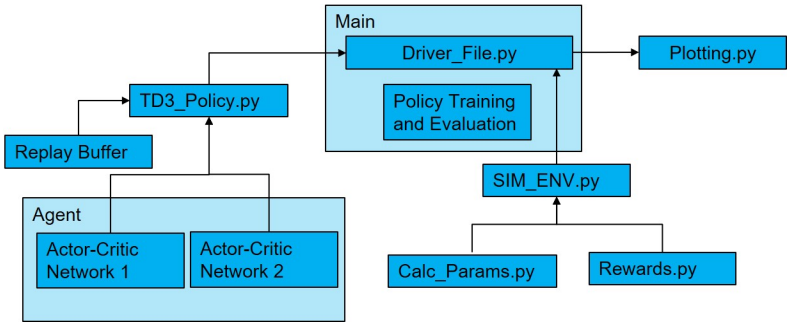


Figure 17: RL Program Structure

3.3.2 Policy Description

The RL algorithm of choice of this research is the state-of-the-art and successor to the Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG (TD3) [82]. This algorithm was chosen to not only handle the continuous observation and action spaces of this study's problem, but to capitalize on its features of policy gradient and deep Q-learning to ensure ideal performance. The sample efficiency of this algorithm and subsequent required fewer interactions with the environment made it an ideal for this problem and it has been demonstrated to perform more effectively than other popular algorithms for continuous control tasks. To the author's knowledge, implementation of this type of algorithm to the general spacecraft attitude control problem with CMG arrays has not been conducted but TD3 falls within the types of algorithms utilized for this type of problem such as PPO and DDPG. As an off-policy algorithm, TD3 trains a deterministic policy that benefits from added mean-zero Gaussian noise to actions during the training period. It was found with DDPG that policy breaking occurred with the dramatic overestimation of Q-values and thus TD3 was proposed as a way to increase the stability and performance with consideration of function approximation error [82].

TD3 addresses this this policy breaking in three ways. First, TD3 learns two Q-functions and uses the smaller of the two Q-values to form the targets in the Bellman

error loss functions, also called mean square Bellman error minimization. The target Q-value is a clipped value that helps to prevent overestimation and an actor-critic network maintain these Q-values. The second feature of TD3 are the delayed policy updates where the original formulation of TD3 [82] describes one update policy for every two Q- function updates. This delay allows for more time for value estimate convergence by limiting the likelihood of repeating updates. The last feature of TD3 that improves DDPG's performance is target policy smoothing. The value function has a smoothing regularization strategy to enforce similar actions having similar values. This is feature is accomplished by adding small amounts of clipped random noises to the selected action and averaging over several small batches. The pseudo-code for this algorithm is found in Appendix A [8]. The variable of c is a selected value clip for the noise parameter, ϵ.

With TD3 defined, the algorithm architecture for the CSCMG and RWCMG arrays are detailed in Table 3 and is derived from the research conducted on several continuous control environments of OpenAI Gym [83]. The architecture for the VSCMG array was found to function with greatly reduced state oscillations the values of Table 3 decreased by a factor of 10. Additional hidden layers were added from the environment of [83], the number of neurons was decreased to improve performance, and the activation function (or node output) was changed to leaky ReLU (LReLU). The leaky ReLU function was found to have better performance compared to ReLU and by sustaining and maintaining weights throughout the propagation process, takes a more balanced approach and avoids zero gradient values that occur using ReLU [67]. It was found that each CMG array functioned with the same neural network architecture with varied hyperparameters.

Table 3: Actor and critic network architectures and activation functions

	Actor/Policy Network		Value/Critic Network	
	# Neurons	Activation Function	# Neurons	Activation Function
Input Layer 1	(# states)	LReLU	(# states)	LReLU
Hidden Layer 1	200	LReLU	150	LReLU
Hidden Layer 2	150	LReLU	200	LReLU
Hidden Layer 3	200	LReLU	150	LReLU
Output Layer 3	(# actions)	Tanh	(# actions)	None

3.3.3 Reward Function Description

For the problems and selected policy presented in this research, the reward function is a combination of the finite-horizon discounted return and infinite-horizon discounted return. Several deep RL algorithms utilize combinations of returns for optimization purposes and TD3 falls within this category. The undiscounted return is optimized by TD3 but additionally uses discount factors to estimate the $Q^*(s, a)$ value function. With the reward function background provided, attention turns to the timestep reward, r_t. The designer must provide rewards to the agent in such a manner that in maximizing them, the agent accomplishes the desired goals. To the author's knowledge, there are no precise rules to follow when choosing these functions, but should properly indicate what the agent should accomplish and not concretely how to achieve the goal. As stated in Section 3.1.1, the RL agent is tasked with accomplishing a desired attitude and spacecraft body angular velocity corresponding to a ground target in a finite amount of time. The reward function formulations of this subsection were applied to each CMG array and spacecraft configuration unless noted otherwise. The quaternions and spacecraft rate are tracked states in addition to parameters defined in Eq. (75) and serve to define the first formulations of the timestep reward. If the agent is time constrained, speed is an important aspect of goal accomplishment and the agent should be penalized per additional timestep taken outside the upper boundary. Another important aspect and key element of this research is the performance of singularity avoidance. The agent must either incur a heavy

penalty when the singularity measure drops below a lower boundary, thus reaching a singularity or receives a large positive reward for maximizing the singularity measure. It was also determined the agent should be responsible for ensuring conservation of angular momentum. The last elements of the timestep reward involve the constraints on the ADCS components and adhering to the limitations of the gimbal rates and acceleration, CMG rotor rates and acceleration, and RW rotor rates and acceleration outlined in Table 2.

With these aspects in mind, the time incentive or reward for the RL agent was initially set to have the episode terminate when the maximum time limit was violated. In order to have the agent accomplish the main goals of its attitude and angular velocity along with maximizing its singularity measure, a reward was first given to encourage the agent to explore the space. Another reward was given to drive the differences between the current and previous timestep values of the quaternion and angular velocity vectors to zero. The final reward signal contribution was the rewarding the agent if the singularity measure remained above a 0.03 threshold. The values for these three rewards were 0.5, 0.5, and 0.2 respectively, placing greater emphasis and thus reward value with the main goals. The total reward function r_t for the initial reward function formulation is detailed in Eq. (107).

$$r_1 = \Delta q_{prev} - \Delta q_{current} \tag{104}$$

$$r_2 = \Delta \omega_{prev} - \Delta \omega_{current} \tag{105}$$

$$r_3 = \text{Above } m_{threshold} \tag{106}$$

$$r_t = r_1 + r_2 + r_3 \tag{107}$$

The limit and constraint enforcement for the simulation is performed entirely within the RL environment. As continuous spaces, the observations and actions have set

ranges that serve to bound the agent search space and emulate hardware limitations of the spacecraft and ADCS. These values are detailed in Section 3.2.3. To further enforce these limits, some constraints were incorporated as penalties inside the reward function. If the maximum limits for the gimbal rates, gimbal accelerations, angular rates, and singularity measure the agent received a penalty of -5 per timestep spent outside these ranges. For this initial reward formulation, it was discovered that the agent took an undesired amount of time to reach a terminal state, hardware constraints of most of the observation space were violated, and final policy evaluation was not possible. To limit the search time, episode termination was incorporated into the constraint violations by having the agent stop searching when values are produced that are not physically realizable. The quaternion and angular velocity errors were also considered instead of time step differences and were substituted in place of the differences with negligible performance improvement.

From these discoveries, the next attempt to improve the performance of the agent was to focus on limiting the search space and eliminating state limit violations. The extreme oscillatory attitude behavior during the collection phase of the spacecraft indicated the spacecraft was not still during its maneuver. It was determined that in order to incentivize the agent to not to reach the goal at undesired angular rates was with the use of a decay function and gradient. As the agent reached the goal, it would be progressively rewarded. Equations (108) and (109) are functions designed to increase the amount of reward for smaller values for taking the two-norm of the difference between the commanded and actual Euler angles Δ_{EA} and the angular rate magnitude. Values of the pointing error Δ_{EA} above the target collection threshold were given double importance compared to the angular rate magnitude to ensure the agent accomplished the attitude. It was presumed that by ensuring the desired attitude was accomplished, other performance attributes would in turn be within desired

values. The hardware constraint violations were therefore omitted. The angular rate differences calculation was retained to further reinforce the need for the spacecraft to have minimal rotation during the collection process. Equation (111) is a piecewise function describing an exponential relationship for the scalar portion of the quaternion error vector that is dependant on Δ_{EA} and is modified from [84]. Larger rewards are provided to the agent if the current time step error value is larger than the previous timestep's and in turn minimizing Δ_{EA} to zero. The scalar portion of the error quaternion is desired to be close to 1. Values in these reward functions were selected to avoid singularities.

$$r_1 = 2(1 - \frac{\Delta_{EA}}{Tgt_{threshold}})^{0.5} \tag{108}$$

$$r_2 = (1 - \frac{|\vec{\omega}|}{|\vec{\omega}_{max}|})^{0.5} \tag{109}$$

$$r_3 = \Delta\omega_{prev} - \Delta\omega_{current} \tag{110}$$

$$r_4 = \begin{cases} e^{\frac{-\Delta_{EA}}{2\pi}}, & for\ q_{e,t} > q_{e,t-1} \\ e^{\frac{-\Delta_{EA}}{2\pi}} - 1, & otherwise \end{cases} \tag{111}$$

$$r_t = r_1 + r_2 + r_3 + r_4 \tag{112}$$

The reward of Eq. (112) was found to eliminate the agent wandering away from the goal and violating any state violations. Appropriate episode termination was conducted and final evaluation revealed the ability of the spacecraft to be controlled to the desired attitude. Remnant undesired behavior in the form of long search space times, short attitude orientation maintenance, and infeasible angular velocity magnitudes drove the need to further improve the performance.

The results from this initial experimentation drove the divergence of the reward functions for each array. The final formulations will be described next. The CSCMG array displayed the best ability out of the arrays to converge to a solution. This may

have been due in part to its smallest level of complexity compared to the two other arrays and ability to handle larger simulation step sizes. In order to determine how to best bound the search space for the array, constraints were levied on the agent in the form of episode terminations, negative rewards, and canted rewards. Initial reward function formulations took the form of Eq. (108) through (111) taken from similar spacecraft attitude control studies and used a multiplicative reward function instead of the addition portrayed in Eq. (112). Agent training revealed an inability of the agent to converge to a solution or reach the 2 degree target threshold for collection despite the gradient approach for the reward formulation. Using the Euler angle error as the deciding factor for guiding the agent to the goal was an inadequate variable and quaternion and angular velocity error was added and produced greater success in the agent recognizing the need to decrease the Euler angle error. The time step for the array was 0.08 seconds. It was discovered that conducting episode termination for any state minimum and maximum violations was an unessential aspect for training and caused the agent to prematurely exit an episode and not maximize learning. A negative penalty value of -10 coupled with continuing an episode despite state constraint violations proved to be a more effective way of encouraging effective learning.

The primary challenge presented during agent training is the conveyance of solution uniqueness and an understanding of the relationship between the control and the environmental states. Providing an Euler angle, quaternion, and angular velocity error did not guarantee the agent's understanding in the significance of a presented solution, therefore causing large exploration. Additional details were added to best describe the solution and included penalizing the agent by -10 for generating solution components of the Euler angles that were not of the same sign as the solution, having a singularity measure lower than the near singularity measure threshold of 0.03, and rewarding by 100 for having further decreases in the Euler angle error towards the 2

degree target threshold. The singularity measure was the only state to carry with it early episode termination due to how a zero singularity measure equates to no producible torque by the system and subsequent non-functionality of the controller. A weighting factor to increase importance of a reward function aspect was captured by the variable α that equaled 10^3. The reward components of describing the exponential calculations pertaining the Euler angle error, angular rate error, and quaternion error share the same dimensionless units. A large positive reward of 1000 was provided to the agent when the 2 degree threshold and below was reached. The final reward function for the CSCMG array is described in Alg. 2 in Appendix D, where the Euler angle importance is captured in two separate reward functions and Alg. 2 called per simulation execution.

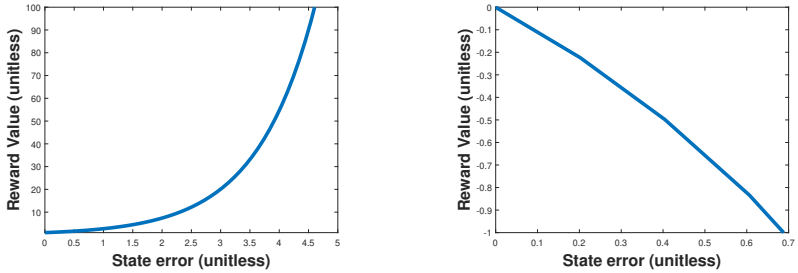
The increased steering logic complexity of the VSCMG array caused a large difference in agent training ability compared to the CSCMG array, requiring a modified reward function formulation. The simulation step size was maintained to allow for the required level of accuracy from the integration method. Severe oscillatory state profiles and near singularity cases was remedied using negative rewards for each state and penalizing the upper limit of the singularity measure. Due to the singularity measure consistently below the 0.03 threshold, this upper limit formulation was found to allow the agent to continue to search the solution space and eventually learn to maximize this value. The exponential reward components for the Euler angle error, angular error, and quaternion error was discovered to cause desired behavior in the VSCMG array and was maintained. Constraint checks for the CMG rotor rate and acceleration were added. The largest modification to the CSCMG reward function for the VSCMG array was the separation of the constraint checks and conditional check for the decreasing trend of the Euler angle error. It was determined that prior to separation, the agent was unable to glean the relationship between the positive

reward of 5 generated when decreasing the Euler angle error and receiving negative rewards from the constraint violations. The constraint violation penalties of -10 was a frequent occurrence until this separation was conducted. The VSCMG array reward formulation is described in Alg. 3 in Appendix D.

The RWCMG array proved to be the most difficult model for conducting agent training and a new methodology was applied to properly bound the search space. Undesired behavior exemplified by a negative total and average reward plots and insufficient solution space searching required a large degree of reward function experimentation. The constraints placed on the states as found the reward functions of the CSCMG and VSCMG arrays were found to produce severe detriments to the ability of the agent to search and produce enough control. The RWCMG reward formulation subsequently does not contain any rewards associated with the states with the exception of bounding the singularity measure between lower and upper boundaries. The threshold of individual Euler angle error components was decreased to further bound the space. A total reward lower and upper threshold, positive singularity measure exponential function, and exponential applied to the states are examples of implemented methods to improve the performance but proved unsuccessful. The RWCMG array reward formulation is described in Alg. 4 in Appendix D.

Among all three CMG array reward function compositions, a gradient-based reward for the Euler angle, quaternion, and singularity measure qualitatively represented in Fig. 18 was found to be an effective method of guiding the agent to the solution and increase the quality (or knowledge gained) of each episode. Less negative reward was obtained if the agent decreased the Euler angle and angular rate and more positive reward for increasing the quaternion scalar component towards a maximum value of 1. The values of the x and y axis of the plots serve to exemplify and illustrate the behavior given a desired state error value. Modifying the initial or

reset values for each CMG array also improved the agent's ability to train and these values are in Appendix C. Other consistent details among all three reward functions include a mixture of positive and negative rewards, with more emphasis placed upon decreasing the Euler angle error, and verifying solution uniqueness. The results of the agent training and ancillary analysis are detailed in Chapter IV.



(a) Positive Trend (e^x) (b) Negative Trend (1-e^x)

Figure 18: Exponential reward function behavior illustrated

3.4 Problem Formulation

The entirety of the methodology section up until this section encompasses the concepts and parameter space of this study. The efforts of this research are condensed into two problem areas of study that seek to comprehensively answer the research objectives and questions introduced in Chapter 1. The training hyper-parameters and observation space values specific each spacecraft's CMG array simulations are found in their individual portions of this section.

3.4.1 Problem A Methodology

Problem A examines the application of RL to the general spacecraft attitude control problem by investigating whether results produced by the RL controlled simulation of this study are comparable to the traditional control methods utilized by Doupe. This problem consists of conducting sequential slews with the SimSat hard-

ware parameters from Table 2 and analyzing the resulting performance. The state or observation space of this problem includes the values found in Eq. (75) and state constraints in Table 4, where boundaries were determined from minimum and maximum values from the baseline simulation. The action space consists of the gimbal angles ranging from $[10, -70, 10-70]^\circ$ to $[120, 50, 120, 50]^\circ$ and the requested control torque range is ± 1 N·m. For agent training, Table 6 contains the hyper-parameters used for the SimSat simulations. The seed for the simulation is zero due to its non-use by any portion of the spacecraft environment.

Table 4: SimSat Environment State Bounds

Parameter	Minimum	Maximum
q1	-0.0249	0.0440
q2	5×10^{-10}	0.1340
q3	-0.3341	0.1647
q4	0.9339	1
ω ($^\circ$/sec)	-4	4
δ ($^\circ$)	-70	70
$\dot{\delta}$ ($^\circ$/sec)	-57	-57
$\ddot{\delta}$ ($^\circ$/sec^2)	-272	272
$\dot{\psi}_{CMG}$ (rpm)	2600	2600
$\ddot{\psi}_{CMG}$ (rpm)	-1000	1000
$\dot{\psi}_{RW}$ (rpm)	-2650	2650
$\ddot{\psi}_{RW}$ (rpm)	-90	90
u (N·m)	-1	1
m	0.3	1.3
t (sec)	0	40
dt	0.08	0.08
h (N·m·s)	0.044	0.044

Table 5: SimSat Commanded Torque Limits

CMG Array	$\dot{\vec{h}}_r$ (N·m)
CSCMG	[2.485,2.485, 2.559]
VSCMG	[3.479, 3.479, 3.584]
RWCMG	[4.677, 4.677, 4.752]

3.4.2 Problem A Evaluation

Section 2.3.4 introduced the simulation and training background required for this study. The purpose of this subsection is to detail the metrics and measures of success

Table 6: Hyper-parameters for RL structure and SimSat simulation

Hyper-parameter	Value
Timestep	0.08
Steps	400
Seed	0
Exploration	5000000
Batch Size	164
Discount factor (γ)	0.99
For soft update of target parameters (τ)	0.01
Policy Noise	0.2
Noise Clip	0.6
Exploration Noise	0.1
Policy Frequency	2
Evaluation Frequency	5000
Reward Threshold	1000000

necessary for validating and analyzing Problem A, the results provided in the final chapter of this research. For the SimSat simulation and observation space, the state profiles will be compared to the baseline established in [2] by examining total root mean square (RMS) error of Eq. (114), standard deviation (σ) of Eq. (115), and total root sum square error (RSS) of Eq. (113). Because the collection portion of the simulation is not optimized as in [2], each simulation run will be compared as a cumulative entity as opposed to only the time steps during the collect mode. The total number of timesteps is denoted by N, x_i is a state sample, and $\bar{x}$ the average state value. RSS gives a settling time measure for the entire maneuver and RMS measures more of the pointing accuracy by dividing by the number of timesteps.

$$RSS_{tot} = \sqrt{\sum_{i-1}^{N} \Delta_{EA_i}^2} \tag{113}$$

$$RMS_{tot} = \sqrt{\frac{\sum_{i-1}^{N} \Delta_{EA_i}^2}{N}} \tag{114}$$

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \overline{x})^2} \tag{115}$$

The metrics in addition to the RSS and RMS error values to be used to evaluate the performance of the RL controller simulation include the completion time or time it takes for the spacecraft to conduct the maneuver, the average singularity measure, and the CMG and controller torques. To evaluate the RL environment, Table 7 contains the utilized metrics. With the metrics and methodology for Problem A outlined, the methodology for Problem B will be detailed next.

Table 7: Metrics for Problem A RL Evaluation

Total Reward Over All Episodes
Average Reward Over All Episodes
Average Reward Per Step

3.4.3 Problem B Methodology

Problem B is an intensive performance review of RL and singularity avoidance with CMG arrays. With examination of the control allocation of the general spacecraft attitude control problem using RL performed by Problem A, attention shifts to performing further analysis by varying the spacecraft mass properties. It is hypothesized that a trained agent from the SimSat simulations and CMG arrays should have a generalized solution formulation useful for other spacecraft. A similar methodology to Problem A was followed for Problem B with regards to parameter usage. The PHR spacecraft physical properties are used to evaluate the simulation realism by its real-world parameters. This problem consists of conducting sequential slews with the PHR hardware parameters from Table 2 and analyzing the resulting performance. The state or observation space of this problem includes the values found in Eq. 75 and state constraints in Table 8. The action space consists of the gimbal angles ranging from $[10, -70, 10 - 70]^\circ$ to $[120, 50, 120, 50]^\circ$ and the control torque range is ± 1 N·m. For agent training, Table 10 contains the hyper-parameters used for the PHR simulations that were kept the same.

Table 8: PHR Environment State Bounds

Parameter	Minimum	Maximum
q1	-0.0231	0.0440
q2	5×10^{-10}	0.1321
q3	-0.3340	0.1530
q4	0.9338	1
ω (°/sec)	-4	4
δ (°)	-70	70
$\dot{\delta}$ (°/sec)	-57	-57
$\ddot{\delta}$ (°/sec^2)	-272	272
$\dot{\psi}_{CMG}$ (rpm)	2600	2600
$\ddot{\psi}_{CMG}$ (rpm)	-1000	1000
$\dot{\psi}_{RW}$ (rpm)	-2650	2650
$\ddot{\psi}_{RW}$ (rpm)	-90	90
u (N·m)	-1	1
m	0.3	2
t (sec)	0	40
dt	0.08	0.08
h (N·m·s)	0.044	0.044

Table 9: PHR Controller Torque Limits

CMG Array	$\vec{\dot{h}}_r$ (N·m)
CSCMG	[4.5649, 4.5659, 4.7021] 10^3
VSCMG	[6.0865, 6.0865, 6.2695] 10^3
RWCMG	[4.5682, 4.5682, 4.7054] 10^3

Table 10: Hyper-parameters for RL structure and PHR simulation

Hyper-parameter	Value
Timestep	0.1
Observation Steps	400
Seed	0
Exploration	5000000
Batch Size	164
Discount factor (γ)	0.3
For soft update of target parameters (τ)	0.3
Policy Noise	0.2
Noise Clip	0.5
Exploration Noise	0.1
Policy Frequency	2
Evaluation Frequency	5000
Episode Reward Threshold	-100
Reward Threshold	100

3.4.4 Problem B Evaluation

As outlined for Problem A, the purpose of this subsection is to detail the metrics and measures of success necessary for validating and analyzing Problem B, the results

provided in the final chapter of this research. The metrics in addition to the RSS and RMS error values to be used to evaluate the performance of the RL controller simulation are included in Table 11, where the intent of Problem B is to further examine the singularity avoidance abilities of the RL controlled system with mass properties unlike SimSat's. The RL agent's performance is evaluated using the metrics in Table 23 in addition to the state properties of attitude, angular rate, gimbal angles, and gimbal rates.

Table 11: Combined Metrics for Problem B Evaluation

Number of Near Singularities
Average Singularity Measure
Controller Torque
Computation Time
Total Average Reward Over All Episodes

This concludes the methodology chapter. A summary of the spacecraft simulation, control scheme, and reinforcement learning control structure was provided. These details were structured into two problem areas of interest that sought to assess the performance capabilities of machine learning elements inside an ADCS. Chapter IV provides the results of applying the methodology and metrics of Chapter III and the analysis of SimSat and PHR simulations.

IV. Results and Analysis

The result sections of this chapter are organized around the two previously presented problems. Section 4.1 contains the results and validation of the RL agent for Problem A with baseline comparisons. Section 4.2 contains the results and validation of the RL agent using the Pleiades High Resolution (PHR) spacecraft parameters, while conducting a singularity avoidance analysis. Due to the averaging of values over several thousands of episodes, minor differences exist between each presented state profile from the RL controlled system. RL controller evaluation conducted after training consists of taking the averaged and best version of a trained model over several time steps and subsequent episodes. It is assumed this final model version would be implemented in a real-time environment and system. The RL environment is run to convergence (goal satisfaction and attainment) and the number of time steps and episodes are not a primary measure of performance. It is also assumed that the several iterations required for convergence are not representative of the time spent on-orbit by the RL controller and instead a final product would be used to conduct attitude control. Data trends are described by individualized comparisons and converged solutions are used in this analysis. A generalized assessment of these problem areas are summarized in Section 4.3

4.1 Problem A Results

The simulation environment was formulated to contain the observation and action spaces, dynamic equations, and integration schemes required for all three CMG arrays as discussed in Section 3.2. Once the environment was developed and integrated with the TD3 algorithm, state profiles were generated. For consistency, the agent begins in the same initial attitude and angular velocity and uses no seed value to change the

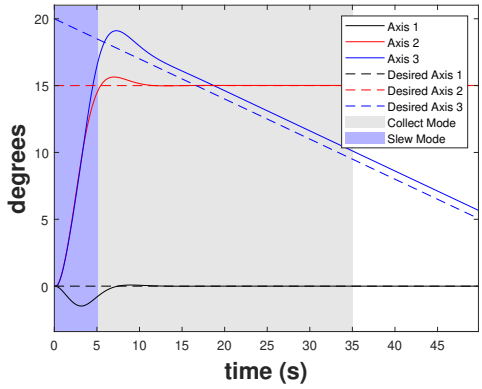
initialization state of a pseudo-random number generator as no generated reward was used in this study. While global convergence was not obtained, the agent successfully found a solution (local convergence) for each CMG array and details are listed in the following sections. It was assumed that global convergence can be feasibly obtained after several weeks of training and modification to the reward functions.

4.1.1 Baseline Results

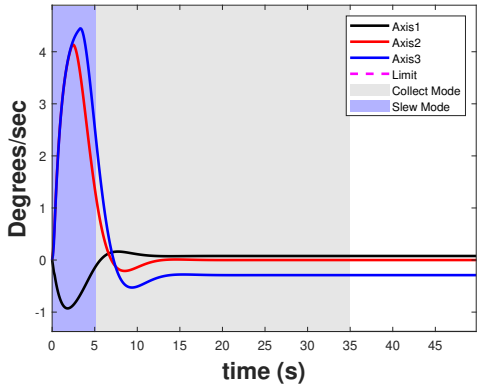
The results generated from [2] serve as baseline values to be compared to the RL results. The baseline simulation was modified to use a singular target at a coordinate of (300, -250) on one target deck and runs using a timestep of 0.08 seconds. Other simulation selections include the use of hardware limit enforcement, the Vadali angles as the preferred/initial gimbal angles, no filtering, no additional Gaussian noise, and a QF controller. The hardware limit enforcement of [2] consists of enforcement by the selected controller to check the maximum spacecraft torque and slew rate, a saturation function, and calculation of anticipated hardware limits. The omission of filtering techniques in the current study and noise is to enable simplistic comparison to the RL results. The QF controller was selected to allow comparison between a novel control technique such as RL to a conventional state feedback control strategy, where the QF gains were tuned for a multiple target slew mission. The state profiles for the CSCMG array are in Fig. 19 and illustrate a full simulation run time of 50 seconds, slew and collection mode tracking, and the desired and measured values of the controller using dotted and solid lines. The desired window of viewing for this simulation is from the beginning of the slew mode to the end of the collect mode, which equates to between 35 to 40 seconds among all arrays. The time from the end of the collect mode is discarded and as a result, the RL results were run for an approximate amount of time of 40 seconds. This research seeks to evaluate the

similarity in state profiles from the RL solution and recognizes that the solution may vary from these profiles. The goal of the RL agent is to achieve the 3-2-1 Euler angle attitude and spacecraft angular velocity at 17 seconds into the simulation, which have the values of [θ, ϕ, ψ] = [0, 15, 15.944] deg and [0.0739, 0.0052, -0.2995] deg/s, respectively. The corresponding quaternion orientation is $[q_1, q_2, q_3, q_4]$ = [-0.0187, 0.1290, 0.1425, 0.9815]. The simulation time of 17 seconds was selected due to the two axis value similarity and crossover that occurs around this time (as seen in Fig. 19a) with the intent for the agent to recognize this similarity and not require a large control effort with the three vastly different attitude components. The QF controller produced constraint violations for only the gimbal acceleration. Mode tracking for the CSCMG array is defined as the CMGs conducting the slew and collect functions. It can be observed according to the shaded regions of the state profiles, slewing occurs from 0 to 5.1 seconds and collection from 5 seconds to 35.1 seconds.

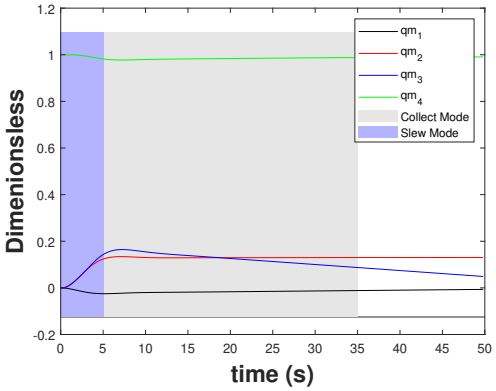
The state profiles for the VSCMG and RWCMG array are similar and are instead found in Appendix E, where the attitude and angular velocity state profiles are omitted due to their identical nature. For the VSCMG array, the QF controller produced constraint violations for only the gimbal rate and acceleration. It can be observed according to the shaded regions of the state profiles, slewing occurs from 0 to 5.1 seconds and collection from 5 seconds to 35.1 seconds. The QF controller used in the RWCMG array case produced constraint violations for only the gimbal acceleration. It can be observed according to the shaded regions of the state profiles, slewing occurs from 0 to 5.1 seconds and collection from 5 seconds to 35.1 seconds.



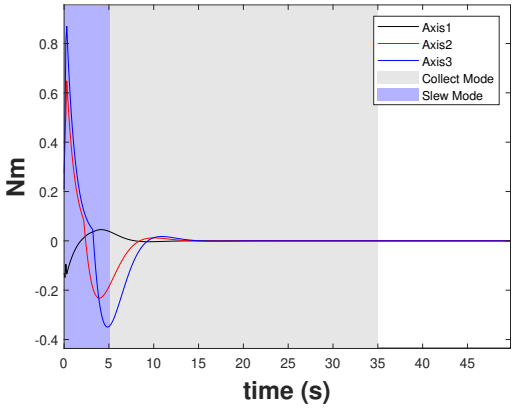
(a) Attitude (Euler Angles)



(b) Angular Rate



(c) Attitude (Quaternions)



(d) CMG Torque

Figure 19: CSCMG array baseline values

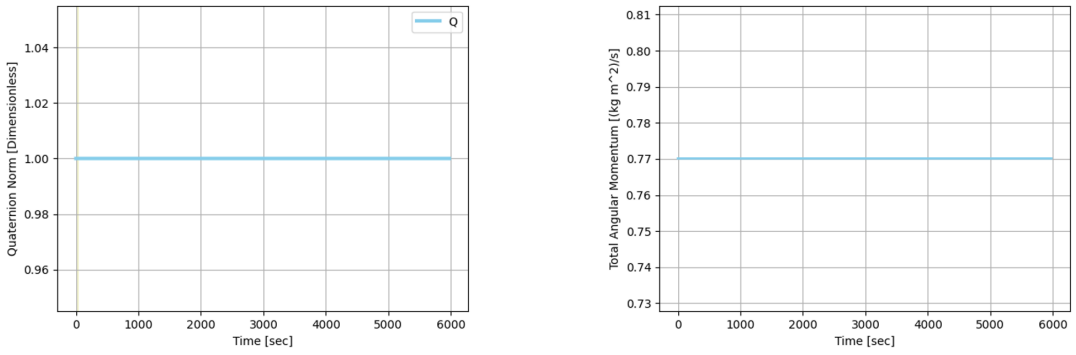
4.1.2 Reinforcement Learning Training Results

A solution satisfying the desired attitude and angular velocity was not obtained after several adjustments to the training process to include hyper-parameters, actor-critic architecture, and initial state values. The agent was able to produce system control for each CMG array but not to the desired attitude. In order to evaluate the agent's overall performance, the training process and results will be detailed next.

The simulations for each CMG array were started at the same time and were set to terminate once goals were reached. This termination accounts for the jagged and rough edges of the training state plots. The time attached to each plot is a one-tenth representation of the actual number of timesteps spent per episode in order to best visualize the training results. Early training termination was conducted after observing general and undesired state trends for each array. As such, each CMG array ran for different lengths of time and with subsequent different numbers of timesteps, episodes, and reward values. The training process began with default values and actor-critic architecture of [83] to generate a baseline of performance. While the agent's performance cannot be assumed by qualitative measures alone, general trends of the states in the first few hundreds of episodes were used to tune the system. State constraint violations were prevalent in early training episodes and during hyperparameter tuning; as such, the simulation environment was modified to bound the agent's search space. Inclusion of two attitude parameterizations in the form of Euler angles and quaternions improved agent search ability by strengthening the importance of achieving a desired attitude. The reward functions were critical to training success and changes to this formulation often created large variations in agent search ability and more so than hyper parameter changes. There were occasions where the agent accomplished or neared the attitude and/or angular rate goal, triggering the 2 degree threshold. However, the objective was for the agent to consistently meet these goals and reflect

this accomplishment through state values mirror the baseline values.

General trends specific to the CSCMG array during agent training were multifarious. One encountered challenge was the impact of early episode termination, as aforementioned in Section 3.3.3. Building early episode termination into the reward formulation generated additional episodes that caused the agent to minutely change axis control values and cause small variations to the state values. Having small variations increased the time spent by the agent finding the solution and this was remedied by changing the reward function as the undesired training results were encountered. Another general trend specific to the CSCMG array was minimal state constraint violations and rapid agent learning of staying within these boundaries. A check conducted for all arrays to ensure state values obeyed system constraints was through the use of the total angular momentum magnitude and quaternion norm. Conservation of these values is required according to the dynamic laws governing spacecraft attitude dynamics and the agent produced practical control, as evidenced by Fig. 20.



(a) Quaternion 2-Norm

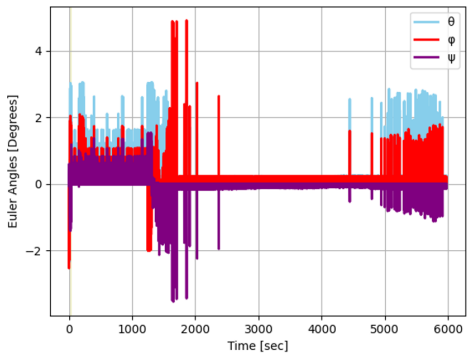
(b) Total Angular Momentum Magnitude

Figure 20: Desired conservation of the quaternion norm value of 1 and the system anuglar momentum magnitude - CSCMG Array

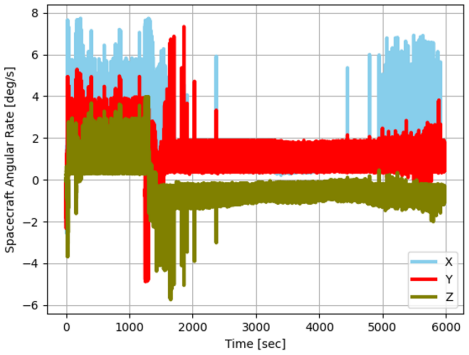
The total training burden for the CSCMG array was 10 days, 21 hours, 1 minute, and 34 seconds, a computationally expensive exercise for the research hardware. A total of 14,801 episodes and 59,708 timesteps were conducted by the agent. The state

profiles for this training process are captured in Fig. 21, where time on the x-axis was scaled to a time step of 0.1 to improve visualization and only attitude, angular velocity, and torque values are provided. The VSCMG and RWCMG array training profiles will follow this presentation style and all other state training profiles are found in Appendix F. The generated gimbal torque was omitted in the context of Problem A. An indication of agent ability in these state values is the observation of the Euler angles and quaternion. The solution requires for these attitude parameterizations to follow the baseline values in Fig. 19a and 19c and deviation away from these values is undesirable. The agent experienced limited success in producing control to replicate the desired state values. It was observed during training the agent conducting small and inefficient control variations per time step, causing the results of Fig. 21c and 21d. Controller saturation and a low number of near singularities are considerable factors that may explain this behavior but additional episodes may reveal new trends.

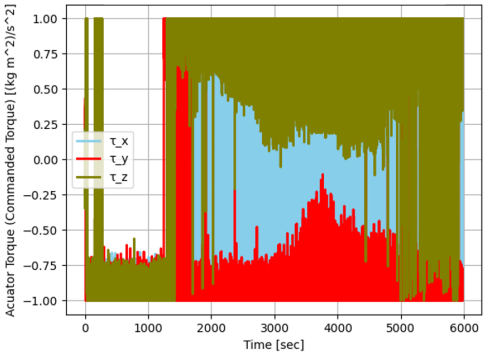
Another indication of agent success during training is through the observation of reward values. An averaged and total reward amount over all episodes were calculated. When training, the average and total reward values should show positive trends to indicate the learning ability of the RL agent. Dips in these graphs throughout the simulation are expected, with larger dips desired in earlier episodes for the agent to learn to avoid obvious and undesirable pitfalls. At simulation termination, the total and average reward values were -9416.499 and -7090.407, respectively. The reward formulations for all arrays allows for positive reward values, where values closer to or above zero indicating greater effects from positive or less negative rewards. These large reward values suggest the agent was unable to be effectively guided towards the goal yet the episode plots of Fig. 22 illustrate a positive slope and gradual improvement of the agent. The small variation of the reward values for accounts for the agent's behavior of changing the control minutely.



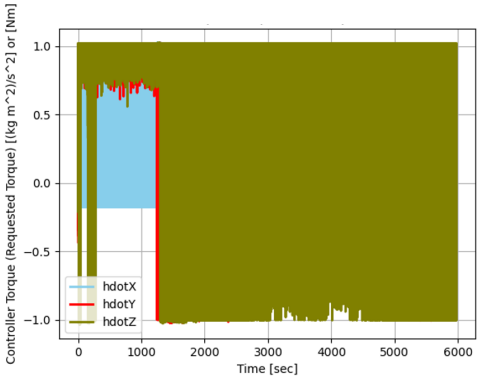
(a) Attitude (Euler Angles)



(b) Angular Rate

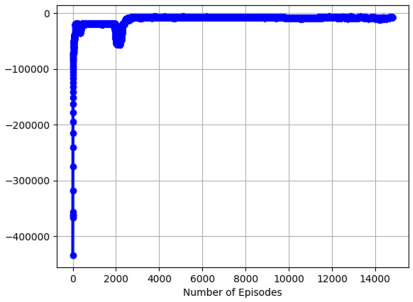


(c) CMG Torque

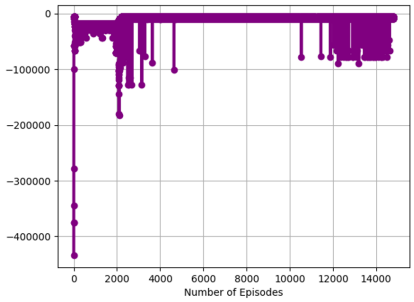


(d) Controller Torque

Figure 21: CSCMG Array Training Values



(a) Average Reward



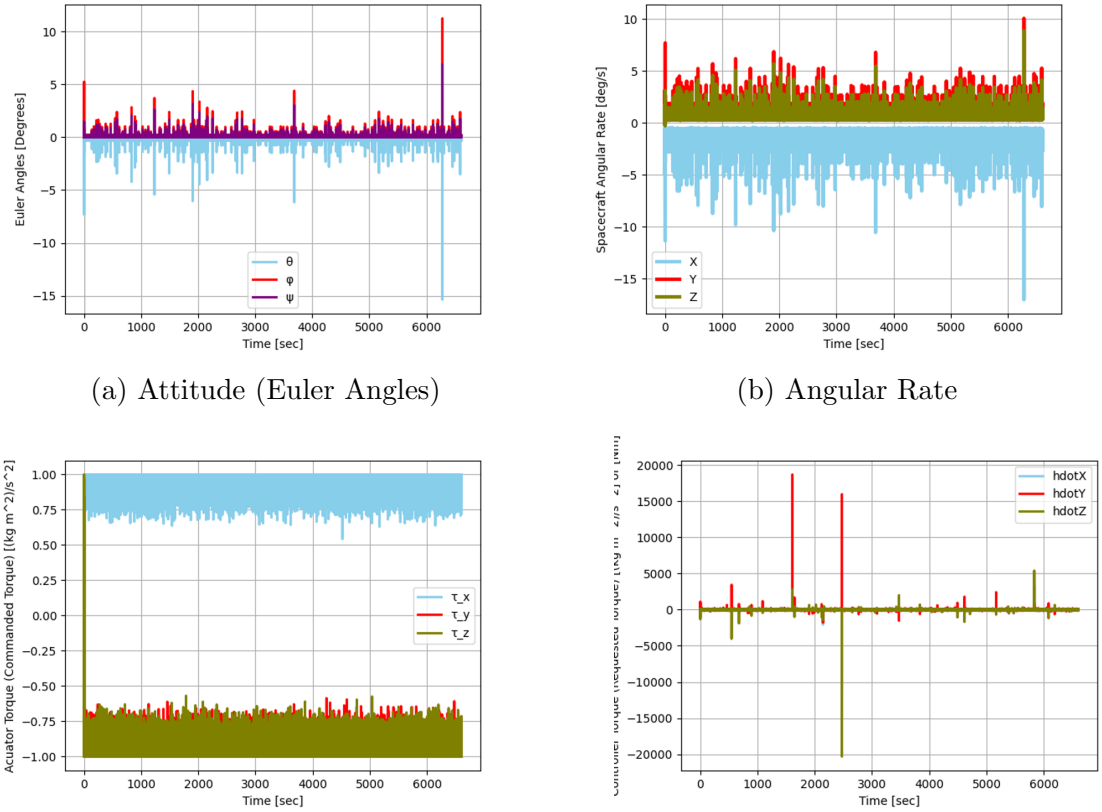
(b) Total Reward

Figure 22: CSCMG Array Training Reward Values Over All Episodes

Training trends specific to the VSCMG array will be described next. Despite a similar reward function formulation to the CSCMG, the VSCMG experienced dis-

similar training trends. The number of near-singularities was a reoccurring behavior that in turn caused several state constraint violations. While the CMG skew angle was unchanged between all three arrays, the momentum envelope appeared to be too small for the VSCMG array with the number of near-singularities. The VSCMG geometry has complexity that the RL controller is unable to compensate for and the near-singularity issue may be remedied with the use of different skew-angles and hardware limits. The training burden for the VSCMG array was 5 days, 1 hours, 38 minutes, and 0 seconds. A total of 16,173 episodes and 65,612 timesteps were conducted by the agent. The state profiles for this training process are captured in Fig. 23. Saturation appeared to be ignored by the agent in its solution search, producing a controller torque large in magnitude. This torque gradually improved over several thousands of episodes but did not avoid saturation. The produced CMG torque of Fig. 23c illustrated some ability by the agent to recognize the required amount of actuator torque by producing values closer in magnitude to the baseline values. The difference in requested and commanded torque value magnitude suggest the agent was therefore overcompensating with the VSCMG array. The rotor acceleration was the most frequently violated state by the agent and several negative rewards were imposed to mitigate this issue. Over time the agent was able to reduce the number of violations which are mostly illustrated by irregular vertical lines in several state plots. The Euler angle attitude and angular rate values share similarity in sign and magnitude to the desired solutions.

At simulation termination, the total and average reward values were 7747.441 and 8555.079, respectively. Figure 24 captures the training reward values. The positive attributes of these reward values coupled with the attitude and angular rate values indicated agent success towards finding the solution. This attribute contrasts with the negative trend in the first 500 episodes and small fluctuation of values reflected



(a) Attitude (Euler Angles)

(b) Angular Rate

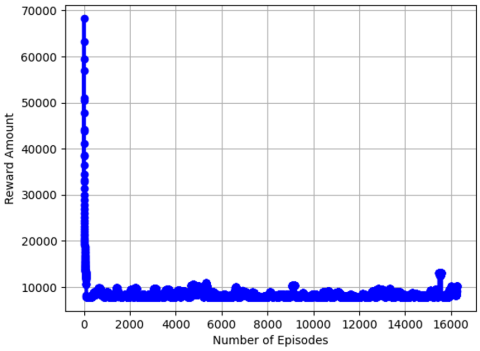
(c) CMG Torque

(d) Controller Torque

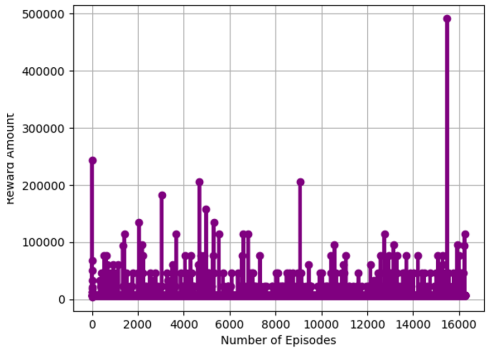
Figure 23: VSCMG Array Training Values

in the reward value history, signifying the agent is slowly searching the solution space and may be captured in a solution well. Observing the conservation of total angular momentum magnitude and the quaternion norm throughout training did not reveal any performance abnormalities, as seen in Fig. 25.

The RWCMG training trends shared some similarity with those of the CSCMG and VSCMG arrays. The corresponding state profiles for this array are found in Fig. 26. The agent spent a considerable amount of episodes exacting a minimal amount of control that caused small variations in state values. Although less saturation can be observed for this array in Fig. 26c and 26d, the agent did conduct control moderation with the exception of some constraint violations in earlier episodes. Adopting a

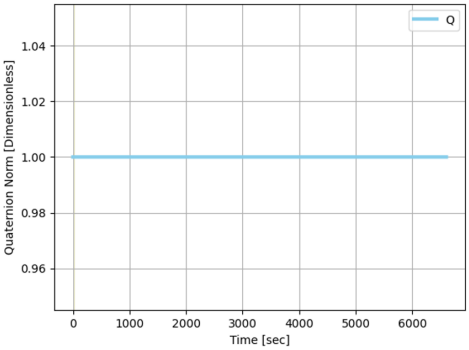


(a) Average Reward

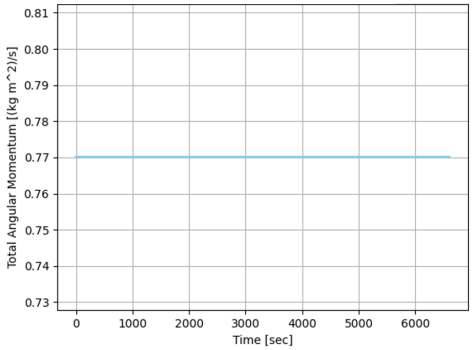


(b) Total Reward

Figure 24: VSCMG Array Training Reward Values Over All Episodes



(a) Quaternion Norm



(b) Total Angular Momentum Magnitude

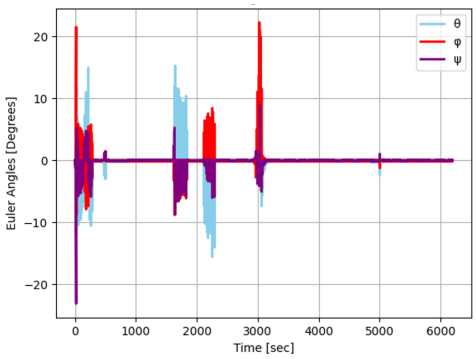
Figure 25: Desired Conservation of Values - VSCMG Array

similar reward function containing negative rewards associated with state constraint violations to the other CMG arrays generated more violations during training and were therefore omitted. The steering logic complexity and dynamics of the RWCMG array may have increased the sensitivity of changes to the simulation and/or agent environment. The least amount of hyperparameter tuning was practiced for this array as the agent appeared to compensate quickly. The training burden for the RWCMG array was 9 days, 14 hours, 29 minutes, and 28 seconds. A total of 18,153 episodes and 61,923 timesteps were conducted by the agent.

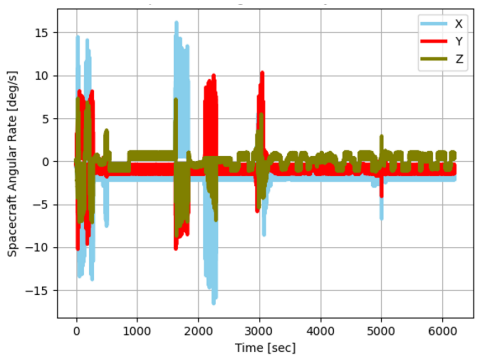
The reward function is purposely structured to improve illustration of agent per-

formance. Using non-normalized values allows for magnification of any large changes of states and reward values. Normalizing the reward function produced the same results, albeit with small numbers that did not fully convey environmental outputs, and the original additive property of the cumulative reward was retained. The total and average reward values were -124,438,788.824 and -124,153,764.146 , respectively, at simulation termination. Figure 27 captures these training reward values. Having large reward magnitude values that are also in the same value vicinity convey the agent is conducting more exploitation than exploration in an undesired solution area. The negative slopes of the reward functions also supports this assertion. Observing the conservation of total angular momentum magnitude and the quaternion norm throughout training did highlight as seen in Fig. 28. While the quaternion norm was maintained, total angular momentum magnitude was not constant throughout the training process. The spikes seen in Fig. 28b originate from numerical integration errors and also mirror the constraint violations in the states of attitude, gimbal angles, gimbal rates, gimbal acceleration, reaction wheel rate, and reaction wheel acceleration. It can be likely surmised that the addition of the reaction wheels may be causing some instability, as this angular momentum behavior was not observed in the non-reaction wheel containing CMG arrays.

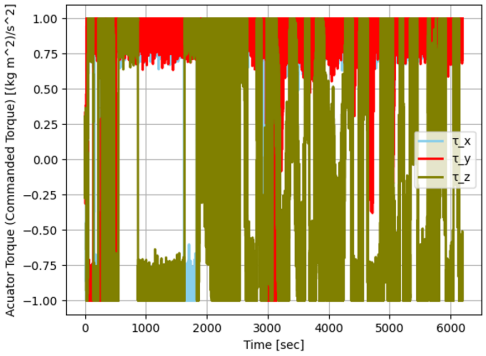
The training process revealed the CSCMG array required the most tuning compared to the other CMG arrays and early training efforts were besieged with infinite and"not a number" (NaN) values as the agent conducted at times wild control efforts. Despite the reward value trends of CSCMG array, it appeared the RL agent was the least successful in searching the solution space. The CSCMG array solution space was the most difficult for the RL agent to navigate. This is further supported by the simulation model saving the actor-critic architecture for post-simulation evaluation the least amount of times, where a save is triggered by the average reward of an episode



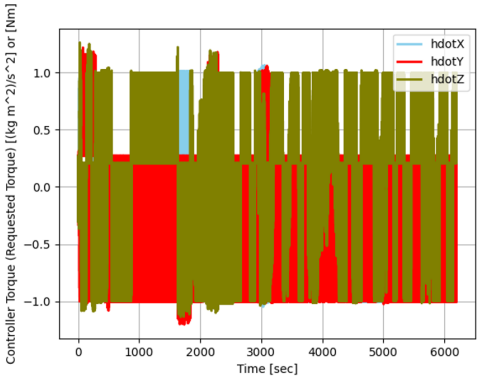
(a) Attitude (Euler Angles)



(b) Angular Rate

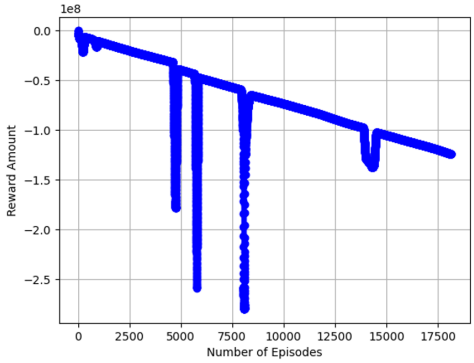


(c) CMG Torque

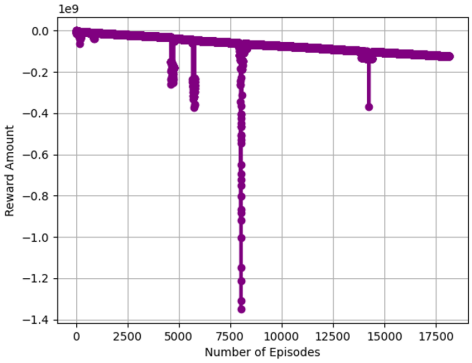


(d) Controller Torque

Figure 26: RWCMG Array Training Values

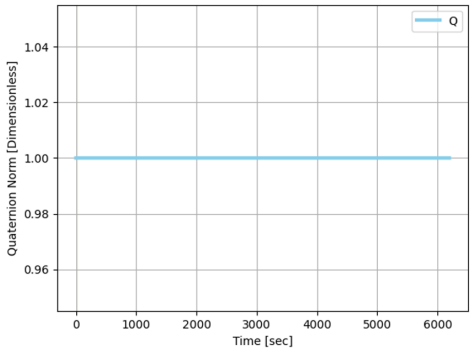


(a) Average Reward

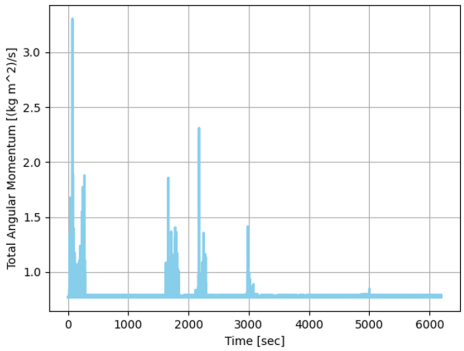


(b) Total Reward

Figure 27: RWCMG Array Training Reward Values Over All Episodes



(a) Quaternion 2-Norm



(b) Total Angular Momentum Magnitude

Figure 28: Desired Conservation of Values - RWCMG Array

being larger than best running average. Reward formulation and CMG skew angle geometry are likely candidates for improving RL agent performance. The VSCMG was the only array to have consistent and positive reward values, which reflected a greater likelihood of the agent searching near the goal. The difference in timesteps and episodes between the arrays provides some additional insight. All CMG arrays had adjacent episode and timestep values, despite differences in simulation run time. The CSCMG simulation ran for almost twice the length of time of the VSCMG, suggesting an efficiency to the VSCMG that was apparent in the agent taking less time to exploring more of the solution space. The VSCMG simulation run time was also indicative of less value gleaned from each episode. The existence of NaN values that caused training termination prohibited long training times but similar numbers of episodes compared to the CSCMG and RWCMG arrays. The agent may have spent more time exploiting one solution that took less time than exploring various solutions. The RWCMG and CSCMG arrays shared similar runtimes, yet the RWCMG took the most number of episodes out of all the arrays. The total number of episodes does not directly correlate to efficiency, but agent behavior was largely unimproved throughout the training duration of the RWCMG array. Along with the individual training

trends and results, an evaluation of the training in one hour was conducted. The results of one hour's worth of training are found in Table 12, where larger differences in values are interpreted to signify the amount of work exacted by the agent during its solution search. The RWCMG array had the largest reward differences that may have been due in part to vacillating applied control from having two sets of actuators.

Table 12: Absolute value differences in one hour of simulation runtime

Array	Total Time Steps	Episode Number	Total Reward	Average Reward
CSCMG	378	110	3578.599	322.074
VSCMG	402	99	10.0288	305.141
RWCMG	390	130	808,621.173	785.369.293

With the training process and results outlined for each CMG array, final agent evaluation will be described in the next section.

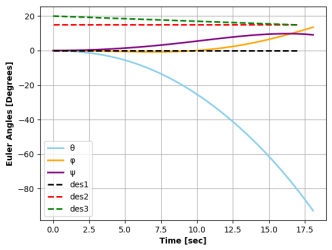
4.1.3 Agent Performance Evaluation

The analytical comparison of the baseline and RL results consists of metrics first introduced in Section 3.4.3. Qualitative analysis of the state profiles will be conducted first and followed by RL agent evaluation. It can be observed that while the RL observation and action space were constrained by the baseline results, there exists no expectation of exact solution attainment. The QF controller produces a unique solution and the RL agent is not expected to produce duplicate state profiles, but will instead produce another unique solution. The comparison between the QF and RL controllers will seek to evaluate the attitude and spacecraft angular rate at 17 seconds and calculate differences between the solutions for the control effort and mode tracking. Due to non-global convergence, the completion time of the RL agent is the amount of time it takes to reach an episode termination condition and will therefore run past the 17 seconds. State profiles from the CSCMG are found in Fig. 29, the VSCMG in Fig. 30, and the RWCMG in Fig. 31. The baseline values

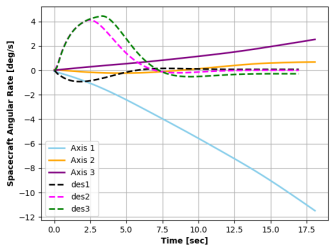
using dotted lines are plotted against the RL solid line results for reference. Ten evaluations of 100 episode runs were conducted for RL agent evaluation. The total average reward values after the CSCMG, VSCMG, and RWCMG array evaluations were -33,694,745.614, 1,981,035.314, and -73,037,158,437.831 respectively. As found within training, values closer to and greater than zero indicate desired performance and a tendency maximization of a cumulative reward by the agent.

It can be observed for the CSCMG array, the agent appeared to follow baseline values more closely for the actuator torque, gimbal rate, gimbal acceleration, and controller torque. These established trends display a degree of learning the agent was able to accomplish despite the large search space. Constraint violations were found in all states during training, but in later episodes these violations were not found with the gimbal rate, acceleration, requested, and commanded torque values. This may have been due to the reinforcement from the reward functions the RL agent was able to follow after several episodes of learning. For control allocation, the requested and commanded torque values of Fig. 29d and 29h reached near desired values and was within saturated value tolerances. The RL agent was therefore partially successful in conducting control, albeit for an undesired attitude and angular rate. Early episode terminations did not allow the agent to search the space and instead negative rewards were provided. Each state profile illustrates the agent staying within the constraints but also violating them throughout the time series history. This suggests that the agent was prioritizing the overall goals despite state values and this is supported by the agent ignoring controller saturation logic as seen in Fig. 29h. Another item of note is the largest constraint violation in the first axis of the Euler angle and angular rate plots. The baseline values for these two states from Fig. 29a and b show negative values for the x-axis. During training, the agent often ignored the solution uniqueness aspect of the first axis but maintained the uniqueness for other axes. In

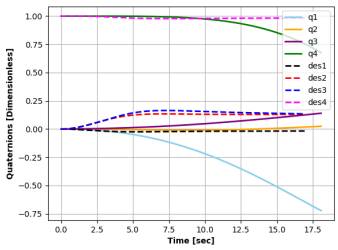
general, the agent learned control ability for this array and suffered from solution uniqueness and a large search space. From the total average reward values of each array, the CSCMG array had the second largest value and despite the negative sign, the value does embody some desired learned behavior.



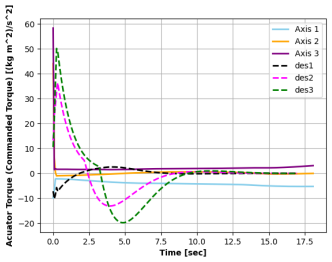
(a) Attitude (Euler Angles)



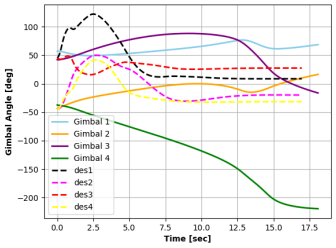
(b) Angular Rate



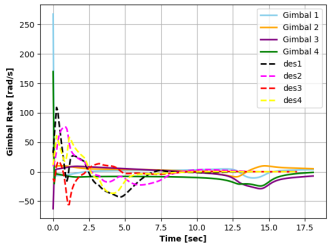
(c) Attitude (Quaternions)



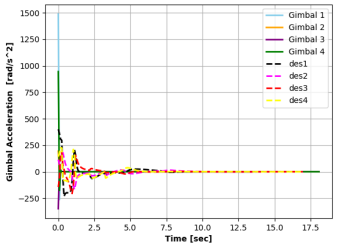
(d) CMG Torque



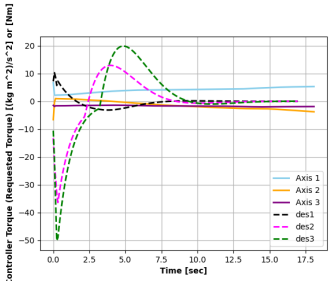
(e) CMG Gimbal Angles



(f) CMG Gimbal Rate



(g) CMG Gimbal Acceleration



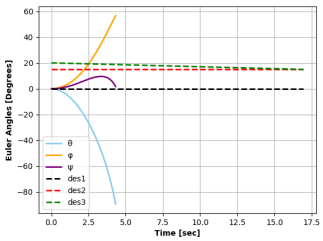
(h) Controller Torque

Figure 29: Evaluated Agent performance - CSCMG Array

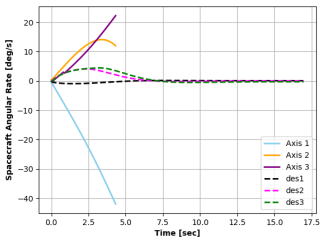
The VSCMG array agent evaluation yielded results that differed the most out of all of the arrays compared to its training results. The agent was unable to follow baseline trends for both attitude parameterizations, the angular rate, gimbal angles

and rate, and the controller torque. The evaluation notably terminates well before 17 seconds which indicates highly brittle performance boundaries the agent was unable to learn. The contrast between the training and evaluation may be due to system instability and the initial amount of acceleration the gimbal and rotors require. The baseline values of Fig. 40 for the gimbal rate and acceleration as well as the rotor rate and acceleration show some constraint violations occurring during initial spin up. The agent appeared to struggle with applying feasible control with these rates and accelerations, generating attitude and angular rate values well beyond their desired ranges. However, the agent followed the trends within these states, suggesting some learned behavior was accomplished. The divergence of evaluated solution demonstrably proves that the total positive reward value for training and testing is not a sole determinant of agent success. The VSCMG array had an actor-critic architecture that originally was a duplicate of the CSCMG and RWCMG architectures. The network was found to acutely inhibit the RL agent from searching the solution space and generate an impractical training results. The number of neurons in this networks was eventually decreased in magnitude, which enabled the agent to produce training and subsequently evaluation results, but at a cost to performance. The agent encountered the most difficulty in prioritizing the goal, despite the reward function formulation. It can be inferred that an unintentional amount of weight was placed upon avoiding constraint violations such that the importance of the attitude and angular velocity goals were minimized. The agent was therefore restricted to an undesirable portion of the search space and become entrapped. It would be necessary to have utilized a more robust constraint violation strategy as opposed to the binary version this research employed.

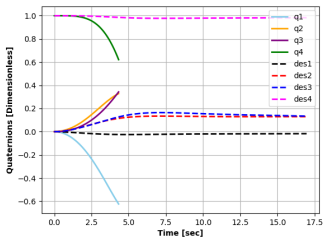
For the RWCMG array, the agent followed more general trends than the VSCMG but less than the CSCMG arrays. The greatest divergence can be observed for the



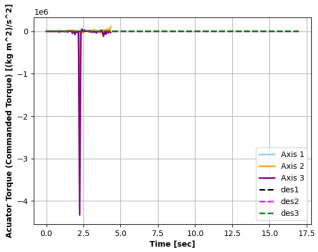
(a) Attitude (Euler Angles)



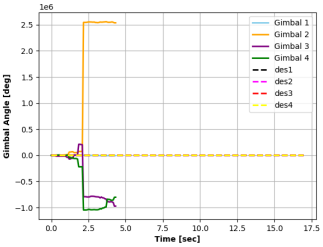
(b) Angular Rate



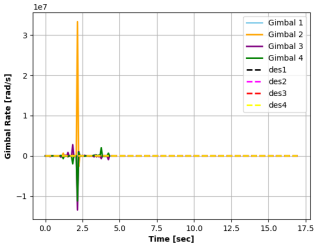
(c) Attitude (Quaternions)



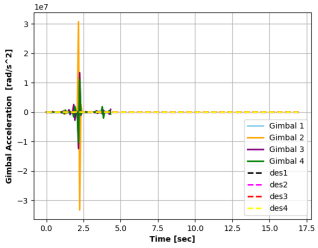
(d) CMG Torque



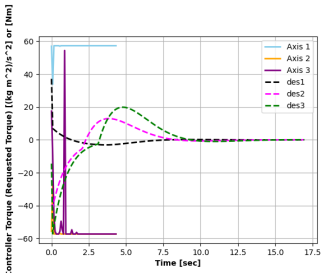
(e) CMG Gimbal Angles



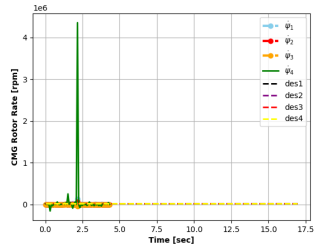
(f) CMG Gimbal Rate



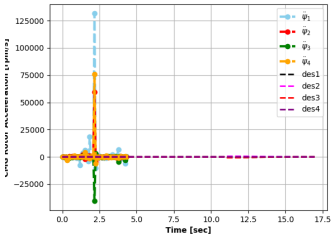
(g) CMG Gimbal Acceleration



(h) Controller Torque



(i) Rotor Spin Velocity



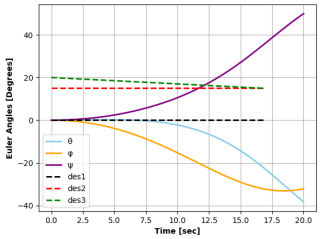
(j) Rotor Spin Acceleration

Figure 30: Evaluated Agent performance - VSCMG Array

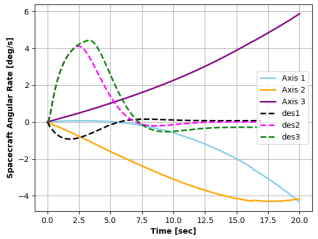
attitude and angular rate states. It was hypothesized that the steering logic and dynamic complexity of this array would generate the most oscillatory and undesired

behavior, which was shown to be partially untrue. While the training state values and total evaluated reward indicate less than ideal amounts of learned behavior, the agent was able to learn trends and in particular steady-state trends for all states besides the attitude and angular rate. This array benefits from having two different types of actuators and as shown in [2], has some additional performance benefits compared to a CSCMG and VSCMG. These performance benefits, while not outperforming the other arrays in the context of this research, may be realized qualitatively by the achievement of reduced oscillatory and unstable behavior. The agent encountered difficulty learning solution uniqueness and overcompensated as seen by the spikes in the plots for gimbal rate and acceleration and reaction wheel spin velocity and acceleration. The largest of these spikes can be observed with the reaction wheel spin velocity and acceleration, where the RL agent is causing the reaction wheels to overcompensate. Based on the weighting matrix of Eq. (99), the CMGs have a higher weight and are therefore the favored device for the majority of the simulation. Some loss of conservation of angular momentum occurs and the reaction wheels struggle to maintain the current state values and cause vertical lines in the reaction wheel state profiles. Excluding the spikes and producing desired state trends would come from placing greater importance on other state values or changing the weighting matrix of the actuator set.

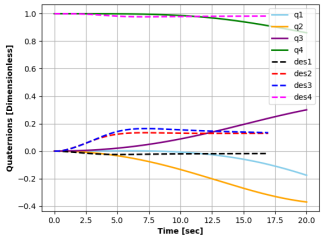
Mode tracking was unable to be accomplished by all three arrays. This tracking occurred when the attitude was within a desired threshold but conditions were not met by the RL agent. It should be noted that while the VSCMG does not have reaction wheels, holding the CMG rotor rates constant will cause its CMGs to behave as reaction wheels and this is explain in Section 2.2.3. As previously mentioned, the agent truly learns a behavior once the behavior is replicated by the agent during the evaluation phase. The brief expression of this behavior during training suggests



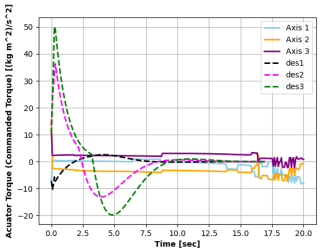
(a) Attitude (Euler Angles)



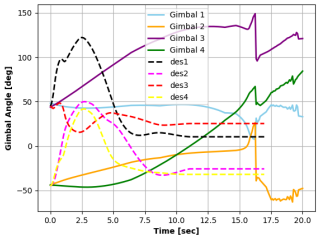
(b) Angular Rate



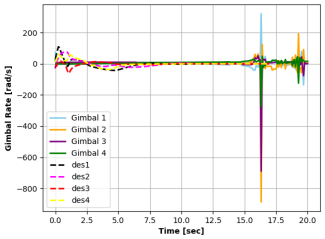
(c) Attitude (Quaternions)



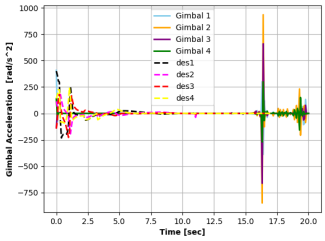
(d) CMG Torque



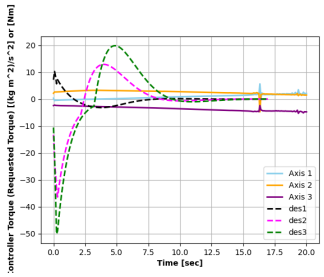
(e) CMG Gimbal Angles



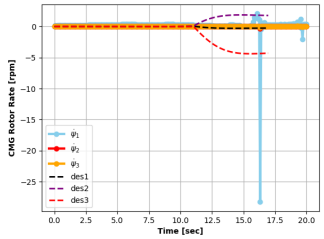
(f) CMG Gimbal Rate



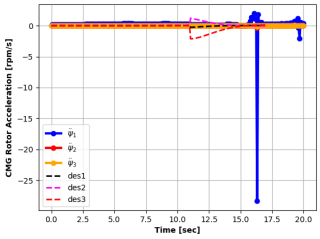
(g) CMG Gimbal Acceleration



(h) Controller Torque



(i) RW Spin Velocity



(j) RW Spin Acceleration

Figure 31: Evaluated Agent performance - RWCMG Array

the agent was close to the solution but wandered away within the search space. The primary challenge during training for each array was to guide the agent to the

solution, but as seen for the mode tracking, too many states and requirements can lead to solution divergence. Mode tracking can possibly be improved from additional state inputs, algorithm application, or reward function formulation.

Having conducted a qualitative analysis of each array, attention turns to the numerical analysis portion. Additional metric comparisons for the CSCMG and RWCMG arrays and are found in Tables 13 and 19, where the RL agent is compared to the QF-produced values. Focus is placed upon the goal parameters of attitude and angular velocity. The Euler angle and angular rates found in these tables are magnitudes and average values over all time steps. Large magnitudes of Euler angles and angular velocities have been wrapped to 2 π. There should be no expected variance in the minimum and maximum values of the Euler angles and angular rates between arrays. Minimum and maximum values for the attitude and angular rate are tabulated in order to observe the bounds the agent performed.

For the CSCMG array, the RL controller saw the best performance in the angular rate, with an increase of standard deviation value of 31 percent and for the maximum value a decrease of 57.57 percent. The RL mean pointing error, defined in Eq. (103), was larger than the first time step's pointing error value of 25 degrees using the QF controller, highlighting the agent had a difficult time initially learning to decrease the pointing error. The large magnitude in RSS and RMS errors values indicates the large distance away the agent was from the actual goal, despite plotted state values. Searching for the unique attitude and angular velocity proved to be unsuccessful. The metrics for the VSCMG array follow similar value trends as the CSCMG array, but almost every metric had a large and undesired magnitude value. As seen in the qualitative analysis, the agent did not learn as many desired behaviors compared to the CSCMG and the large metric values of Table 19 reflect this fact. Analysis of these large magnitudes will subsequently not be explored. The RWCMG array had metric

value performance between the CSCMG and VSCMG arrays with slight differences in trends. Absolute differences for the Euler angle and angular rate standard deviations and mean values were between those of the CSCMG and VSCMG arrays, yet the smallest percent difference of 1.43 percent for the maximum angular rate was found. The smaller absolute and percent differences displayed with the RWCMG array metrics suggest the agent was able to some extent the angular rate importance, albeit independently of the attitude.

Table 13: CSCMG Metric Comparisons - Problem A

Metric	QF	RL
Completion Time (sec)	17	20.24
Mean Pointing Error (deg)	0.9396	29.489
Mean Euler Angle (deg)	19.1454	849.292
Mean Angular Rate (deg/s)	1.6794	2.694
Euler Angle Std. dev.	0.1272	14.823
Angular Rate Std. dev.	0.0351	0.0462
Min Euler Angle	9.923E-08	1.858
Min Angular Rate	9.923E-08	-11.486
Max Euler Angle	24.6497	776.286
Max Angular Rate	5.9656	2.531
Total Euler Angle RSS Error (deg)	26.1196	1.11E+09
Total Euler Angle RMS Error (deg)	0.6946	2.21E+03

Table 14: VSCMG Metric Comparisons - Problem A

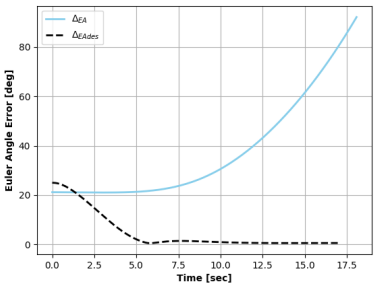
Metric	Euler Angle	Angular Rate
Completion Time (sec)	17	20.24
Mean Pointing Error (deg)	0.9291	1.38E+03
Mean Euler Angle (deg)	19.2902	0.9261
Mean Angular Rate (deg/s)	1.6866	2.037
Euler Angle Std. dev.	0.1246	8.46E+03
Angular Rate Std. dev.	0.0351	3.86E+02
Min Euler Angle	9.923E-08	0.796
Min Angular Rate	9.923E-08	-41.948
Max Euler Angle	24.6497	4.160
Max Angular Rate	5.9656	22.233
Total Euler Angle RSS Error (deg)	26.1196	3.04E+08
Total Euler Angle RMS Error (deg)	0.6946	1.11E+09

The metric analysis has revealed the RL agent appears to follow some attitude and angular rate trends but somewhat independently of each other, producing large errors in the process. The CSCMG array had the smallest percent and absolute differences out of all arrays. Reductions in error were best accomplished by the RWCMG per parameter but overall performance suffered compared to the CSCMG

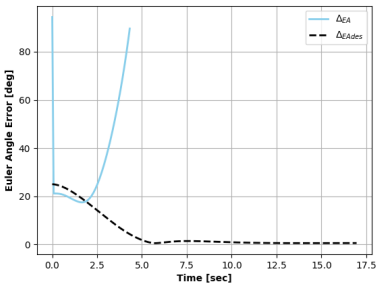
Table 15: RWCMG Metric Comparisons - Problem A

Metric	QF	RL
Completion Time (sec)	17	20.24
Mean Pointing Error (deg)	0.9396	22.403
Mean Euler Angle (deg)	19.1454	136.967
Mean Angular Rate (deg/s)	1.6794	120.448
Euler Angle Std. dev.	0.1272	2.391
Angular Rate Std. dev.	0.0351	2.102
Min Euler Angle	9.923E-08	5.398
Min Angular Rate	9.923E-08	-4.33
Max Euler Angle	24.6497	1.1507
Max Angular Rate	5.9656	5.88
Total Euler Angle RSS Error (deg)	26.1196	4.26E+08
Total Euler Angle RMS Error (deg)	0.6946	1.30E+03

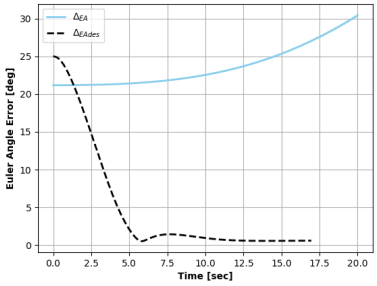
array. The VSCMG array saw the least desirable performance as a result of the agent having the least intuition as where the goals were. To better understand the attitude performance, a pointing accuracy evaluation will be discussed next and time series plots of the mean pointing error defined by Eq. (103) for each array will be utilized. It can be observed in Fig. 32 the agent's inability to target the desired attitude orientation. The agent was rewarded for reducing this error and produced varying results. All three arrays displayed similar behavior of the agent diverging within the first few seconds of the simulation. The smallest difference between the RL agent results and desired values was with the CSCMG array. It is assumed that the reduced steering logic complexity of this array eased agent learning and this is partially quantified by the evaluation plots.



(a) CSCMG Array



(b) VSCMG Array



(c) RWCMG Array

Figure 32: RL resulting pointing errors for each CMG array

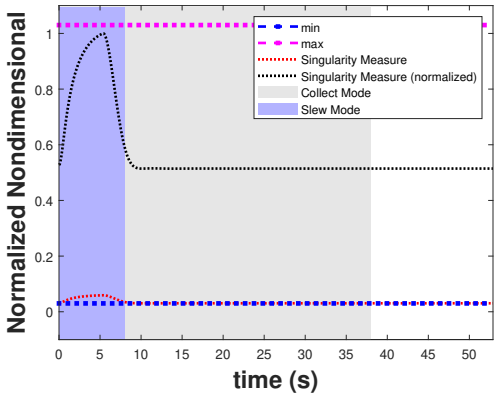
4.2 Problem B Results

Problem B follows the same simulation configuration as Problem A with differences arising in the environment with the use of the PHR spacecraft physical characteristics. This problem seeks to evaluate if the trained agent can be used for any set of spacecraft physical properties if so desired. The trained agent from Problem A is therefore evaluated using the spacecraft physical properties of the PHR spacecraft with more focus on singularity measures.

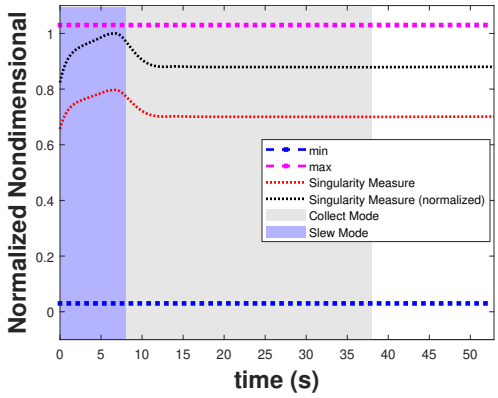
4.2.1 Baseline Results

The baseline results for Problem B use the same simulation environment from Problem A but directs attention to the singularity avoidance capabilities of the RL agent. As outlined in Section 3.1.2, the PHR spacecraft is physically larger than

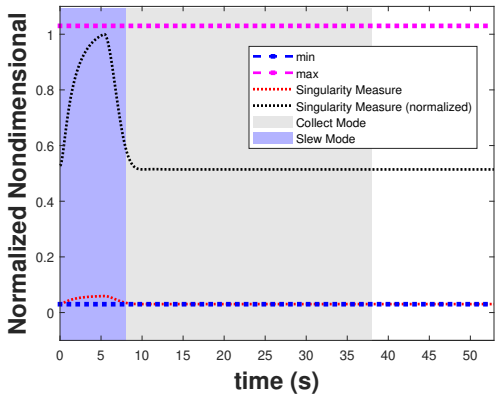
SimSat and employs a larger angular momentum envelope for its CMG array. Problem B seeks to evaluate the impact of a different spacecraft configuration to the trained RL environment. The average singularity measure, number of near singularities, and normalized state values for the singularity measures are described. The baseline singularity profiles are found in Fig. 33 It can be observed according to the shaded regions of the state profiles, slewing occurs from 0 to 8 seconds and collection from 8 seconds to 37.9 seconds.



(a) CSCMG Normalized Singularity Measure



(b) VSCMG Normalized Singularity Measure



(c) RWCMG Normalized Singularity Measure

Figure 33: Baseline singularity measures for each CMG array

4.2.2 Agent Performance Evaluation

A key item of research interest is the impact to singularity avoidance the RL agent may have. This was quantified through the use of the singularity measure m and time series plots of this variable. Ten evaluations of 100 episode runs were conducted for RL agent evaluation as performed in Problem A. The total average reward values after the CSCMG, VSCMG, and RWCMG array evaluations were -114334.242, 993966.564, and -45758102.673 respectively. It can be observed that the total average reward values for all three arrays were closer to zero in value than their Problem A counter parts in Section 4.1.3.

As a larger spacecraft, PHR was anticipated to cause greater variance in values due to its larger torque and angular momentum envelopes. The singularity measure use within the reward formulation signalled importance to the agent and was also used a general performance measure during training. Near-singularity measure values less than 0.03 were undesirable and parameters were adjusted to steer the agent away from such states. As seen in Fig. 34 through 36, the actual, normalized, and desired singularity measure values were plotted for each array. The CSCMG array displayed the closest behavior and numerical values to its baseline values of Fig. 33a compared to the VSCMG and RWCMG arrays. Early episode termination occurred within the first few seconds of the simulation for the VSCMG and RWCMG arrays, creating distinct vertical lines.

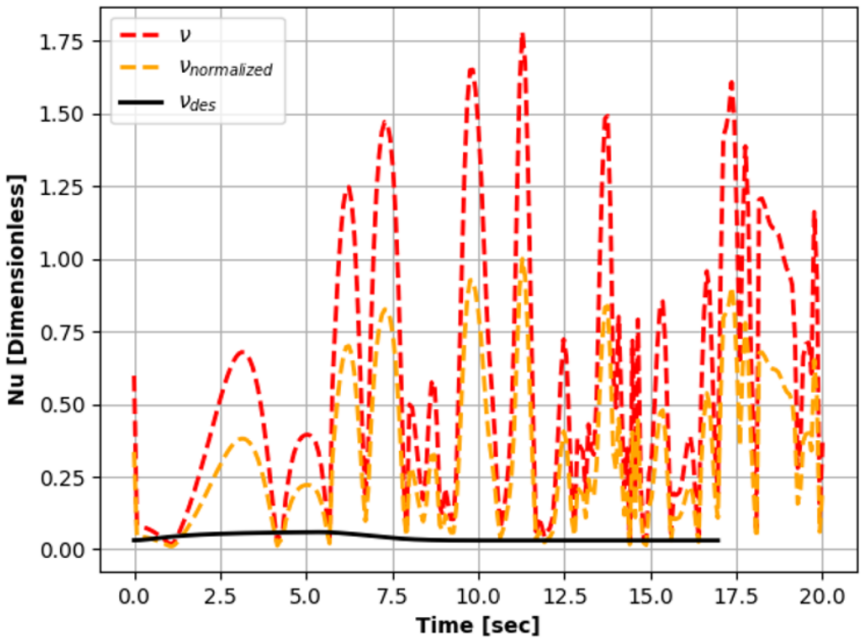


Figure 34: CSCMG Normalized Singularity

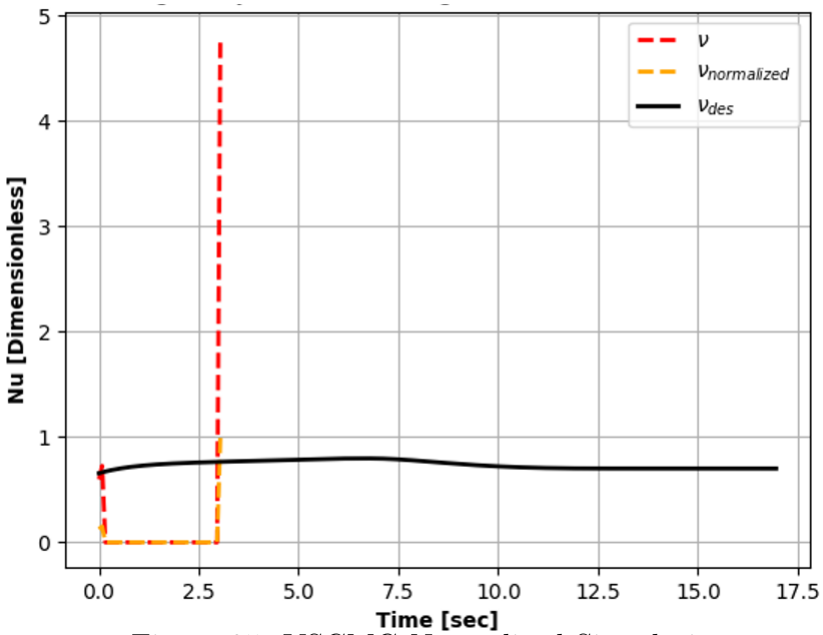


Figure 35: VSCMG Normalized Singularity

As demonstrated in Problem A, using the evaluated reward values is an insufficient

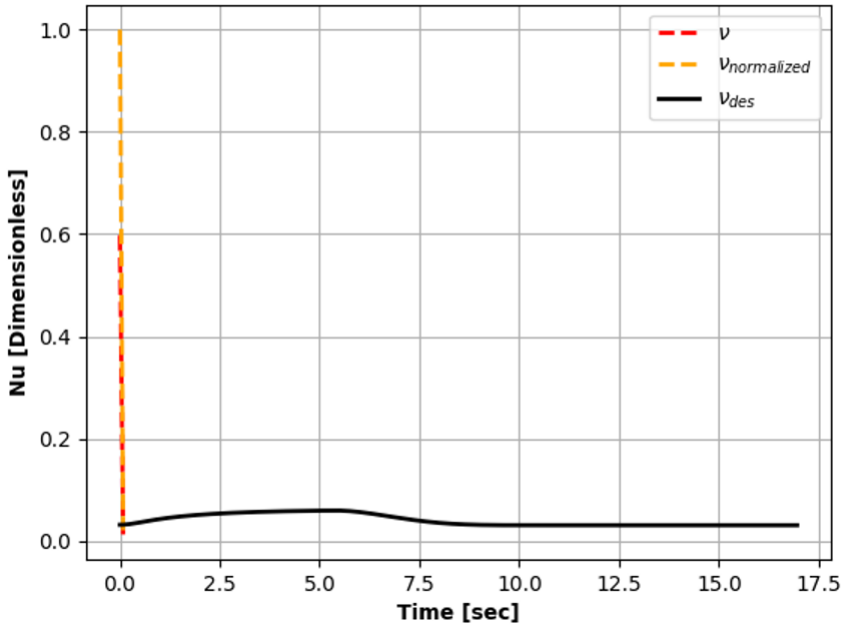


Figure 36: RWCMG Normalized Singularity

single measure of agent performance. To further evaluate the performance measures of Problem B, a similar metric analysis will be performed. Tables 21, 22, and 23 contain metrics of interest for this problem, where the singularity measure is a dimensionless value. From these tables it can be determined that the CSCMG array maintained the trend from Problem A by outperforming the other two arrays in terms of the attitude and angular velocity. With the smallest number of near-singularities and average singularity measure, the RL agent displayed greater success with the CSCMG array. The VSCMG array plot minutely reflects the number of near sinuglarites and the average singularity measure is in gross excess of the maximum value of 1.3. The Euler angle RSS error and RMS errors increased for the CSCMG and VSCMG. For the RWCMG array, the RSS decreased by a factor of 2 or with a percent difference of 99.34 percent and the RMS decreased by 8.46 percent. The change in attitude error for the RWCMG can be attributed to the inclusion of reaction wheels for attitude control and the agent learning to relationships from that actuator type. One near

singularity coupled with a singularity measure exceeding the maximum value, the agent was unable to fully utilize the array for singularity avoidance for the PHR spacecraft.

Table 16: CSCMG Metric Comparisons - Problem B

Metric	QF	RL
Number of Near-Singularities (< 0.03)	0	5
Average Singularity Measure	0.5651	0.587
Total Euler Angle RSS Error (deg)	28.8140	1.73E+10
Total Euler Angle RMS Error (deg)	0.5882	8.30E+03

Table 17: VSCMG Metric Comparisons - Problem B

Metric	QF	RL
Number of Near-Singularities (< 0.03)	0	36
Average Singularity Measure	0.8940	25.889
Total Euler Angle RSS Error (deg)	28.8140	3.42E+08
Total Euler Angle RMS Error (deg)	0.5882	2.96E+03

Table 18: RWCMG Metric Comparisons - Problem B

Metric	QF	RL
Number of Near-Singularities (< 0.03)	0	1
Average Singularity Measure	0.5649	19.21
Total Euler Angle RSS Error (deg)	26.1196	2.82E+06
Total Euler Angle RMS Error (deg)	0.6946	1.19E+03

4.3 General Performance Evaluations

Problem A and B shared some similarities in agent performance and several insights were gathered from the training phase. Goal solution convergence was not realized in either problem and attempting to obtain this could possibly require several weeks worth of simulation run time. The Euler angle errors were consistently large in magnitude among the arrays and the attitude parameterization types had a minimal effect on general performance. The solution uniqueness had the sign values of the Euler angles and quaternions built into the reward functions and the agent was unable to conduct a proper mix of exploration and exploitation to occupy the

vicinity around the goal. Between both problems, the CSCMG displayed the closest goal behavior and values compared to the VSCMG and RWCMG arrays. The most impact to the singularity measure was observed in the VSCMG, where despite its researched uses in conducting improved singularity avoidance compared to SGCMGs, the agent was the most sensitivity to its steering logic. Findings from the results section are formally summarized and further conclusions are drawn within the next section of Chapter V.

V. Conclusions

This chapter outlines the conducted research and discusses the key findings. Each problem will be presented separately before a joint summary is provided. Recommendations for future work are additionally discussed.

5.1 Summary

A simulated spacecraft environment was developed with the intent to serve as a generalized case for various spacecraft ADCS devices and desired attitudes and velocities. Additional variable aspects of the environment include integration schemes, terminal goals, and reward functions. The Twin-Delayed Deep Deterministic Policy Gradient (TD3) was the learning algorithm for the simulation. This research determined a spacecraft attitude control system can incorporate RL elements, where control torques were produced using three CMG arrays. However, desired orientations and angular velocities were not realized as the the RL agent was unable to successfully control the spacecraft to those states. Metric analysis also revealed large discrepancies between the the baseline and RL state profiles and solutions. Similar trends of learned RL agent behavior was observed between two spacecraft classes, where the agent produced solutions closest to baseline values for the CSCMG. The RWCMG results were the next closest and the largest error reductions and percent differences for the pointing error and angular rate were observed. The VSCMG had the least desirable performance among all the arrays, producing unfeasible results. Lastly, mode tracking in the form of detailed slew and collection functions was unable to be accomplished for any CMG array, where a brief expression of this capability was observed during VSCMG array training. Despite the limited success of the RL agent to conduct low-level commanding in the form of producing torque for an actu-

ator set, this research has provided details on the integration of new concepts with traditional methods as performance enhancement and agility is pursued for ADCS of modern spacecraft. The summarized findings for each individual problem will be described next and how they have accomplished the objectives outlined in Section 1.3.1.

5.1.1 Problem A

After training and evaluation, an RL agent was able to produce control for an ADCS using three separate types of CMG arrays. The machine learning technique of reinforcement learning was able to be applied to spacecraft attitude control and control allocation. While target attitude and angular rates were not accomplished due to agent difficulties with attitude dynamics relationships, trends and agent responsiveness to the environment were readily observed. It was hypothesized that generalization among all three arrays would be accomplished by the agent. The CSCMG and RWCMG arrays displayed similar attitude and angular velocity errors compared to the baseline results, partially substantiating this hypothesis. The results produced by the RL agent resembled traditional attitude control results the most for the CSCMG array, despite large errors. The agent was observed to reduce the angular rate at the cost of the attitude and learn state profile behavior for these arrays. Of note was the VSCMG array performance, where training and evaluation result differences were the largest. The agent encountered the most difficulty learning to produce desired control with this array and the steering logic complexity appears to have aided in this avenue. In total, the agent was able to produce some results that resemble a traditional method for attitude control, where attitude parameterization had minimal bearing on agent learning success. The RL agent did not produce the goal attitude and angular rate for any CMG array and the hybridized control system

incurred large errors. Another result of introducing RL into the closed loop were long training times and the agent searching solution space inefficiently. While the reinforcement learning method did not enhance attitude control performance in this research, a quantification method of performance and error was established. Placing an RL agent inside a closed-loop has allowed for the examination of how machine learning methods interacts with traditional control methods.

5.1.2 Problem B

The investigation of the generalized performance properties of the trained agent in Problem A was conducted in a second problem. Problem B examined the application of the trained agent from Problem B to a spacecraft of a different class compared to the SimSat. It was postulated the agent would produce similar control allocation for the PHR spacecraft having learned and applying the dynamic relationships from Problem A. The singularity measure was used a distinct performance metric to evaluate the abilities of the agent. Similar agent behavior and trends from Problem A, with the most notable being the reduction in attitude error for the RWCMG array. The agent produced the best results for the CSCMG array, where the assumption of simplest steering array logic allowed the agent to search the solution space more efficiently. The complexity of the VSCMG steering logic and dynamics was surmised to cause greater difficulty for the agent compared to the RWCMG array and this theory was supported by the array having the largest and number of near-singularities and average singularity measure. Altogether, the agent successfully performed control allocation for a second spacecraft, preserving its learned behavior from Problem A.

5.2 Contributions and Recommendations

It has been determined that an RL agent can successfully produce control for a spacecraft attitude control system with CMG arrays and also a combined RW and CMG configuration. Contributions first include the development of a novel system. This is the first time such a simulation has been developed in conjunction with an ADCS, to the knowledge of the author. This integration allows for the further evaluation of state-of-the-art AI agents inside spacecraft systems and viable control options. This work has also facilitated an examination of theoretical systems and their findings applied to real-world constraints, where the VSCMG and RWCMG arrays generated quantifiable results apart from basic simulations. This research has illustrated the challenges and problem areas of implementing RL into an ADCS and recommendations for improving results and adapting portions of the simulation will be described next. The limited success of the RL agent to conduct low-level commanding in the form of producing torque for an actuator set could be adapted to do a different part of the control allocation. Uncertainty was introduced into the system with the approximation conducted by the RL agent and an investigation of where such an agent can best benefit the system would be beneficial. The precision required of spacecraft missions and the continued application of CMGs can benefit from application of machine learning. Specifically, if the resulting error from any approximation from reinforcement learning is minimized. An additional recommendation is to further scope the problem to a single CMG actuator array and conduct a sensitivity analysis, or examine the requested torque without actuation. This analysis would enable closer examination of the agent's ability for producing control or other desired state values. These recommendations for future researchers are expanded upon in the future work section following this discussion.

5.3 Future Work

The application of RL to spacecraft attitude control is novel when considering conventional attitude control algorithms. With the development of a simplified environment that can be controlled using RL, higher fidelity models are possible given the findings from this research. The addition of environmental disturbances, other physical spacecraft values, and more targets are examples of properties that can be used to add realism to the simulation.

Singularity Avoidance: Another focus of future work would be the investigation of other singularity avoidance and/or escape algorithms and the implications of null motion. The complexity of these concepts was beyond the scope of this research, but future work can observe the ability of the agent to adapt to the variance of these algorithms to feasibly improve performance.

Target Deck Optimization: A second problem area of interest can be added to this research in the form of using optimization during target collection. Sequential collection of multiple targets can benefit from the use of a modified traveling salesman problem [85] to select which targets to visit and maximize the number of targets collected in a given amount of time. This problem would require the use of the environment developed in this research and also a second RL environment to track the states associated with the multiple targets.

Stability Analysis: While not explored in this research, the stability of the closed-loop with the RL controller can be examined to help increase the effectiveness of the controller. An example of an experiment that would explore the stability of the closed loop would be to develop a set of simplified spacecraft attitude control problems and conduct a stability analysis. While there are a limited amount of studies that have conducted rigorous Lyapunov analysis on RL-controlled systems and actor-critic architectures [86, 87], existing literature would benefit from a study derived from this

research.

Computational Efficiency: Lastly, the computational expense of the simulation can be overcome with the use of parallelization and optimization. Achieving global solution convergence would be readily realized with the use of faster computation to speed training and aid designers in diagnosing the system. A converged solution would also enable the use of possible hardware integration.

5.4 Final Remarks

The success of using an RL agent to control an ADCS with CMG is the first of its kind, to the awareness of the author. Various areas and topics of interest can be accomplished using the findings of this research, to include eventual hardware implementation and test by future researchers and mission planners. While the findings of this research facilitate the exploration of a machine learning technique such as reinforcement learning for spacecraft attitude control, such techniques still require additional research and meticulous tuning to realize potentially advantageous control for autonomous spacecraft missions.

Appendix A. Twin Delayed DDPG (TD3) Algorithm

Algorithm 1 Twin Delayed DDPG [82]

Input: initial policy parameters θ, Q-function parameters ϕ_1,ϕ_2, empty replay buffer $\mathcal{D}$

Set target parameters equal to main parameters $\theta_{target} \leftarrow \theta$, $\phi_{target,1} \leftarrow \phi_1$, $\phi_{target,2} \leftarrow \phi_2$

repeat

  Observe state s and select $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$, where $\epsilon \sim \mathcal{N}$

  Execute a in the environment

  Observe next state s', reward r, and done signal d to indicate whether s is terminal

  Store (s,a,r,s',d) in replay buffer $\mathcal{D}$

  If s' is terminal, reset environment state

  if its time to update ***then***

    for j in range(however many updates) ***do***

      Randomly sample a batch of transitions, $B = (s,\ a,\ r,\ s',\ d)$

      Compute target actions

$$a'(s') = \text{clip}\ (\mu_{\theta_{target}}\ (s') + \text{clip}(\epsilon, \text{-c, c}),\ a_{Low}, a_{High})$$

      Compute target

$$\text{y}(r,\ s',\ d\ ) = r + \gamma(1\text{- }d) \min_{i=1,2} Q_{\phi_{target,i}}\ (s', a'(s'))$$

      Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i}\ \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_{\phi_i}(s, a) - y(r, s', d))^2,\ \text{for i} = 1,2$$

      if j mod *policy delay*$= 0$ ***then***

        Update policy by one step of gradient ascent using

$$\nabla_\phi\ \frac{1}{|B|} \sum_{s \in B}$$

        Update target networks with

$$\phi_{target,i} \leftarrow \rho\phi_{target,i} + (1 - \rho)\phi_i$$
$$\phi_{target} \leftarrow \rho\phi_{target} + (1 - \rho)\phi$$

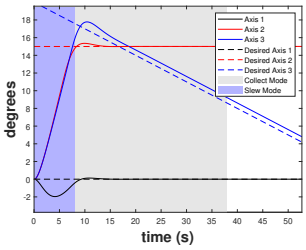
      end if

    end for

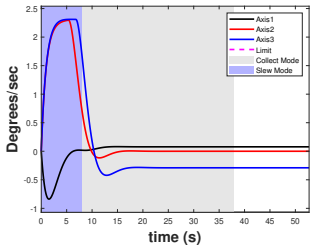
  end if

until convergence

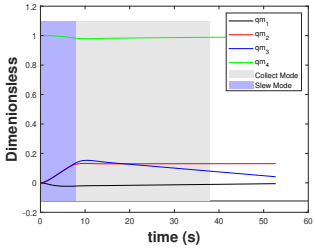
Appendix B. Problem B Baseline State Profiles



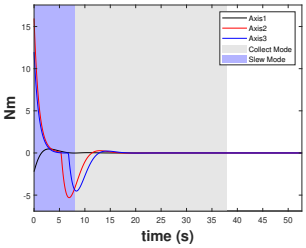
(a) Attitude (Euler Angles)



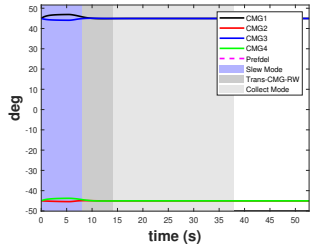
(b) Angular Rate



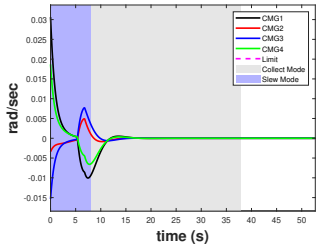
(c) Attitude (Quaternions)



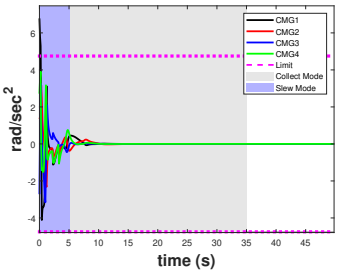
(d) CMG Torque



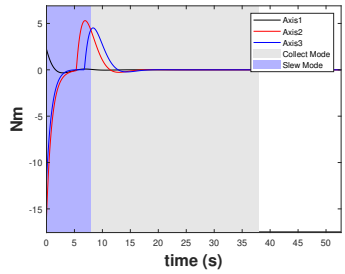
(e) CMG Gimbal Angles



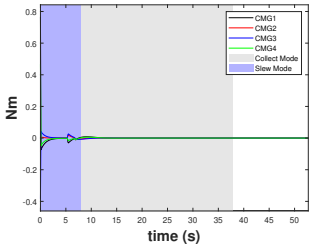
(f) CMG Gimbal Rate



(g) CMG Gimbal Acceleration

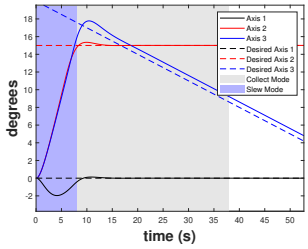


(h) Controller Torque

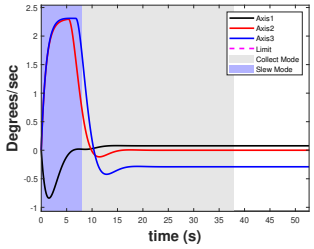


(i) Gimbal Torque Generated

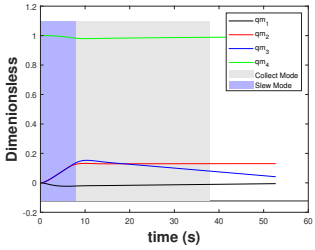
Figure 37: CSCMG Array Baseline Values - PHR



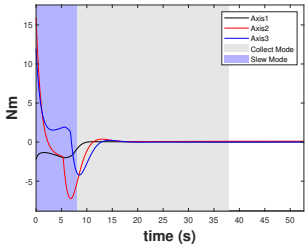
(a) Attitude (Euler Angles)



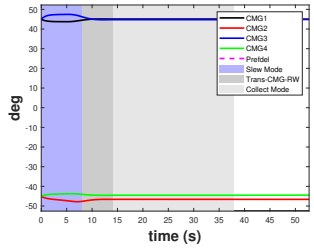
(b) Angular Rate



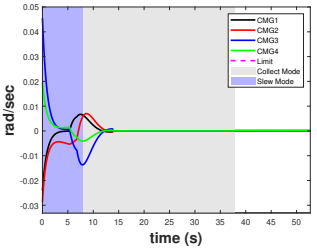
(c) Attitude (Quaternions)



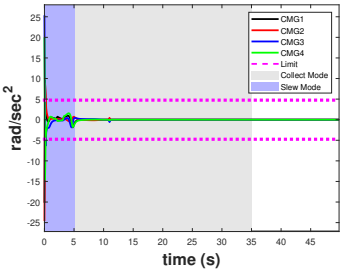
(d) CMG Torque



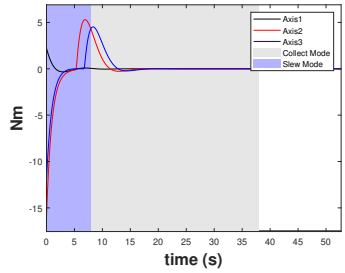
(e) CMG Gimbal Angles



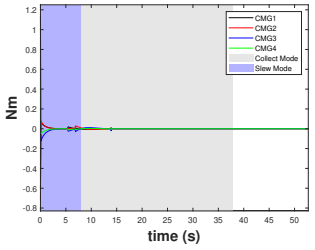
(f) CMG Gimbal Rate



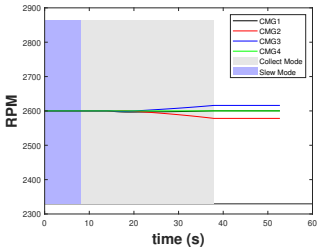
(g) CMG Gimbal Acceleration



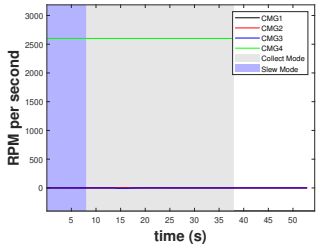
(h) Controller Torque



(i) Gimbal Torque Generated

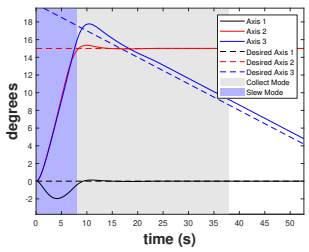


(j) Rotor Spin Velocity

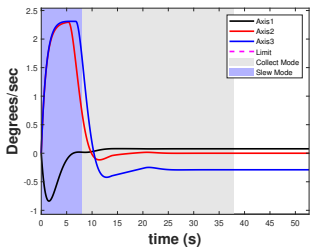


(k) Rotor Spin Acceleration

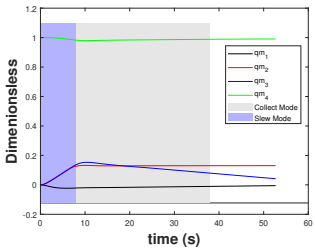
Figure 38: VSCMG Array Baseline Values - PHR



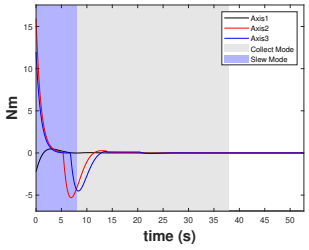
(a) Attitude (Euler Angles)



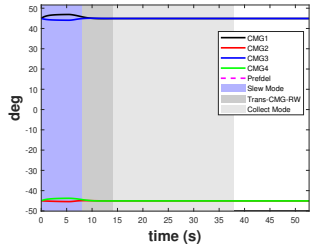
(b) Angular Rate



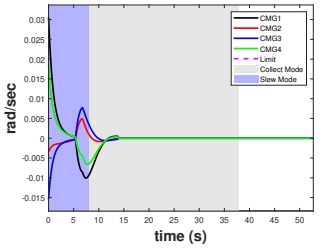
(c) Attitude (Quaternions)



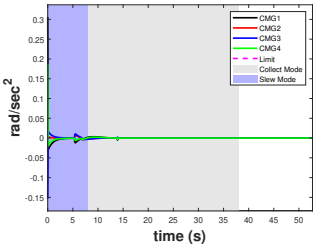
(d) CMG Torque



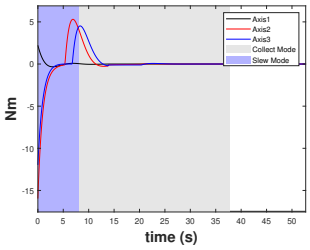
(e) CMG Gimbal Angles



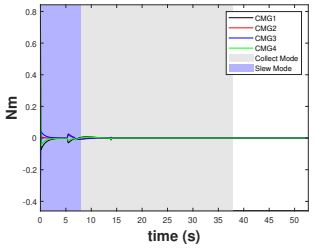
(f) CMG Gimbal Rate



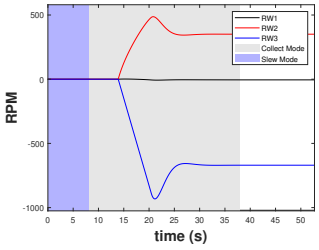
(g) CMG Gimbal Acceleration



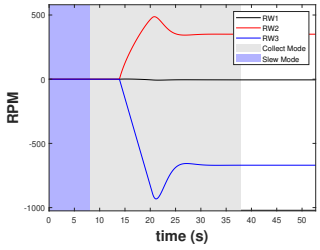
(h) Controller Torque



(i) Gimbal Torque Generated



(j) RW Spin Velocity



(k) RW Spin Acceleration

Figure 39: RWCMG Array Baseline Values - PHR

Appendix C. Initial State Values for SimSat, PHR, and CMG Arrays

Table 19: SimSat Environment State Bounds - CSCMG Array

Parameter	Value
q	[0, 0, 0, 1]
ω	0 radians/sec
δ	[0.7854, -0.7854, 0.7854, -0.7854] rad
$\dot{\delta}$	[2.596 , 0.339, -0.608, 1.649] rad/sec
$\ddot{\delta}$	[25.964, 3.387, -6.083, 16.493] rad/sec^2
$\dot{h_r}$	[-0.173, 0.768, 1.018] N·m
u	[0.173, -0.768, -1.018] N·m
m	0.5977
t	0 sec
dt	0.08 sec
h	0.077 N·m·s

Table 20: SimSat Environment State Bounds - VSCMG Array

Parameter	Value
q	[0, 0, 0, 1]
ω	0 radians/sec
δ	[0.7854, -0.7854, 0.7854, -0.7854] rad
$\dot{\delta}$	[-2 , -2.463, -2, -2] rad/sec
$\ddot{\delta}$	[-50, -50, 50, 50] rad/sec^2
$\dot{\psi}$	[272.271, 272.271, 272.271, 272.271] rad/sec
$\ddot{\psi}$	[-2.114× 10^{-4}, -3.750× 10^{-4}, -3.742× 10^{-4}, -2.087× 10^{-4}] rad/sec^2
$\dot{h_r}$	[-0.173, 0.768, 1.018] N·m
u	[0.173, -0.768, -1.018] N·m
m	0.5977
t	0 sec
dt	0.08
h	0.077 N·m·s

Table 21: SimSat Environment State Bounds - RWCMG Array

Parameter	Value
q	[0, 0, 0, 1]
ω	0 radians/sec
δ	[0.7854, -0.7854, 0.7854, -0.7854] rad
$\dot{\delta}$	[0.7125 , 0.2375, -0.2375, 0.2375] rad/sec
$\ddot{\delta}$	[7.125, 2.375,-2.375, 2.375] rad/sec^2
$\dot{\psi}$	[10^{-9}, 10^{-9}, 10^{-9}, 10^{-9}] rad/sec
$\ddot{\psi}$	[1.296× 10^{-4}, 1.939× 10^{-4}, -6.242× 10^{-4}] rad/sec^2
$\dot{h_r}$	[-0.173, 0.768, 1.018] N·m
u	[0.131, -0.274, -0.208] N·m
m	0.5977
t	0 sec
dt	0.08
h	0.077 N·m·s

Table 22: PHR Environment State Bounds - CSCMG Array

Parameter	Value
q	[0, 0, 0, 1]
ω	0 radians/sec
δ	[0.7854, -0.7854, 0.7854, -0.7854] rad
$\dot{\delta}$	[0.031 , -0.0034, -0.0154, 1.0187] rad/sec
$\ddot{\delta}$	[0.3063, -0.0341, -0.1535, 0.1868] rad/sec^2
$\dot{h_r}$	[-2.2212, 15, 12] N·m
u	[2.2212, -15.9582, -11.9687] N·m
m	0.5977
t	0 sec
dt	0.08 sec
h	0.077 N·m·s

Table 23: PHR Environment State Bounds - VSCMG Array

Parameter	Value
q	[0, 0, 0, 1]
ω	0 radians/sec
δ	[0.7854, -0.7854, 0.7854, -0.7854] rad
$\dot{\delta}$	[-0.0240 , -0.-286, 0.0455, 0.0195] rad/sec
$\ddot{\delta}$	[-0.2401, -0.2861, 0.4548, 0.1954] rad/sec^2
$\dot{\psi}$	[272.271, 272.271, 272.271, 272.271] rad/sec
$\ddot{\psi}$	[-3.2111× 10^{-6}, -5.1323× 10^{-6}, -4.9689× 10^{-6}, -2.5952× 10^{-6}] rad/sec^2
$\dot{h_r}$	[-2.2212, 15, 12] N·m
u	[2.2212, -15.9582, -11.9687] N·m
m	0.5977
t	0 sec
dt	0.08
h	0.077 N·m·s

Table 24: PHR Environment State Bounds - RWCMG Array

Parameter	Value
q	[0, 0, 0, 1]
ω	0 radians/sec
δ	[0.7854, -0.7854, 0.7854, -0.7854] rad
$\dot{\delta}$	[0.0306 , -0.0034, -0.0154, 0.0187] rad/sec
$\ddot{\delta}$	[0.30635, -0.0341, -0.1535, 0.1868] rad/sec^2
$\dot{\psi}$	[10^{-9}, 10^{-9}, 10^{-9}, 10^{-9}] rad/sec
$\ddot{\psi}$	[6.0059× 10−9, 7.2392× 10^{-9}, -1.5306× 10^{-9}] rad/sec^2
$\dot{h_r}$	[-2.2212, 15, 12] N·m
u	[2.2212, -15.9582, -11.9687] N·m
m	0.5977
t	0 sec
dt	0.08
h	0.077 N·m·s

Appendix D. Reward Function Formulations

Algorithm 2 CSCMG Array Reward Function

procedure REWARD(s) ▷ Reward values r are generated using state values s
  $R \leftarrow r_n \in r$ ▷ The total reward is R
  if time $\geq$ maximum time **then** ▷ Sim runtime dictates reward value
    if Δ_{EA_1} **then**
      $r_1 =$ Negative value
    if Δ_{EA_2} **then**
      $r_2 =$ Negative value
    if Δ_{EA_3} **then**
      $r_3 =$ Negative value
    $r_4 = (1 - e^{\Delta_{EA_3} dt})\alpha$
    $r_5 = (1 - e^{\omega_{error} dt})$
    $r_6 = e^{q_{e,t} dt}$
    Episode termination
    $r_7 =$ Negative value
  if $t_{num} \geq 2$ **and** $\Delta_{EA_t} < \Delta_{EA_{t-1}}$ **then** ▷ Constraint violation checks
    $r_8 =$ Positive value
    if sign(EA_{meas}) = sign(EA_{des}) **then**
      $r_9 =$ Positive value
    else
      $r_{10} =$ Negative value
    if sign($quat_{meas}$) = sign($quat_{des}$) **then**
      $r_{11} =$ Positive value
    else
      $r_{12} =$ Negative value
  else if $t_{num} \geq 2$ **and** $\Delta_{EA_t} > \Delta_{EA_{t-1}}$ **then**
    $r_{13} =$ Negative value
    if gimbal angle constraint violation **then**
      Episode termination
      $r_{14} =$ Negative value
    if gimbal rate/acc and angular rate constraint violation **then**
      $r_{15-17} =$ Negative value
  if $\Delta_{EA} \leq 2^\circ$ **then**
    $r_{18} =$ Large positive value ▷ The goal has been reached
  $R = \sum_{n=1}^{18} r_n$

  return R ▷ The total reward is generated

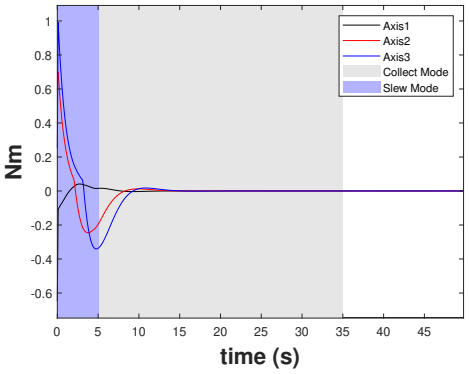
Algorithm 3 VSCMG Array Reward Function

procedure REWARD(s) ▷ Reward values r are generated using state values s
  $R \leftarrow r_n \in r$ ▷ The total reward is R
  if time $\geq$ maximum time **then** ▷ Sim runtime dictates reward value
    if Δ_{EA_1} **then**
      $r_1 =$ Negative value
    if Δ_{EA_2} **then**
      $r_2 =$ Negative value
    if Δ_{EA_3} **then**
      $r_3 =$ Negative value
    $r_4 = (1 - e^{\Delta_{EA_3} dt})\alpha$
    $r_5 = (1 - e^{\omega_{error} dt})$
    $r_6 = e^{q_{e,t} dt}$
    if $m_t \geq m_{max}$ **then**
      Episode termination
      $r_7 =$ Negative value
    if $t_{num} \geq 2$ **and** $\Delta_{EA_t} < \Delta_{EA_{t-1}}$ **then** ▷ Constraint violation checks
      $r_8 =$ Positive value
    if $t_{num} \geq 2$ **and** $\Delta_{EA_t} > \Delta_{EA_{t-1}}$ **then**
      $r_9 =$ Negative value
    if sign(EA_{meas}) = sign(EA_{des}) **then**
      $r_{10} =$ Positive value
    else
      $r_{11} =$ Negative value
    if sign($quat_{meas}$) = sign($quat_{des}$) **then**
      $r_{12} =$ Positive value
    else
      $r_{13} =$ Negative value
    if gim angle/rate/acc, ang rate, and rotor rate/acc violations **then**
      $r_{14-20} =$ Negative value
    if $\Delta_{EA} \leq 2^\circ$ **then**
      $r_{21} =$ Large positive value ▷ The goal has been reached
    $R = \sum_{n=1}^{21} r_n$
  return R ▷ The total reward is generated

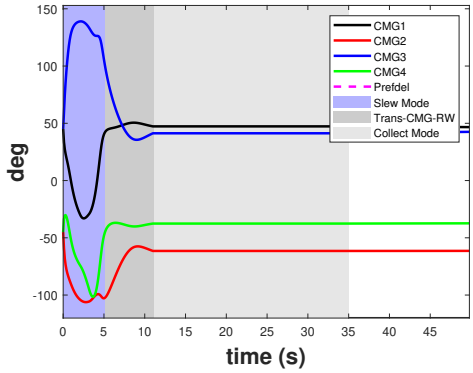
Algorithm 4 RWCMG Array Reward Function

1: **procedure** REWARD(s) ▷ Reward values r are generated using state values s
2: $R \leftarrow r_n \in r$ ▷ The total reward is R
3: **if** time $\geq$ maximum time **then** ▷ Sim runtime dictates reward value
4: **if** Δ_{EA_1} **then**
5: $r_1 =$ Negative value
6: **if** Δ_{EA_2} **then**
7: $r_2 =$ Negative value
8: **if** Δ_{EA_3} **then**
9: $r_3 =$ Negative value
10: $r_4 = (1 - e^{\Delta_{EA_3} dt})\alpha$
11: $r_5 = (1 - e^{\omega_{error} dt})$
12: $r_6 = e^{q_{e,t} dt}$
13: **if** $m_t \geq m_{max}$ **or** $m_t \leq m_{min}$ **then**
14: Episode termination
15: $r_7 =$ Negative value
16: **if** $t_{num} \geq 2$ **and** $\Delta_{EA_t} < \Delta_{EA_{t-1}}$ **then** ▷ Constraint violation checks
17: $r_8 =$ Positive value
18: **if** sign(EA_{meas}) = sign(EA_{des}) **then**
19: $r_9 =$ Positive value
20: **else**
21: $r_{10} =$ Negative value
22: **if** sign($quat_{meas}$) = sign($quat_{des}$) **then**
23: $r_{11} =$ Positive value
24: **else**
25: $r_{12} =$ Negative value
26: **else if** $t_{num} \geq 2$ **and** $\Delta_{EA_t} > \Delta_{EA_{t-1}}$ **then**
27: $r_{13} =$ Negative value
28: **if** gimbal angle constraint violation **then**
29: Episode termination
30: $r_{14} =$ Negative value
31: **if** gimbal rate/acc and angular rate constraint violation **then**
32: $r_{15-17} =$ Negative value
33: **if** $\Delta_{EA} \leq 2^\circ$ **then**
34: $r_{18} =$ Large positive value ▷ The goal has been reached
35: $R = \sum_{n=1}^{18} r_n$
36: **return** R ▷ The total reward is generated

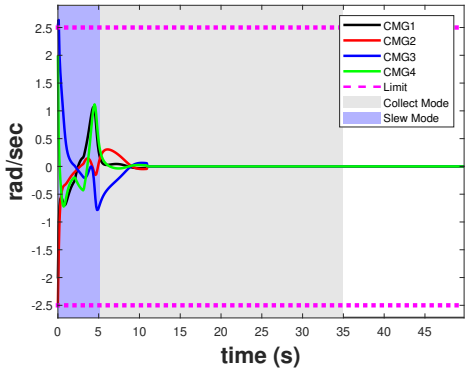
Appendix E. Problem A VSCMG and RWCMG Baseline Results



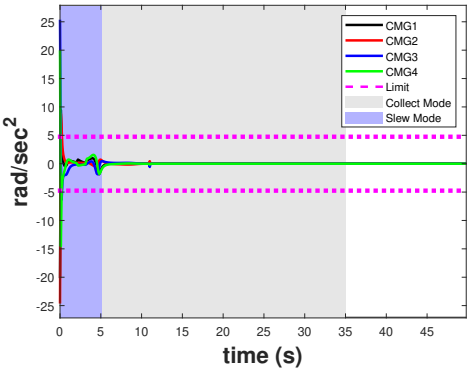
(a) CMG Torque



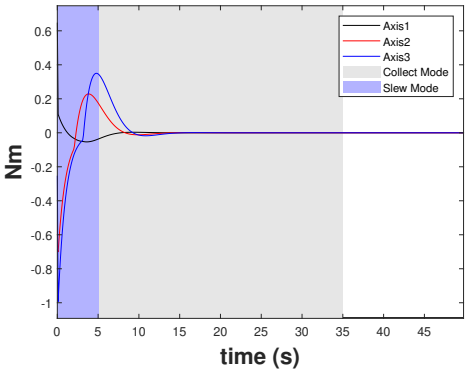
(b) CMG Gimbal Angles



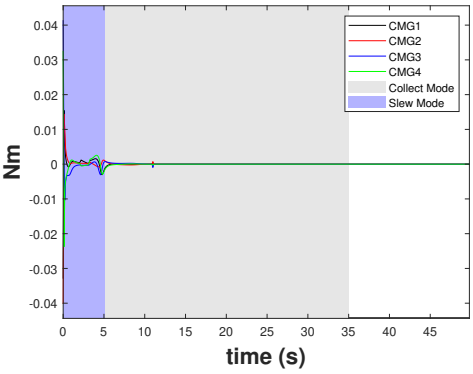
(c) CMG Gimbal Rate



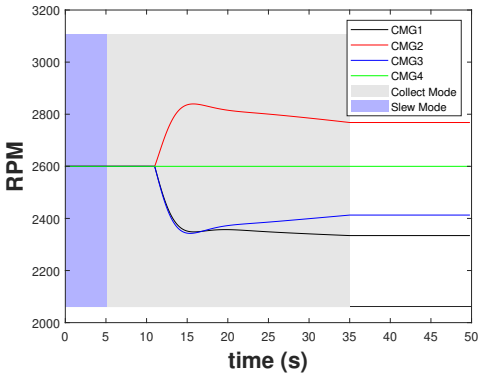
(d) CMG Gimbal Acceleration



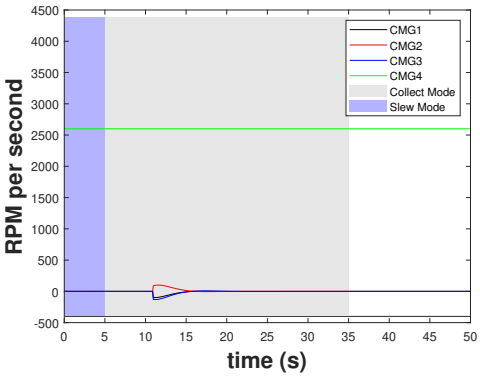
(f) Controller Torque



(g) Gimbal Torque Generated

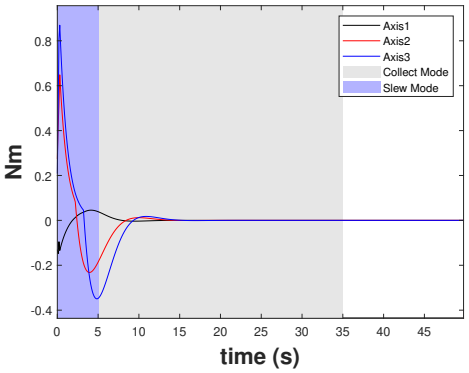


(h) Rotor Spin Velocity

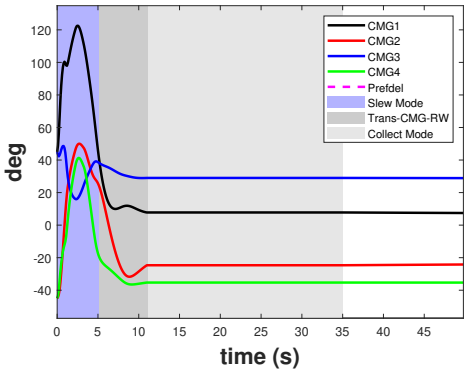


(i) Rotor Spin Acceleration

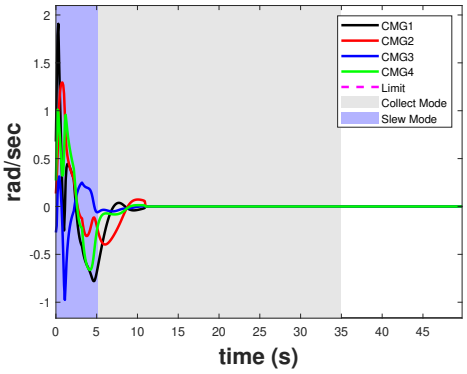
Figure 40: VSCMG array baseline values



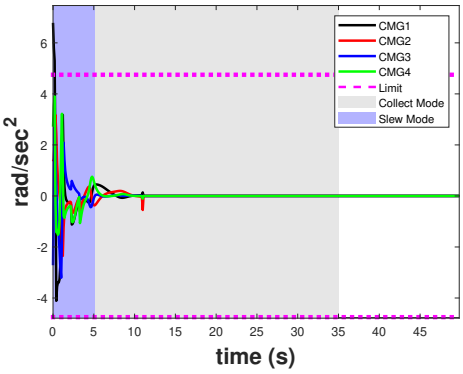
(a) CMG Torque



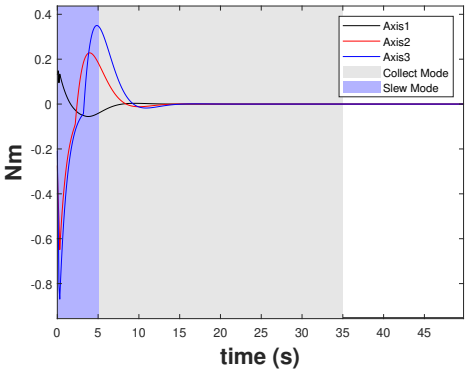
(b) CMG Gimbal Angles



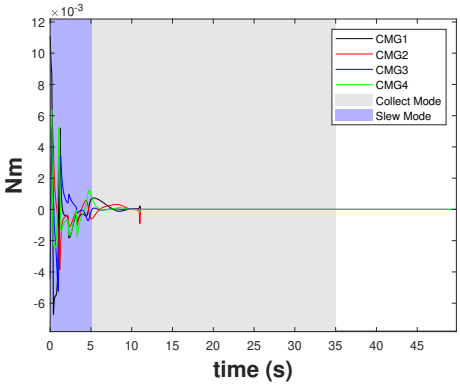
(c) CMG Gimbal Rate



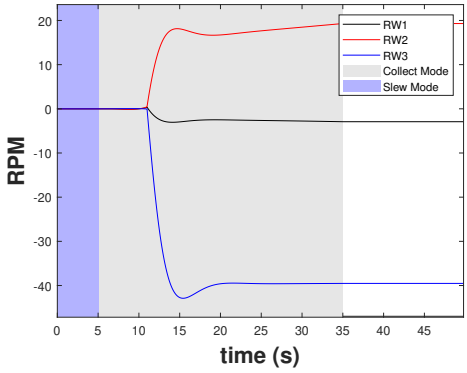
(d) CMG Gimbal Acceleration



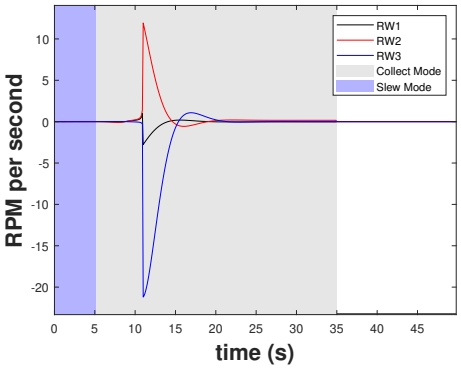
(f) Controller Torque



(g) Gimbal Torque Generated



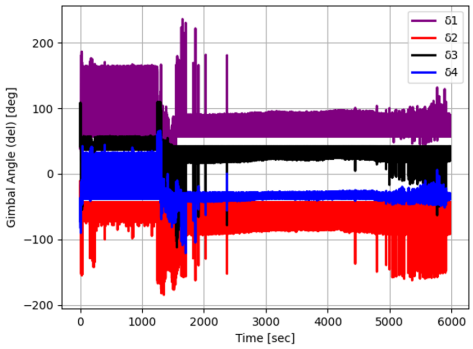
(h) RW Spin Velocity



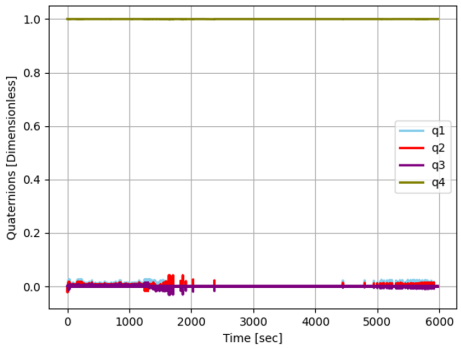
(i) RW Spin Acceleration

Figure 41: RWCMG array baseline values

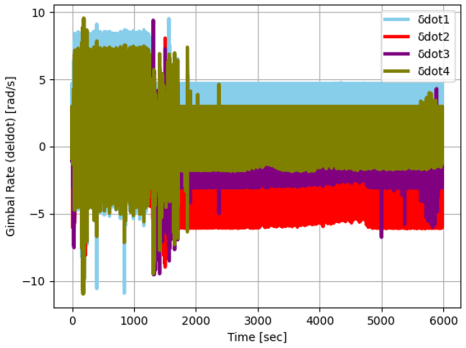
Appendix F. Problem A RL Training Results - All CMG Arrays



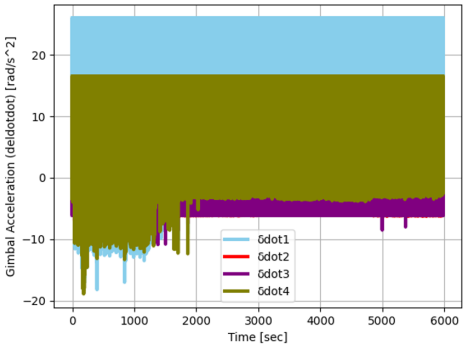
(a) CMG Gimbal Angles



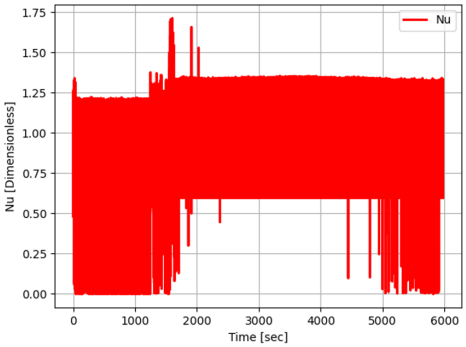
(b) Attitude (Quaternions)



(c) CMG Gimbal Rate

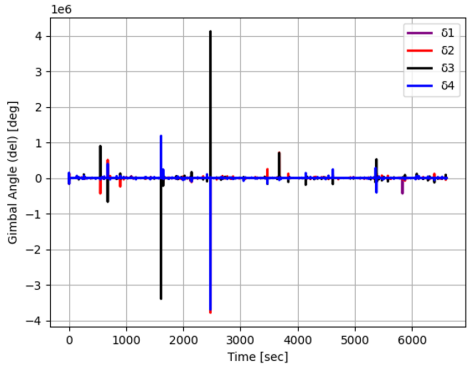


(d) CMG Gimbal Acceleration

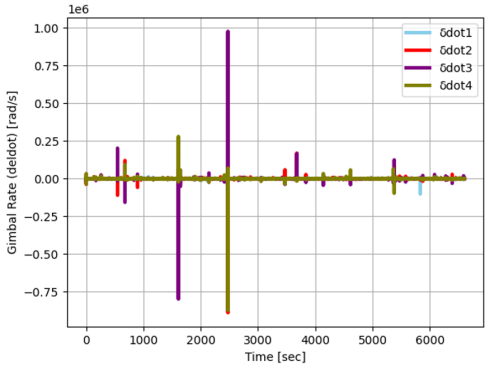


(e) Singularity Measure

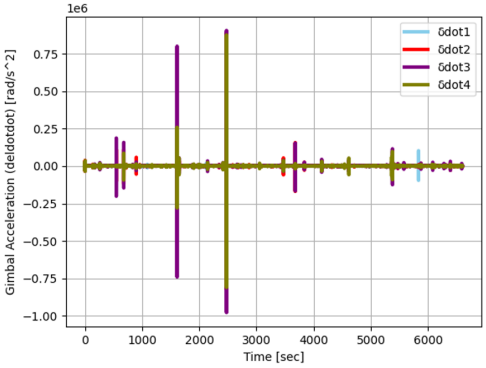
Figure 42: CSCMG Array Training Values



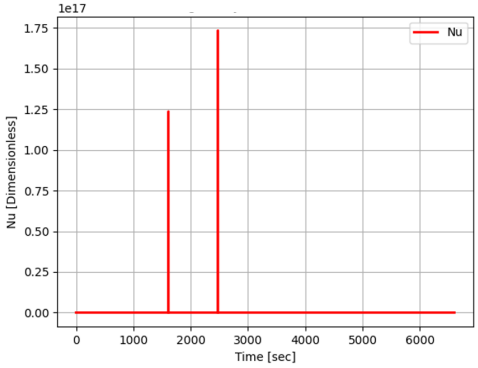
(a) CMG Gimbal Angles



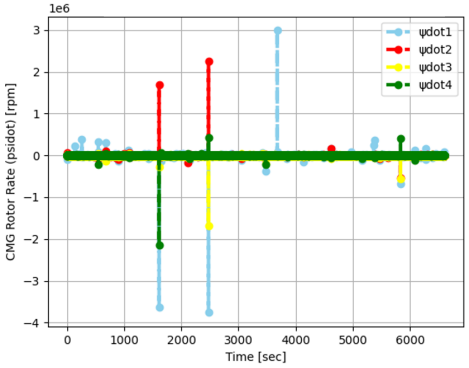
(b) CMG Gimbal Rate



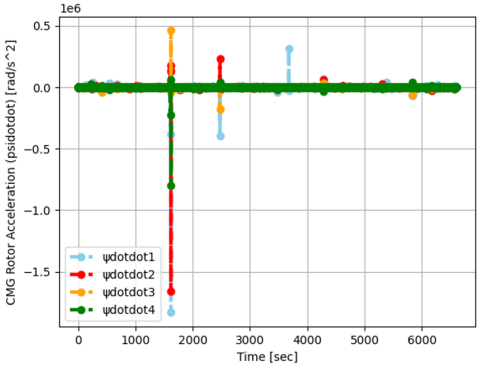
(c) CMG Gimbal Acceleration



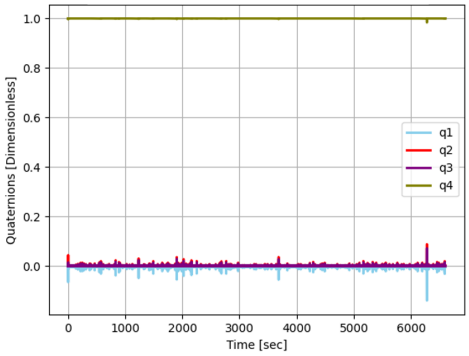
(d) Gimbal Torque Generated



(e) Rotor Spin Velocity

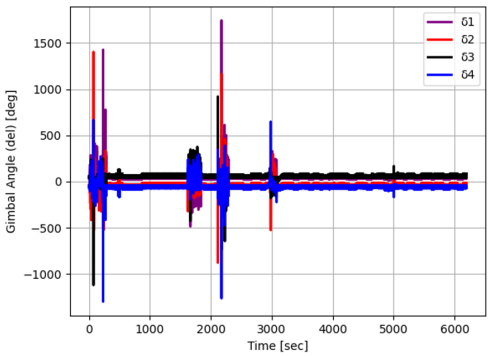


(f) Rotor Spin Acceleration

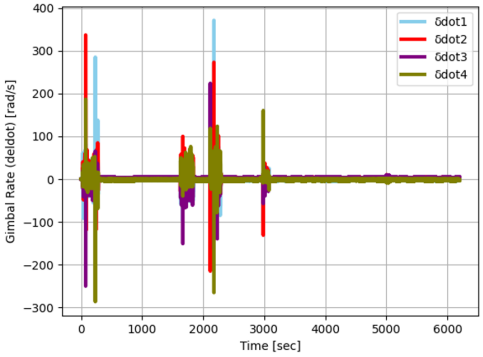


(g) Attitude (Quaternions)

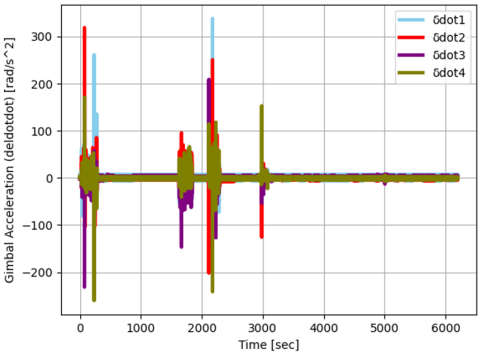
Figure 43: VSCMG Array Training Values



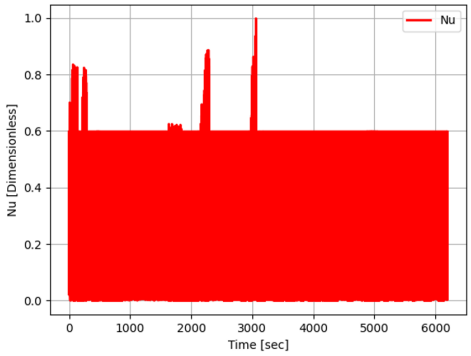
(a) CMG Gimbal Angles



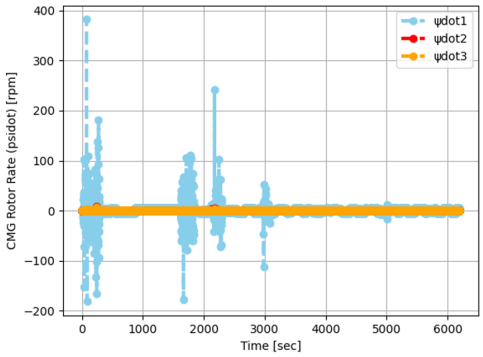
(b) CMG Gimbal Rate



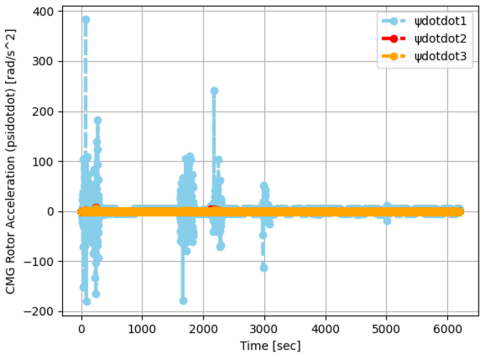
(c) CMG Gimbal Acceleration



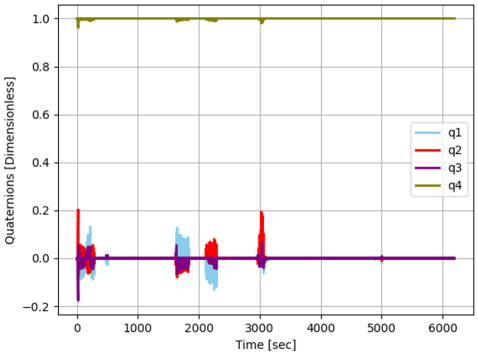
(d) Gimbal Torque Generated



(e) Rotor Spin Velocity



(f) Rotor Spin Acceleration



(g) Attitude (Quaternions)

Figure 44: RWCMG Array Training Values

Bibliography

1. Y. Wu, F. Han, B. Hua, Z. Chen, D. Xu, and L. Ge. Review: A survey of single gimbal control moment gyroscope for agile spacecraft attitude control. *Journal of Harbin Institute of Technology (New Series)*, 25:22–45, 12 2018.

2. Cole Doupe and Eric Swenson. Optimal attitude control of agile spacecraft using combined reaction wheel and control moment gyroscope arrays. 01 2016.

3. Bong Wie. *Space Vehicle Dynamics and Control*. American Institute of Aeronautics and Astronautics, Reston, VA, 2nd edition, 2008.

4. Frederick Leve. Novel steering and control algorithms for single-gimbal control moment gyroscopes. 08 2010.

5. Jae Jun Kim. "Matlab script: singular_surface_4cmg m-file". 2005.

6. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

7. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

8. OpenAI Spinning Up. Kinds of algorithms. Online, 2020.

9. Víctor Uc-Cetina. A novel reinforcement learning architecture for continuous state and action spaces. *Advances in Artificial Intelligence*, 2013, 04 2013.

10. B. N. Biswas, S. Chatterjee, S. P. Mukherjee, and S. Pal. A discussion on euler method: A review. 2013.

11. ESA Earth Observation Portal. Pleiades-hr (high-resolution optical imaging constellation of cnes). Online, 2020.

12. Nathan Weinberg. *Computers in the information society*. Routledge, 2018.

13. G. E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter*, 11(3):33–35, 2006.

14. Tor Johansen and Thor Fossen. Control allocation—a survey. *Automatica*, 49:1087–1103, 05 2013.

15. Dario Izzo, Marcus Märtens, and Binfeng Pan. A survey on artificial intelligence trends in spacecraft guidance dynamics and control. 12 2018.

16. Francois Chollet. *Deep Learning with Python*. Manning Publications Co., USA, 1st edition, 2017.

17. Robert Hoehndorf and Núria Queralt-Rosinach. Data science and symbolic ai: Synergies, challenges and opportunities. *Data Science*, pages 1–12, 06 2017.

18. Quanming Yao, Mengshuo Wang, Hugo Jair Escalante, Isabelle Guyon, Yi-Qi Hu, Yu-Feng Li, Wei-Wei Tu, Qiang Yang, and Yang Yu. Taking human out of learning applications: A survey on automated machine learning. *CoRR*, abs/1810.13306, 2018.

19. Sumant Sharma and Simone D'Amico. Pose estimation for non-cooperative rendezvous using neural networks. *CoRR*, abs/1906.09868, 2019.

20. G.D'Angelo, M.Tipaldi, L. Glielmo, and S.Rampone. Spacecraft autonomy modeled via markov decision process and associative rule-based ma-

chine learning. In *2017 IEEE International Workshop on Metrology for AeroSpace(MetroAeroSpace)*, pages 324–329, 2017.

21. Cole B. George. Optimal and robust neural network controllers for proximal spacecraft maneuvers. 2019.

22. V.Vedant and Alexander Ghosh. Dynamic programming based attitude trajectories for underactuated control systems. In Cheryl, A. H. Walker, editor, *Guidance,navgiation, and control, 2018*, Advances in the Astronautical Sciences, pages 191–202. Univerlt Inc., January 2018.

23. Robert E. Roberson. Two Decades of Spacecraft Attitude Control. *Journal of Guidance Control Dynamics*, 2(1):3–8, January 1979.

24. Jeff B. Burl. *Linear Optimal Control: H(2) and H (Infinity) Methods*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 1998.

25. Hanspeter Schaub and John Junkins. *Analytical Mechanics of Space Systems, Fourth Edition*. 07 2018.

26. Jay Mcmahon and Hanspeter Schaub. Simplified singularity avoidance using variable-speed control moment gyroscope null motion. *Journal of Guidance Control and Dynamics*, 32:1938–1943, 11 2009.

27. Leonhard Euler. General formulas for the translation of an arbitrary rigid body. Translation of original text published by Johan Sten.

28. Charles Gurrisi, R. Seidel, S. Dickerson, S. Didziulis, peter. p. frantz, and K. Ferguson. Space station control moment gyroscope lessons learned. 2010.

29. Katsuhiko Ogata. *Modern Control Engineering*. Prentice Hall PTR, USA, 4th edition, 2001.

30. G. Ellis. Observers in control systems: A practical guide. 2002.

31. Bong Wie, David Bailey, and Christopher Heiberg. Rapid multitarget acquisition and pointing control of agile spacecraft. *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, 25:96–104, 01 2002.

32. Bong Wie, David Bailey, and Christopher Heiberg. Singularity robust steering logic for redundant single-gimbal control moment gyros. *Journal of Guidance Control and Dynamics*, 24:865–872, 09 2001.

33. Hanspeter Schaub and John Junkins. Singularity avoidance using null motion and variable-speed control moment gyros. *Journal of Guidance Control and Dynamics*, 23:11–16, 01 2000.

34. Hyungjoo Yoon and Panagiotis Tsiotras. Singularity analysis of variable speed control moment gyros. *Journal of Guidance Control and Dynamics*, 27:374–386, 05 2004.

35. Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

36. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

37. Burr Settles. Active learning literature survey. 07 2010.

38. Muhammad Iqbal and Zhu Yan. Supervised machine learning approaches: A survey. *International Journal of Soft Computing*, 5:946–952, 04 2015.

39. Steven Umbrello. Ai winter. In Michael Klein and Philip Frana, editors, *Encyclopedia of Artificial Intelligence: The Past, Present, and Future of AI*. Santa Barbara, USA: ABC-CLIO, forthcoming.

40. Carver Mead. *Analog VLSI and Neural Systems*. Addison-Wesley Longman Publishing Co., Inc., USA, 1989.

41. Dave Anderson and George McNeil. Artificial neural networks technology. Kaman Sciences Corporation, USA, 1992.

42. R. Sutton. Integrated modeling and control based on reinforcement learning. In *NIPS*, 1990.

43. S. Ravichandiran. *Deep Reinforcement Learning with Python - Second Edition*. Expert insight. Packt Publishing, 2020.

44. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.

45. S. Bhasin. Reinforcement learning and optimal control methods for uncertain nonlinear systems. 2011.

46. Christopher Watkins. Learning from delayed rewards. 01 1989.

47. Christopher Watkins and Peter Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 05 1992.

48. Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, 09 2015.

49. Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. *CoRR*, abs/1604.06778, 2016.

50. David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. *31st International Conference on Machine Learning, ICML 2014*, 1, 06 2014.

51. D. Kirk. Optimal control theory : an introduction. 1970.

52. RICHARD E. BELLMAN and STUART E. DREYFUS. *Applied Dynamic Programming*. Princeton University Press, 1962.

53. D. P. Bertsekas. Reinforcement learning and optimal control by. 2019.

54. Hilbert Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. *Cooperative Behavior in Neural Systems Volume 887 of American Institute of Physics Conference Series*, 02 2007.

55. Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning, 2017.

56. Draguna Vrabie, Kyriakos G. Vamvoudakis, and Frank Lewis. *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*. 01 2012.

57. Jacob Broida and Richard Linares. Spacecraft rendezvous guidance in cluttered environments via reinforcement learning. 01 2019.

58. Daniel Miller and Richard Linares. Low-thrust optimal control via reinforcement learning. 01 2019.

59. Kirk Hovell and Steve Ulrich. On deep reinforcement learning for spacecraft guidance. 01 2020.

60. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. 2017.

61. James Allison, Matthew West, Alexander Ghosh, and Fnu Vedant. Reinforcement learning for spacecraft attitude control. 10 2019.

62. Daniel Wierstra. A study in direct policy search. 10 2020.

63. Stefan Willis, Dario Izzo, and Daniel Hennes. Reinforcement learning for spacecraft maneuvering near small bodies. pages 16–277. American Institute ofAeronautics and Astronautics, 02 2016.

64. Brian Gaudet and Roberto Furfaro. Robust spacecraft hovering near small bodies in environments with unknown dynamics using reinforcement learning. 08 2012.

65. Ch Bishop. *Neural Networks For Pattern Recognition*, volume 227. 01 2005.

66. L.H. Graesser and W.L. Keng. *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*. Addison-Wesley Data & Analytics. Addison Wesley, 2019.

67. Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *CoRR*, abs/1811.03378, 2018.

68. Melanie Coggan. Exploration and exploitation in reinforcement learning. 2004.

69. Jason Smith. Attitude model of a reaction wheel/fixed thruster based satellite using telemetry data. 01 2005.

70. Jonathan Wright. *Advancements of In-Flight Mass Moment of Inertia and Structural Deflection Algorithms for Satellite Attitude Simulators*. PhD thesis, 03 2015.

71. S. Johnson. Design of a control moment gyroscope attitude actuation system for the attitude control subsystem proving ground. 2013.

72. Honeywell Aerospace. Momentum control systems, 2020.

73. M. Drusch, U. Del Bello, S. Carlier, O. Colin, V. Fernandez, F. Gascon, B. Hoersch, C. Isola, P. Laberinti, P. Martimort, A. Meygret, F. Spoto, O. Sy, F. Marchese, and P. Bargellini. Sentinel-2: Esa's optical high-resolution mission for gmes operational services. *Remote Sensing of Environment*, 120:25 – 36, 2012. The Sentinel Missions - New Opportunities for Science.

74. Bradford Space. Reaction wheel unit. Online, 2020.

75. Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

76. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

77. Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

78. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

79. Wah Loon Keng and Michael Schock. Awesome deep rl. `https://github.com/kengz/awesome-deep-rl#readme`, 2019.

80. Delin Tan and Zheng Chen. On a general formula of fourth order runge-kutta method. 2012.

81. S. Vadali, S. Walker, and Hwa-Suk Oh. Preferred gimbal angles for single gimbal control moment gyros. *Journal of Guidance Control Dynamics*, 13:1090–1095, 11 1990.

82. Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477, 2018.

83. Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1582–1591, 2018.

84. Jacob Elkins, Rohan Sood, and Clemens Rumpf. Autonomous spacecraft attitude control using deep reinforcement learning. 10 2020.

85. Karla Hoffman and Manfred Padberg. *Traveling salesman problem*. 01 2001.

86. Yinlam Chow, Ofir Nachum, Edgar A. Duéñez-Guzmán, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *CoRR*, abs/1805.07708, 2018.

87. A. Kumar and R. Sharma. A stable lyapunov constrained reinforcement learning based neural controller for non linear systems. In *International Conference on Computing, Communication Automation*, pages 185–189, 2015.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
03–03–2021	Master's Thesis	Aug 2019 — Mar 2021

4. TITLE AND SUBTITLE

Hybridized Spacecraft Attitude Control via
Reinforcement Learning using Control Moment Gyroscope Arrays

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Agu, Cecily, C., 1Lt, USSF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

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

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENY-MS-21-M-328

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

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

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

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Machine learning techniques in the form of reinforcement learning (RL) can solve complex nonlinear problems found within spacecraft attitude determination and control systems (ADCS). Three CMG arrays were implemented in two simulated spacecraft environments using a reinforcement learning controller. The performance of the controllers were evaluated using target profiles from traditional control law implementations, singularity measure, and variable initial state values. The current research demonstrates that while RL techniques can be implemented, further exploration is needed to investigate the operational efficacy of an approach for producing comparable performance attributes with respect to traditional control laws.

15. SUBJECT TERMS

Spacecraft Attitude Control; Deep Reinforcement Learning; Control Moment Gyroscopes; Singularity Avoidance

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Major Joshuah A. Hess, AFIT/ENG
U	U	U	UU	148	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, ext 4713; joshuah.hess@afit.edu

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