Space Environment Modelling and Torque-Optimal Guidance for CubeSat Applications

by

Siddharth S. Kedare, **B.Sc.**

A thesis submitted to the

Faculty of Graduate and Postdoctoral Affairs

in partial fulfillment of the requirements for the degree of

Master of Applied Science in Aerospace Engineering

Ottawa-Carleton Institute for Mechanical and Aerospace Engineering

Department of Mechanical and Aerospace Engineering

Carleton University

Ottawa, Ontario

December, 2014

The undersigned hereby recommends to the

Faculty of Graduate and Postdoctoral Affairs

acceptance of the thesis

Space Environment Modelling and Torque-Optimal Guidance for CubeSat Applications

submitted by **Siddharth S. Kedare**, **B.Sc.**

in partial fulfillment of the requirements for the degree of

Master of Applied Science in Aerospace Engineering

Professor Steve Ulrich, Thesis Supervisor

Professor Metin Yaras, Chair,
Department of Mechanical and Aerospace Engineering

Ottawa-Carleton Institute for Mechanical and Aerosapce Engineering

Department of Mechanical and Aerospace Engineering

Carleton University

December, 2014

Abstract

CubeSats and nano-satellites provide flexible low-cost platforms for the academic and scientific communities to conduct cutting-edge research in the harsh environment of space. The mission life of nano-satellites is often limited by the attitude actuators, and it is therefore beneficial to reduce torque and angular momentum usage during reorientation maneuvers.

In this capacity, a computationally lightweight torque-optimal guidance algorithm was formulated, solved using pseudospectral methods, and validated in a MATLAB-Simulink environment. A low-computation atmospheric density model, developed in support of this research, was extensively validated via performance assessment of passive CubeSat aerostabilization.

Results indicate that this torque-optimal guidance algorithm demonstrates substantial improvements in performance and pointing accuracy over an Eigenaxis controller for similar maneuvers, with low to moderate computational overhead. In doing so, it presents a significant advancement towards the development of intelligent GN&C systems for small satellites.

In memory of Columbia *and* Challenger.

This research is dedicated to those who have taught me that true happiness in life is not about reaching a destination, but in making memories from the adventures and challenges during the journey towards it.

Sic itur ad astra

Acknowledgments

I am deeply grateful to a number of individuals for their involvement in this research. First and foremost, I would like to thank my thesis supervisor, Professor Steve Ulrich, for providing me with this incredible opportunity to pursue research on a topic in my field of interest. His steadfast guidance, enthusiasm, and a shared love for the cosmos was invaluable throughout this undertaking.

My thanks to Dr. Anil V. Rao, and his team for their insights into the internal workings and development of GPOPS. I also express my gratitude to Dr. Philip Gill at the University of California, San Diego for his assistance in obtaining the SNOPT solver.

I also wish to recognize the unwavering support of all of my friends and colleagues during this endeavour. Through thick and thin, you kept my mind open, my heart steady, and my aim true. My greatest thanks go out to my parents for their unyielding love, support, and encouragement, without which this accomplishment would be naught but a distant possibility.

– *Siddharth S. Kedare*

Table of Contents

List of Tables

List of Figures

Nomenclature

List of Acronyms

AAS	American Astronautical Society
ACS	Attitude Control System
AIAA	American Institute of Aeronautics and Astronautics
BCI	Body-Centered Inertial reference frame
CATIA	Computer Aided Three-dimensional Interactive Application
CIRA	COSPAR International Reference Atmosphere
CMG	Control Moment Gyroscope
CMM	Centralized Momentum Management
COSPAR	Committee on Space Research
CPU	Central Processing Unit
CSDS-1	CubeSat DragSail-1
DCM	Direction Cosine Matrix
DRDC	Defence Research and Development Canada
ECI	Earth-Centered Inertial reference frame
ELEO	Equatorial Low Earth Orbit
GFLOPS	Giga Floating Point Operations Per Second
GN&C	Guidance, Navigation, and Control
GPOPS	General Pseudospectral Optimal Software
GPU	Graphics Processing Unit
IADC	Inter-Agency Space Debris Coordination Committee
IDVD	Inverse Dynamics in the Virtual Domain

IPOPT	Interior Point Optimizer
ISS	International Space Station
KNITRO	Nonlinear Interior-point Trust Region Optimization
LEO	Low Earth Orbit
LG	Legendre-Gauss
LGL	Legendre-Gauss-Lobatto
LGR	Legendre-Gauss-Radau
MATLAB	Matrix Laboratory
MINOS	Modular In-core Nonlinear Optimization System
MSIS	Mass Spectrometer and Incoherent Scatter Radar
NASA	National Astronautics and Space Administration
NLP	Nonlinear Programming
NOAA	National Oceanic and Atmospheric Administration
NRLMSISE-00	Naval Research Lab MSIS Extended 2000
NSERC	Natural Sciences and Engineering Research Council of Canada
OBC	On-board computer
OLE	Orbital Life Extension
PAMS	Passive Aerodynamically-stabilized Magnetically-damped Satellite
PD	Proportional-Derivative
PID	Proportional-Integral-Derivative
PS	Pseudospectral
RCS	Reaction Control System
RPY	Roll-Pitch-Yaw
SNOPT	Sparse Nonlinear Optimizer
SoC	System-on-Chip
SOCS	Sparse Optimal Control Software
SPeAD	Semi-empirical Piece-wise exponential Atmospheric Density
STM	State Transition Matrix
USAF	United States Air Force
ZPM	Zero Propellant Maneuver

List of Symbols

Throughout this thesis, scalar values are typeset in italics (e, U, ν), matrices are boldface ($\mathbf{J}$, $\mathbf{q}$, $\mathbf{\Phi}$), and vectors are represented with arrows ($\vec{r}$, $\vec{T}$, $\vec{\omega}$).

Symbol	Definition	Unit
a	Semi-major axis	km
e	Eccentricity	–
i	Inclination	deg.
Ω	Right ascension of ascending node	deg.
ω	Argument of periapse	deg.
ν	True anomaly	deg.
ρ	Atmospheric density	$\mathrm{kg{\cdot}m^{-3}}$
$\mu_\oplus$	Standard gravitational parameter of Earth	$\mathrm{km^3{\cdot}s^{-2}}$
A	Panel area	$\mathrm{m^2}$
c	Control damping coefficient	–
C_D	Satellite drag coefficient	–
J	Cost function	–
k	Control gain coefficient	–
$R_\oplus$	Equatorial radius of Earth	km
U	Gravitational potential function	–
z	Geometric altitude	km
Γ_{RAM}	Spacecraft angle from ram direction	deg.
$\mathbf{J}$	Spacecraft inertia tensor	$\mathrm{kg{\cdot}m^2}$
$\mathbf{q}$	Attitude quaternion	–
$\mathbf{\Phi}$	State Transition Matrix	–
$\mathbf{\Omega}$	Skew-symmetric angular rate matrix	$\mathrm{rad{\cdot}s^{-1}}$

Symbol	Definition	Unit
$\vec{H}_{RW}$	Reaction wheel angular momentum	$\mathrm{kg{\cdot}m^2{\cdot}s^{-1}}$
$\vec{r}_B$	Positional vector (BCI)	km
$\vec{v}_B$	Velocity vector (BCI)	$\mathrm{km{\cdot}s^{-1}}$
$\vec{r}_G$	Positional vector (ECI)	km
$\vec{v}_G$	Velocity vector (ECI)	$\mathrm{km{\cdot}s^{-1}}$
$\vec{u}$	Control torque	N·m
$\vec{T}$	Net torque in BCI	N·m
$\vec{\omega}$	Spacecraft angular rate vector	$\mathrm{rad{\cdot}s^{-1}}$

This thesis uses metric units in keeping with the standard practices and conventions of the satellite and spaceflight industry.

Chapter 1

Introduction

Over the past decade, there has been a significant growth in the number of spacecraft launches, particularly to Low Earth Orbit (LEO). In 2013, there were 48 successful launches to LEO, compared to 40 in 2012, and 25 in 2003[1]. With the establishment of the private sector as a reliable space supply line to the International Space Station (ISS), and the growing space presence of Asian countries such as India and China, accessibility to space appears to be on the rise. The increased accessibility opens up LEO to a variety of smaller payloads such as nano-satellites, pico-satellites and CubeSats. These spacecraft provide a cost-effective alternative to larger spacecraft, and can be utilized as Earth-imaging satellites, space technology test-beds, and student-built university science payload platforms. However, due to their simplicity and size, smaller spacecraft in LEO tend to have limited operational lifetimes.

Recent advances in nanosatellite technology and mobile computing have provided the capacity for increased on-orbit computing power. In 2013, STRaND-1 was launched into orbit, carrying a classic CubeSat computer and a Google Nexus One smartphone[2]. Additionally, as part of the Small Spacecraft Technology Program,

[1]Kyle, E., Space Launch Report, 2014 (accessed Mar. 12, 2014), www.spacelaunchreport.com

[2]Surrey Satellite Technology Limited, STRaND-1 Smartphone Nanosatellite, 2014 (accessed May 6, 2014), http://www.sstl.co.uk/Missions/STRaND-1–Launched-2013/STRaND-1/

NASA has launched five PhoneSats into LEO, the most recent in April 2014 aboard a SpaceX Falcon IX launch vehicle[3]. These missions demonstrated the ability to successfully incorporate smartphone technology into CubeSats, and presented a significant advancement over existing CubeSat processors such as the ISIS OBC, which is built around an ARM9 architecture clocked at 400 MHz[4].

1.1 Motivation

The improvements in satellite computational power open up the potential for the inclusion of on-board orbit and attitude propagation systems. Such a system has been previously designed for orbit determination [1], but requires frequent ground-based updates as it neglects perturbation effects in an effort to minimize computational cost. However, real-time knowledge of these perturbations and their effects – through on-board orbit and attitude propagation – could prove valuable for science missions, low-cost Earth observation platforms, and spacecraft guidance. For example, a 3U CubeSat could autonomously predict an increase in atmospheric drag, and accordingly plan a maneuver to reorient itself so as to minimize orbital decay. Nevertheless, from a scientific perspective, CubeSat payloads should have computational priority, and it is therefore desirable to minimize the processing power necessary for Guidance, Navigation and Control (GN&C) operations while including major perturbation effects.

As secondary or tertiary payloads on launch vehicles, CubeSats are typically deployed in LEO in a variety of inclinations. One class of these orbits is the Equatorial

[3]Hall, L. and Dunbar, B., Small Spacecraft Technology Program, 2014 (accessed May 15, 2014), http://www.nasa.gov/directorates/spacetech/small_spacecraft/#.U4KVjPldXXQ

[4]CubeSatShop.com, ISIS On Board Computer, 2014 (accessed May 26, 2014). http://www.cubesatshop.com/index.php?

Low Earth Orbit (ELEO). It is a valuable asset for the future of the scientific community and humankind located between an altitude of 200 - 1200 km, primarily due to the low delta-V required to attain it from an equatorial launch site. In 2007, the International Astronautical Federation (IAF) encouraged research into the utilization of ELEO for space science applications, satellite telecommunications and Earth observation [2]. The Japanese space industry identified ELEO as the orbit of choice for their SPS2000 prototype solar power satellite [3]. Furthermore, ELEO has been proposed as an ideal location for future in-orbit construction of power facilities [4] and spaceports[5].

1.2 Problem Statement

For a space mission to be successful, a satellite must be capable of accurately pointing its payload at the required target [5]. The associated slewing maneuvers are typically executed by a series of motorized rotating masses such as reaction wheels, momentum wheels, or control moment gyroscopes (CMG), which provide maneuvering torque and angular momentum storage [6]. Over the duration of a mission, these devices accumulate angular momentum while countering persistent external disturbance torques from the space environment. This momentum accumulation necessitates the use of additional momentum control systems, such as magnetorquers or thrusters, to desaturate the wheels in a process known as momentum dumping. However, a full-size attitude control system (ACS) is too large and expensive to be installed on nano-satellites or CubeSats [7], leading to the widespread use of pico-satellite reaction wheels [8] and passive control systems [9, 10].

[5]Sonter, M., Equatorial Low Earth Orbit, an ideal orbit for In-Space Construction. Asteroid Enterprises Pty. Ltd., 2005 (accessed November 12, 2013). http://www.spacefuture.com/archive/

Magnetorquers are often used alongside an active ACS for momentum dumping tasks [11], due to their low cost, reliability, and simplicity. Though highly efficient, magnetorquers are not without their disadvantages. The Michigan Exploration Laboratory suspects that in October 2011, their M-Cubed CubeSat unintentionally became magnetically conjoined to Explorer-1 Prime, a second CubeSat released simultaneously, via strong onboard magnets used for passive attitude control[6]. Such an occurrence highlights the necessity to explore alternate momentum dumping techniques as applicable to nano-satellites and CubeSat class payloads. Additionally, the magnetic field of the Earth is not well known, and is continuously being affected by solar weather. Near the geomagnetic equator, only roll and yaw momentum can be dumped [12, 13], allowing for the saturation of pitch momentum. Furthermore, the use of magnetic dipoles generates an additional magnetic field on the spacecraft, which may disrupt the normal operation of sensors and spacecraft systems.

Though microthruster attitude control systems are being developed for nanosatellite applications [14], existing systems are prone to leakage and require bulky pressure vessels for propellant storage. Valve actuation solenoids, though typically reliable, have the potential to be stuck open or closed, leading to mission-threatening scenarios, as was experienced during the Gemini 8 mission[7] in 1966.

The research within this thesis focuses on the development and simulation of a near real-time low computational-cost guidance algorithm which exploits gyroscopic (internal) and space environment (external) disturbance torques to minimize the use of torque actuators during attitude maneuvers, hence minimizing the accumulation

[6]Grayzeck, E., National Space Science Data Center COSPAR ID: 2013-072H. NASA, 2009 (accessed March 24, 2014), http://nssdc.gsfc.nasa.gov/nmc/spacecraftDisplay.do?id=2013-072H

[7]Mueller, G. E., Mathews, C. W., and Schneider, W. C., National Space Science Data Center COSPAR ID:1966-020A. NASA, 2009 (accessed March 30, 2014), http://http://nssdc.gsfc.nasa.gov/nmc/spacecraftDisplay.do?id=1966-020A

of angular momentum. The use of the space environment in this manner is comparable to the captain of a naval vessel using the inertia of ship and ocean currents to navigate between ports instead of sailing in a straight line, in an effort to reduce fuel consumption. Such an algorithm could significantly reduce the frequency of momentum dump events, thereby decreasing satellite downtime. Additionally, minimizing the usage of ACS actuators would potentially increase their lifespan. For larger satellites, this technique could reduce RCS propellant usage over the duration of the mission, and hence lower lifetime operational costs. Furthermore, reduced RCS usage would minimize the risk of spacecraft surface contamination by the associated propellants. Similar guidance algorithms have been successfully implemented for large-angle reorientation of the International Space Station (ISS). On November 5, 2006 [15] and March 3, 2007 [16], Zero Propellant Maneuver (ZPM) trajectories were utilized for 90° and 180° rotation of the ISS respectively. Combined, the two ZPMs saved approximately US$1,500,000 in propellant costs [17]. However, these ZPM trajectories were generated offline at ground stations before being uploaded to the ISS control system.

1.3 Previous Work

Typical satellite control systems perform pointing, slewing, and maneuvering by a rotation about an Eigenaxis. An Eigenaxis rotation is the shortest angular path between two attitude states, a rationale that is commonly accepted based on its widespread implementation. However, from quaternion kinematics, it can be shown [18] that Eigenaxis maneuvering is not necessarily time-optimal. Furthermore, Eigenaxis trajectories require the controller to continually fight against the various environmental disturbance torques experienced by the spacecraft [12], leading to an increased torque

requirement.

It is often beneficial to optimize maneuvers to minimize a specific "cost" – typically maneuver duration or associated actuator usage. Doing so can significantly increase spacecraft agility, reduce the need for bulky control actuators, and extend control system lifespan. The optimal satellite reorientation problem is therefore of significant interest in the field of aerospace engineering. Prior work in the field, which has focused on time optimal and torque optimal guidance laws, shall now be presented. Additionally, the use of pseudospectral methods as a means to obtain numerical solutions for the optimal guidance problem shall be outlined. Finally, an overview of prior CubeSat missions is provided to familiarize the reader with the CubeSat platform.

1.3.1 Time Optimal Guidance

The general case of minimum-time reorientation of an asymmetric rigid body was first addressed by Proulx and Ross [19]. For control systems in which the three orthogonal components of control are independently constrained, Bilimoria and Wie discovered that non-Eigenaxis maneuvers require less maneuver time than Eigenaxis maneuvers, as the nutational components can provide additional torque along the reorientation axis [18]. This was subsequently proved via numerical simulation by Bai and Junkins in their investigation of time-optimal spacecraft reorientation [20]. Time-optimal attitude maneuvers have been performed in flight, most recently by NASA's Transition Region and Coronal Explorer (TRACE) spacecraft. The design and flight implementation of the associated guidance algorithms, presented by Karpenko et al. [21], successfully demonstrated the ability of conventional Eigenaxis controllers to follow time-optimal attitude trajectories.

1.3.2 Torque Optimal Guidance

Torque-optimal guidance focuses on the reduction of the net angular momentum required for satellite reorientation maneuvers. These maneuvers are typically performed utilizing reaction wheels, CMGs, thrusters, or a combination thereof. However, reaction wheels and CMGs have limited angular momentum capacity. Once the minimum/maximum limit is reached, the actuator is said to be saturated, and can no longer provide reliable control authority. To recover attitude control capability, thrusters are used to maintain spacecraft attitude while the individual actuators are reoriented or spun down to decrease the total momentum magnitude. This process is known as desaturation, and is a routine on-orbit operation for spacecraft.

CubeSat class satellites typically do not possess thrusters due to volume, cost, and mass constraints. They rely solely on passive techniques, such as gravity gradient torque and magnetorquers, to desaturate their momentum exchange devices. Related work on the use of environmental torques for actuator desaturation dates back to the Skylab program, where gravity gradient torque was used to desaturate momentum accumulated during the daylight portion of an orbit [22, 23]. Additional applications regarding the use of gravity gradient torque to desaturate CMG momentum have been previously proposed by Powell [24] and Tong [25]. Tong showed in his study that despite complete momentum dumping not being achieved, a significant attitude maintenance CMG desaturation fuel saving of over 40% was attainable, thus demonstrating the potential for extending the life span of a spacecraft.

Prior research has predominantly focused on applications for space stations and large spacecraft. In 2000, an approach for optimizing the attitude command sequence to the ISS CMG attitude hold controller was proposed by Chamitoff et al. [26] This

technique focused on minimizing fuel usage while including nonlinear system dynamics. An ISS 90° yaw maneuver was performed without use of propellant, by propagating a simplified model of the vehicle dynamics and environment which included Euler components and gravity gradient, but excluded aerodynamic effects. However, at the end of the maneuver, the CMGs were saturated.

Expanding upon this existing research, Bedrossian and Bhatt developed the ZPM in collaboration with Draper Laboratory [16]. Its origins can be traced back to the mid-1990's when Bedrossian proposed this approach during the development of a general Centralized Momentum Management (CMM) concept [27]. The goal of the CMM was to increase satellite life by trading off between satellite performance and control variables, while incorporating the ability to "look-ahead" in guidance algorithm decision making. In an effort to achieve these goals, a general optimal control problem framework was proposed.

The ZPM can be defined as the development of an attitude trajectory shaped in a manner that takes advantage of the nonlinear system dynamics and environmental disturbance torques to reduce or eliminate the net cost of a rotation [16]. As described earlier in Sec. 1.3, an Eigenaxis maneuver, though kinematically the shortest past, typically results in an increased "cost". By considering a kinematically longer path, the path dependence of the system dynamics can be utilized to lower this cost. In the early 2000s, the ZPM concept was used to establish the feasibility of general three-axis momentum desaturation without the use of thrusters [28]. The study indicated that by suitably maneuvering the spacecraft in a disturbance field, the CMGs could be desaturated.

From the spacecraft dynamics and kinematics, and the associated environmental models, a nonlinear, constrained optimal control (more accurately *guidance*) problem can be defined. Solving such a problem is still considered difficult, despite the

mathematics of dynamical optimization theory being established by Euler in the 18^{th} century [29]. In 1962 Pontryagin's principle revolutionalized optimization theory by providing a method to determine optimal control in a constrained problem. However, it is only with recent advances in pseudospectral (PS) methods that have allowed for efficient solutions to optimal control problems in nonlinear dynamics systems.

1.3.3 Pseudospectral Methods

Pseudospectral methods are a class of direct collocation wherein the optimal control problem is transcribed into a nonlinear programming problem (NLP) by parameterizing the dynamical states and associated control variables using global polynomials and collocating the differential-algebraic equations using nodes obtained from a Gaussian quadrature [30].

Three of the most common sets of collocation points are the *Legendre-Gauss* (LG), *Legendre-Gauss-Radau* (LGR), and *Legendre-Gauss-Lobatto* (LGL) points. All these sets are defined on the non-dimensionalized time domain $[-1, 1]$, but differ significantly in their handling of endpoints. LG points include neither of the endpoints, the LGR points include one endpoint, and LGL points include both endpoints. Of these, the Lobatto PS method (LPM) [31, 32] and the Gauss PS method (GPM) [33–35] have been extensively documented in recent years. A detailed comparison of the performance of the LG, LGR, and LGL schemes in a first-order optimality system was performed by Garg et al [30], and the results indicated that the Gauss or Radau collocation methods provided state and control accuracies several orders of magnitude more accurate as compared to the Lobatto method. However, in applications where the exact solution was unknown, all three methods provided similar results, with the Lobatto costate oscillating about the correct costate.

1.3.4 Software for Solving Optimal Problems

Multiple software packages are available to solve the optimal control (or guidance) problems outlined in Sec. 1.3.1 and Sec. 1.3.2. MATLAB provides an optimization toolbox, called using the `optimtool` command, which allows the user to solve constrained and unconstrained problems with or without the inclusion of control states [36]. However, there are multiple limitations regarding the use of `optimtool`, including the inability to configure and solve multi-phase problems. An example of such a problem would be to have a spacecraft reorient itself, pause for a specified time, and then execute a subsequent reorientation in quick succession.

Dedicated software packages, which combine a pseudospectral collocation scheme with a Non-Linear Problem (NLP) solver, have been developed over the years. Examples of such programs are DIDO [37], SOCS (Sparse Optimal Control Software) [38], and GPOPS (General Pseudospectral OPtimal Software) [39], all of which allow for the evaluation of nonlinear optimal control or guidance problems with path constraints [40].

DIDO was developed by Ross and associates at the Naval Postgraduate School in 2001, and implements the Legendre PS collocation method alongside the SNOPT (Sparse Nonlinear OPTimizer) NLP solver [41]. It requires no prior knowledge of optimization techniques, nor does it require the user to provide an initial guess. Due to its ease of use, it is used extensively in academia, industry and government laboratories [42]. DIDO has also been flight-tested by NASA, and played an integral role in the formulation of the ZPM outlined earlier in Sec. 1.3.2.

The GPOPS package, on the other hand, employs a Gaussian pseudospectral method. By utilizing the SNOPT solver, GPOPS simultaneously solves the entire trajectory over a small number of nodes based on the path constraints, initial conditions, and initial guesses provided by the user. Due to its open source nature and

free availability, the GPOPS package was selected to generate the optimal trajectories. Furthermore, in the context of this thesis, the use of GPOPS allows for direct comparison of time-optimal guidance trajectories with the work of Boyarko et al. [43] This simplifies the validation of the optimization environment prior to the development of a torque-optimal guidance algorithm. Further details regarding the setup and structure of the GPOPS environment are provided in Sec. 5.2.

1.3.5 CubeSat Missions

CubeSats are part of a niche satellite market emerging from the desire of the university community to perform rapidly developed, cost-effective missions on very small spacecraft. Having a cross-sectional profile of 10 cm $\times$ 10 cm, measuring between 10 and 30 cm in length, and weighing between 1 and 4 kilograms, CubeSats form a standardized platform for building satellites [44]. First developed in the late 1990's at California Polytechnic State University and Stanford University, the CubeSat was originally intended for educating students about the capabilities of Sputnik, the first man-made satellite put into orbit around the Earth .

CubeSats can provide a low-cost capability for university students and the academic community to test scientific and space technology payloads in a microgravity environment. For example, in September 2013, the Colorado Student Space Weather Experiment (CSSWE) 3U CubeSat, designed and built by students at the University of Colorado in Boulder, CO, was launched into space[8].

[8]University of Colorado, Colorado Student Space Weather Experiment. 2014 (accessed June 1, 2014), http://lasp.colorado.edu/home/csswe/

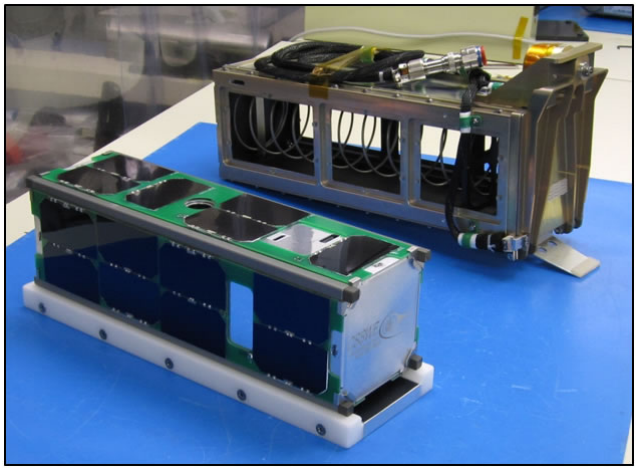


Figure 1.1: The CSSWE CubeSat with P-POD launcher (Image credit: UC Boulder)

Funded by the National Science Foundation, the mission focused on investigating space weather phenomena through use of an energetic particle telescope. CubeSat platforms also allow developing nations access to space for scientific purposes. In May 2013, the Ecuadorian NEE-01 *Pegaso* technology demonstration satellite, pictured in Fig. 1.2, was launched on a Long March 2D rocket[9].

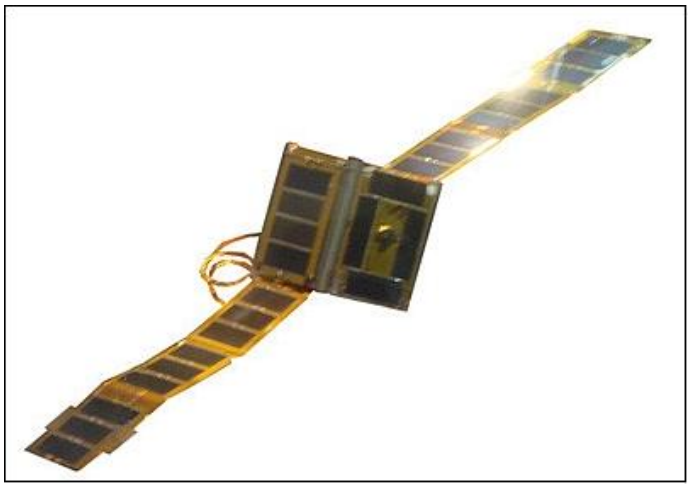


Figure 1.2: The *Pegaso* CubeSat with solar arrays deployed (Image credit: EXA)

[9]European Space Agency, NEE-01 Pegasus (Ecuadorian Space Ship-01, a CubeSat Mission). 2013 (accessed April 27, 2014). https://directory.eoportal.org/web/eoportal/satellite-missions/n/nee-01-pegasus

The mission focused on evaluating various thermal management technologies. In addition it demonstrated the successful in-flight implementation of ultra-thin deployable solar panels on a CubeSat platform [45]. The use of the space environment for spacecraft control has also been demonstrated in flight, though it has been limited to orbit management. The NanoSail-D2 3U CubeSat, deployed in January 2011, was built by NASA's Marshall Space Flight Center and Ames Research Center to study the deployment and effectiveness of solar sails[10].

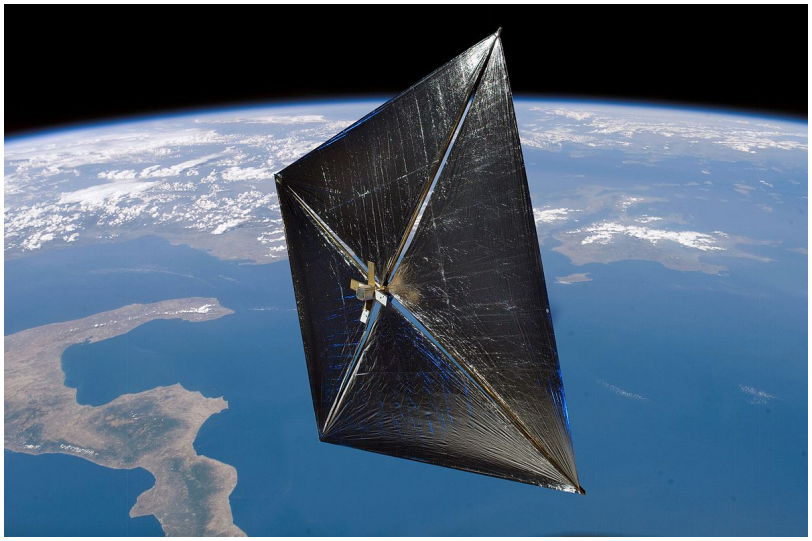


Figure 1.3: Artist's impression of the NanoSail-D2 in Low Earth orbit with deployed solar sail (Image credit: NASA)

The NanoSail-D2 demonstrated that a small, low-cost CubeSat platform can successfully utilize the space environment for orbit control. Similar research has been conducted into the use of atmospheric drag for passive attitude stabilization of CubeSats [46], though flight tests have not yet been conducted. Optimal guidance, as described in Sec. 1.3, has not been previously pursued specifically for CubeSat applications. The research within this thesis aims to fill this knowledge gap.

[10]NASA, NASA's First Solar Sail NanoSail-D Deploys in Low-Earth Orbit. 2011 (accessed May 14, 2014). http://www.nasa.gov/mission_pages/smallsats/11-010.html

1.4 Objectives

A 3U CubeSat located in a circular ELEO at an altitude of 400 km is considered for the design and simulation of this attitude guidance algorithm. The selected orbital altitude is comparable to that of the ISS[11]. The primary objective is to demonstrate the successful implementation and simulation of a torque-optimal spacecraft guidance routine in the MATLAB-Simulink environment. The results shall be quantitatively compared with maneuvers of identical duration performed using an Eigenaxis rotation. Additionally, the model and algorithm are designed with minimum computational cost in mind, so as to allow the guidance routine to be implemented in near real-time aboard nanosatellite platforms. Two simulation scenarios considered in this thesis are:

1. a 90° rotation about the yaw axis in 100 seconds, and

2. a 180° rotation about the yaw axis in 100 seconds.

These scenarios are adapted from the in-flight demonstrations of ZPMs [15, 16]. The choice of the maneuver duration, derived from Bilimoria and Wie [18], is detailed in Sec. 5.4.6.

1.5 Approach and Assumptions

Figure 1.4 presents the overview of the Simulink model for this research. The optimal guidance routine, detailed in Chapter 5, utilizes the current spacecraft attitude, target attitude, and environmental torques to formulate a minimum-torque guidance trajectory. The torque-optimal path is then simulated in a real-world dynamics model,

[11] Peat, C., Heavens-Above: ISS - Orbit. 2014 (accessed June 18, 2014), http://www.heavens-above.com/orbit.aspx?satid=25544

allowing for evaluation of the performance of the guidance routine. Figure 1.4 depicts the flow of state data between each of the blocks in the Simulink model.

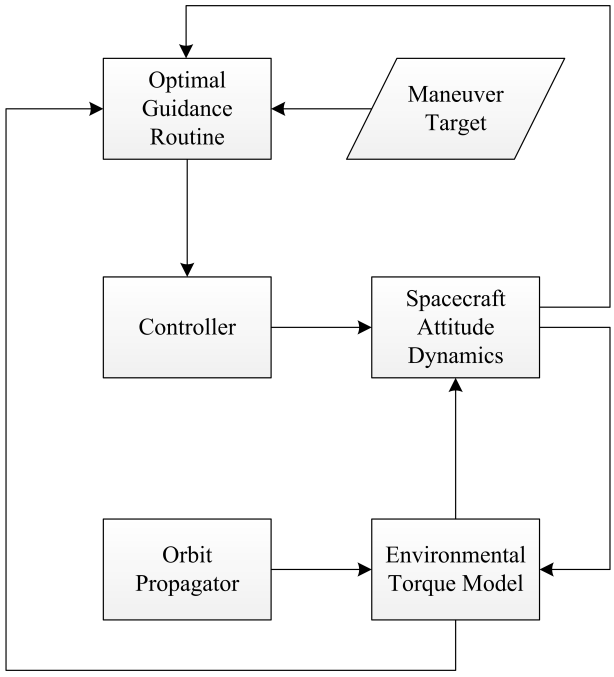


Figure 1.4: Simulation Overview

For the purposes of this study, solar radiation pressure and magnetic torques were not considered in the equations of orbital motion, vehicle dynamics, and kinematics. Sec. 2.1 and Sec. 2.2.3 provide further details regarding the space environment as applicable to the considered orbit. Furthermore, the satellite will be considered as a fully rigid body, and the effects of flexible structures shall be neglected. The implementation of the vehicle dynamics, guidance module, controller and environmental torque model utilizes quaternions in the MATLAB-Simulink environment, hence preventing the occurrence of singularities and numerical instability typically associated with Euler angle based simulations [47]. A final note is that the guidance and validation routines assume that the measured and actual attitude states are equal. In reality, the actual and sensed attitude vary due to sensor noise and error. However, inclusion of the methods for calculating the attitude from sensor data using filters is

beyond the scope of this thesis.

1.6 Contributions

This thesis has three contributions. The first is the development, implementation, and testing of a computationally lightweight atmospheric density model for on-board implementation on CubeSats. The second is the use of this lightweight density model to examine the undamped aerostabilization characterisitics of a 3U CubeSat equipped with drag sails. Both these contributions are ancillary, and assist in the development and validation of the main contribution – a torque-optimal guidance routine for CubeSat applications. Figure C.3 in Appendix C outlines the interlinking of various thesis components towards the development and validation of the guidance routine.

1.7 Organization

The purpose of this thesis is to demonstrate the conceptual viability of simulating torque-optimized spacecraft maneuvers in real-time on board CubeSat class satellites. This thesis is organized as follows. Details regarding the modelling of the spacecraft orbit and attitude dynamics are presented in Chapter 2. Chapter 3 presents the development and validation of the SPeAD-M86 atmospheric density model which was developed in support of this research. Chapter 4 examines the potential for passive aerostabilization of a 3U CubeSat with drag sails while simultaneously validating the SPeAD-M86 model and real-world simulation environment. In Chapter 5, the dynamical optimization approach and guidance algorithm is presented. Chapter 6 presents the results and validation of the torque-optimal guidance algorithm. Finally, Chapter 7 presents conclusions and the potential for future work.

Chapter 2

Spacecraft Dynamics Modelling

The modelling of the spacecraft orbital mechanics and attitude dynamics shall now be discussed. Pertinent details regarding the mathematical modelling of the space environment and associated equations are included.

2.1 Orbit Propagation

There are several perturbing accelerations which affect the orbit of a satellite in space. These perturbations include atmospheric drag, Earth oblateness, luni-solar gravitational forces, micrometeorite impacts, and solar radiation pressure. Figure 2.1, adapted from Montenbruck and Gill [48], summarizes the effects of several space environment perturbations on the orbit of a satellite as a function of geocentric satellite distance.

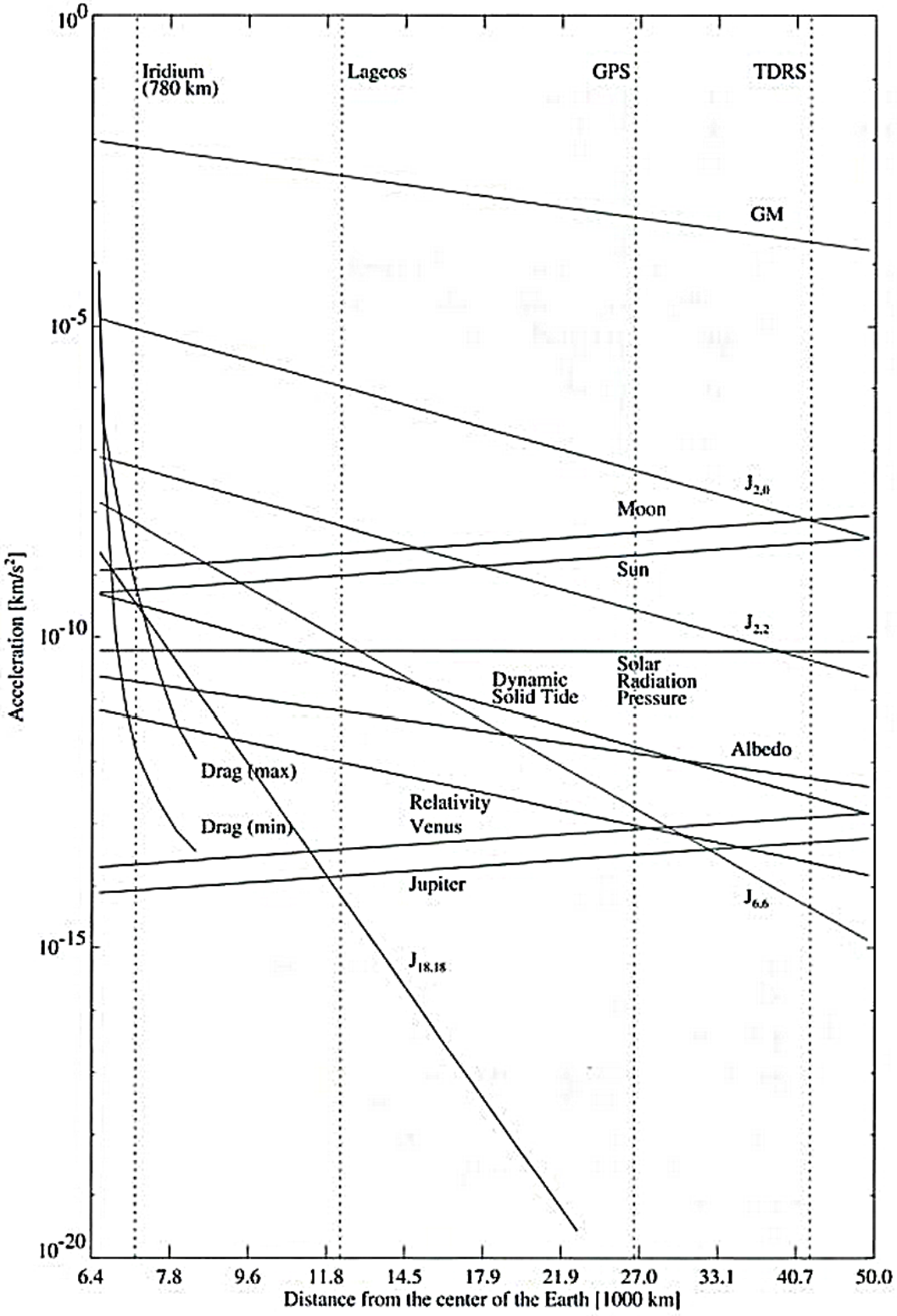


Figure 2.1: Orders of Magnitude of Satellite Perturbation [48]

From Fig. 2.1, at an altitude of 400 km, corresponding to a geocentric distance of 6778 km, accelerations due to the Earth's gravitational field and atmospheric drag dominate. The central body attraction, GM, along with the spherical harmonic gravitational terms J_{20}, J_{22}, and J_{66} exert accelerations of up to a magnitude of $10^{-8}\frac{km}{s^2}$ on the satellite. Perturbations below this order of magnitude were not considered for propagation, as they require implementation of the computationally complex n-body problem.

Having established that gravitational and atmospheric accelerations dominate the motion of the spacecraft at the selected altitude, the scope of modelling the space environment is greatly reduced. The governing equation for the orbital state of a satellite in LEO is given by the sum of $\ddot{\vec{r}}_U$ and $\ddot{\vec{r}}_{aero}$, the gravitational and aerodynamic accelerations respectively.

$$\ddot{\vec{r}}_G = \ddot{\vec{r}}_U + \ddot{\vec{r}}_{aero} \tag{2.1}$$

$\ddot{\vec{r}}_{aero}$ is given by

$$\ddot{\vec{r}}_{aero} = -\frac{1}{2} C_D \frac{A}{m} \rho |\vec{v_r}| \vec{v_r} \tag{2.2}$$

where,

$$A = \sum_{i=1}^{n} A_n \begin{cases} A_n = 0, & \text{if } \vec{A}_n \cdot \vec{v_r} < 0. \\ A_n = \vec{A}_n \cdot \vec{v_r} & \text{otherwise.} \end{cases} \tag{2.3}$$

m is the spacecraft mass, C_D is the drag coefficient, A is the total projected area perpendicular to $\vec{v_r}$, $\vec{A}_n$ is the area vector for the n^{th} spacecraft panel in the body frame, ρ is the atmospheric density (can be obtained from a number of models – refer Chapter 3), and $\vec{v_r}$ is the velocity of the spacecraft relative to the atmosphere. The spacecraft drag coefficient, C_D, is a difficult parameter to determine, due to the complex interaction of spacecraft geometry, reflection, molecular content, and spacecraft attitude [49]. Typically, for continuum flows, C_D is in the range of 2 –

4. Vallado and Finkleman [50] provide an overview of research into spacecraft drag coefficient determination. For the purposes of this study, a C_D of 2.2 was selected, corresponding to the free molecular regime as defined by Montenbruck and Gill [48].

Keplerian orbits, which treat the central body as a spherical point mass, provide a reasonable first approximation of satellite motion. Though the first zonal harmonic term, J_2, is sufficient for obtaining an approximation of an orbit, precision orbit propagation is essential to determine the initial and final orbital states required for solving an optimal guidance problem. The use of spherical harmonic gravity models allows for accurate modelling of the Earth's gravitational potential. Due to the daily rotation of the Earth, and various mass concentrations throughout the lithosphere, the Earth resembles an oblate spheroid. The geopotential function of the Earth, which describes this uneven internal mass distribution, can be expressed as a series of Legendre polynomials. The resulting acceleration due to gravitational potential, in the form of spherical harmonics, is presented in Eq. (2.4) as given by Milani et al. [51]

$$\ddot{\vec{r}}_U = \nabla U = \nabla \frac{GM_\oplus}{r} \sum_{n=0}^{\infty} \sum_{m=0}^{n} \frac{R_\oplus^n}{r^n} \bar{P}_{nm}(sin\phi)(\bar{C}_{nm}cos(m\lambda) + \bar{S}_{nm}sin(m\lambda)) \tag{2.4}$$

where the normalized associated Legendre functions are given by

$$\bar{P}_{nm} = \sqrt{\frac{(2\delta_{0_m})(2n+1)(n-m)!}{(n+m)!}} P_{nm} \tag{2.5}$$

In Eq. (2.4), G is the universal gravitational constant, $M_\oplus$ is the mass of Earth, $R_\oplus$ is the equatorial radius of Earth, r is the geocentric distance, ϕ represents latitude, and λ represents longitude. $\bar{C}_{nm}$ and $\bar{S}_{nm}$ are the normalized geopotential coefficients. Geopotential coefficients with $(m = 0)$ are called zonal coefficients, as they describe the part of the potential which is independent of longitude. Similarly, coefficients with

$(m = n)$ are called sectorial coefficients, and describe the longitudinal variance in the gravitational potential due to the elliptical cross section of the Earth at its equator. Lastly, coefficients with $(m < n)$ are known as tesseral coefficients, and are dependent on both latitude and longitude. The tesseral coefficients quantify the existence of mass concentrations, or 'mascons', in the lithosphere of the Earth. These mascons produce local gravitational anomalies, such as those present near the Indonesian archipelago and the South Indian Ocean[1]. Various gravity models have been developed over the years based on surface gravity data and, more recently, through orbital measurements of the gravity field. Within this thesis, all spherical harmonic gravity models were implemented using the EGM2008 model within the Simulink environment.

2.2 Spacecraft Attitude Dynamics

This section presents the dynamics and kinematic equations of motion for a spacecraft and the mathematical formulation of the dominant environmental torques in LEO.

2.2.1 Euler Equation of Motion

The angular acceleration of an object, in the body frame of reference with the origin of the principal axes located at the center of mass, can be defined using Euler's equation for rigid body dynamics [13], presented in Eq. (2.6).

$$\dot{\vec{\omega}} = \mathbf{J}^{-1}\left(\vec{T} - \vec{\omega} \times \mathbf{J}\vec{\omega}\right) \tag{2.6}$$

The torque, $\vec{T}$, is the vector summation of the control torque and environmental torques, while $\vec{\omega}$ is the angular rate vector of the spacecraft.

[1]PIA04652: New Views of Earth's Gravity Field from GRACE, JPL Photojournal, July 2003 (accessed April 18, 2014). http://photojournal.jpl.nasa.gov/catalog/PIA04652

2.2.2 Quaternion Kinematics

A quaternion attitude representation was selected over the use of Euler angles to avoid issues associated with singularities. The rate of change of the quaternion attitude state is defined by quaternion kinematics [13], presented in Eq. (2.7).

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{\Omega q} = \frac{1}{2}\begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{2.7}$$

Components q_1 through q_3 of $\mathbf{q}$ represent the Eigenaxis, while q_4 represents the rotation angle. The time rate of change of the attitude states, $\dot{\vec{\omega}}$ and $\dot{\mathbf{q}}$, are concatenated together to form the derivative of the state matrix, which describes the instantaneous rates of change of the attitude states.

2.2.3 Perturbing Torques

The environmental perturbation torques predominantly stem from gravitational, atmospheric, and solar forces acting upon a spacecraft. From Schrello [52], and GN&C criteria specified by NASA [53], aerodynamic and gravity gradient torques are the dominant perturbations in LEO between 300 km and 650 km.

Gravity Gradient Torque

Any object with finite dimensions orbiting the Earth experiences a differential gravitational force at different radii along its length. This force is akin to the tidal force exerted by the Moon and the Sun on the Earth, the effects of which are visible as

daily ocean tides. The differential gravitational force imparts a torque on the satellite about its center of mass[2]. Equation (2.8) presents the generalized expression for the gravity gradient torque [54].

$$\vec{T}_{gg} = \frac{3\mu_{\oplus}}{r_B^3}\left(\frac{-\vec{r}_B}{r_B} \times \mathbf{J}\frac{-\vec{r}_B}{r_B}\right) \tag{2.8}$$

An asymmetric body in a gravitational field will experience a torque aligning the axis of least inertia with the field direction [55]. The transformation of the satellite position from inertial to body coordinates is performed using a direction cosine matrix derived from the quaternion state of the spacecraft, given by

$$DCM(\mathbf{q}) = \begin{bmatrix} q_4^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & q_4^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & q_4^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.9}$$

Atmospheric Torque

The aerodynamic forces acting on a spacecraft do not necessarily act through the center of mass of a spacecraft. This offset results in an aerodynamic torque acting upon the spacecraft, which can be expressed by

$$\vec{T}_{aero} = \sum_{i=1}^{n} \vec{F}_{drag_i} \times \vec{r}_{cp_i} \tag{2.10}$$

where $\vec{F}_{drag_i}$ is the drag force acting on each spacecraft panel, and $\vec{r}_{cp_i}$ is the center of pressure location of each associated spacecraft panel in the body frame.

[2]Hall, C., Attitude Dynamics and Kinematics, AOE 4140 Lecture Notes, Virginia Tech, 2009 (accessed April 12, 2014), http://www.dept.aoe.vt.edu/ cdhall/courses/aoe4140/

2.3 Propagation Environments

Three different orbit and attitude propagation environments are utilized over the course of this thesis. The real-world propagator is utilized for guidance trajectory validation, while the on-board propagator (Sec. 5.4) generates the initial and final conditions required by the guidance algorithm. The optimization propagator, detailed in Sec. 5.4, is specifically designed to minimize computations within the optimization algorithm. Table 2.1 summarizes the specifications for the real-world propagator.

Parameter	Real-world Propagator
Time step, Δt (sec)	0.1
Gravity model	Spherical harmonic
Order of gravity model	8
Atmospheric Model	NRLMSISE-00
Orbit Integrator	ode3
Attitude propagation	Quaternion
Attitude Integrator	ode3

Table 2.1: Summary of Real-world Propagator

The Simulink block diagram for the real-world environment is presented in Figure C.1 in Appendix C.

Chapter 3

SPeAD-M86 Model

Atmospheric drag plays a significant role in affecting the orbit and attitude of a spacecraft in LEO, as presented in Eq. (2.2) and Eq. (2.10). As these perturbations are directly proportional to the local density, it is essential to accurately model variations in atmospheric density with position. However, for CubeSat applications, this accuracy must be maintained while minimizing the required computational cost. This chapter presents the development and testing of a Semi-empirical Piece-wise exponential Atmospheric Density model based on MSIS-86 data (SPeAD-M86).

3.1 Existing Atmospheric Density Models

Eight atmospheric density models which are commonly utilized for precision orbit determination shall now be discussed briefly, with focus given to their respective strengths and weaknesses.

1. **Simple Exponential Model:** The exponential relation of density with altitude is apparent from data published in standard and reference atmospheres [56]. Therefore, as a first approximation, an exponential relation dependent on

the geometric altitude can be defined.

$$\rho = \rho_0 \exp\left(\frac{z - h_0}{H}\right) \tag{3.1}$$

where ρ is the density at any altitude, ρ_0 is the density at some specified base altitude h_0, H is the scale height at h_0, and z is the spacecraft altitude above sea level. The scale height is defined as the height over which the atmospheric pressure drops to e^{-1} of that at the base altitude. Such a model, based on Vallado [57], was implemented during ground testing of the PROBA 1 and 2 spacecraft GN&C systems. Though simple, this model is constrained to a small altitude range to remain accurate, and therefore cannot be utilized for real-time orbital and attitude propagation on-board nanosatellites in high eccentricity orbits.

2. **Harris-Priester Model:** The Harris-Priester model is based on upper atmosphere data derived from the heat conduction equation. It accounts for the daily temperature variations in the atmosphere by means of a cosine variation. Presented below is the general expression for the Harris-Priester model [48, 58].

$$\rho(z) = \rho_m(h) + [\rho_M(h) - \rho_m(h)] \cdot cos^n\left(\frac{\Psi}{2}\right) \tag{3.2}$$

where $\rho_m(h)$ and $\rho_M(h)$ are piece-wise functions of the Harris-Priester atmospheric density coefficients at specified altitude intervals and the spacecraft altitude z, n is dependent on the orbit inclination, and Ψ is the angle between the position of the spacecraft and the density peak. Calculating this angle requires data on the sun's location with respect to the spacecraft, which is not always readily available on CubeSats.

3. **Jacchia 1971 Model:**

   L.G. Jacchia published a number of empirical density models, which evolved between 1965 and 1977. Initially focused on geodetic height and temperature as the parameters, these models eventually provided density variations as a function of time. The Jacchia 1971 (J71) model provides time-variant atmospheric density between altitudes of 90-2500 km [59]. This model does however require numerical integration, making it computationally intensive to implement.

4. **Jacchia-Roberts:** Robert's method is based on the analytical solutions of the barometric and diffusion differential equations obtained by integration of the governing partial differential equations. Derived from the J71 model, the results closely agree with Jacchia for altitudes above 125 km. Furthermore, Robert's model eliminates the need for numerical integration, improving computational speed. [60]

5. **CIRA 1972:** The COSPAR International Reference Atmosphere (CIRA) is an empirical model of atmospheric temperature and density from 0 - 2000 km. It combines mean data from 25 - 500 km, obtained from satellite drag measurements and ground observations, with J71 data from 110 - 2000 km [61]. CIRA has been periodically updated as new data is obtained, most recently in 1986.

6. **1976 US Standard Atmosphere:** As a result of extensive rocket data and revised theories for the upper atmosphere, the existing 1962 US Standard Atmosphere was updated and published in a joint effort between NOAA, NASA and the USAF [62]. Atmospheric properties up to 1000 km altitude based on mid-latitude measurements are tabulated, requiring the storage and use of large lookup tables.

7. **Jacchia-Bowman 2008:** The Jacchia-Bownman 2008 (JB-2008) model is an empirical, thermospheric density model, developed to improve upon the JB-2006 model, which was developed from the CIRA 1972 model diffusion equations [63]. The modifications implement new exospheric temperature and semi-annual density equations, factor in geomagnetic storm effects, and include updated solar indices.

8. **NRLMSISE-00 Atmospheric Model:** NASA Goddard Space Flight Center developed several empirical neutral atmospheric models, referred to as the Mass Spectrometer Incoherent Scatter (MSIS) model class, based on mass spectrometer and incoherent radar scatter data from satellite and ground measurements. Inputs to the model, are year, day of year, universal time, altitude, geodetic latitude and longitude, local apparent solar time, solar $F_{10.7}$ flux and geomagnetic index. The MSIS-86 model constitutes the upper part of the CIRA-1986 [64]. In 1990, MSIS-86 was subsequently revised in the lower thermosphere and extended into the mesosphere and lower atmosphere to provide a single analytic model refered to as MSIS-90. [65]. The NRLMSISE-00 model, developed by the Naval Research Laboratory (NRL), incorporates data collected since the MSIS-86 and MSIS-90 models, and improves upon them by accounting for atomic oxygen, revising molecular constituents in the lower atmosphere and adding further nonlinear terms to account for the effects of solar activity [66].

3.2 Formulation

Though the atmospheric density models presented in Sec. 3.1 have been used previously for ground-based precision orbit determination, they require large lookup tables or multiple inputs in addition to spacecraft altitude. For potential CubeSat

applications, minimizing computational storage and processing requirements while maintaining reasonable accuracy is critical. Therefore, it was deemed essential to develop a simplified model which closely follows published data from existing models while striving for a minimal computational footprint.

3.2.1 Piece-wise Exponential Density Model

The simple exponential model, though severely limited by its inability to account for variable solar flux and geomagnetic effects in real time, offers a computationally inexpensive correlation between altitude and density. As previously mentioned, it fails to maintain accuracy over a large range of altitudes. Vallado and McClain, [57] and Walter [58] provide insight into the piece-wise exponential approach as an alternative formulation. It draws on similarities with the Harris-Priester model.

$$\rho = \rho_i \exp\left[\frac{-(z-h_i)}{H_i}\right] \quad @ \quad h_i < z < h_{(i+1)} \tag{3.3}$$

where z is the geometric altitude above sea level, h_i are the base altitudes for a given altitude interval, ρ_i is the corresponding base density, and H_i is the scale height applicable for the interval. The model provided by Vallado and McClain [57] was accompanied by a lookup table calibrated against the CIRA72 model. For completeness, this model was implemented, and found to be insufficient at providing a continuous density formulation. Results illustrating this are presented in Sec. 3.4.1. Table A.1 in Appendix A contains the associated altitude interval data.

3.2.2 Calibration

The accuracy of a piece-wise approximation is dictated by the empirical model against which it is calibrated. Given the outdated nature of CIRA72, and the desire to obtain

a model which best reflects the mean density, MSIS-86 atmospheric data and scale heights tabulated by Wertz and Larson [5] were used for the calibration. An MSIS model was selected for the calibration data set as it demonstrates slight improvements over the Jacchia models [67]. The data are averaged across the Earth with a 30° step size in longitude and a 20° step size in latitude. All data were for a mean $F_{10.7} = 118.7 \times 10^{-22}$ W·m^{-2}·Hz^{-1}. The $F_{10.7}$ index is a measure of overall solar activity and represents the solar radio flux per unit frequency at a wavelength of 10.7 cm. Density data corresponding to a maximum and minimum $F_{10.7}$ were not included in the calibration process, but are included in the results for comparison purposes. To allow for a better curve fit with the existing MSIS, the scale heights were modified for an altitude below 150 km. Furthermore, at altitudes above 1000 km, the density is set to zero. This proposed model is hereafter referred to as the SPeAD-M86b model, indicating the use of a baseline altitude. Columns 1-3 of Table A.2 in Appendix A contain the interval data employed during this calibration.

3.2.3 Reduction of Variable Dependence

In an effort to further simplify the density model, and reduce the number of lookup variables, the piece-wise exponential model was reduced to be solely dependent on the altitude above sea level, by eliminating the base height lookup variable. This approach led to the base density being redefined as a *scale density*, which does not necessarily correspond to the density at the beginning of an altitude interval.

$$\rho = \rho_{s_i} \exp\left(\frac{-z}{H_i}\right) \quad @ \quad h_i < z < h_{(i+1)} \tag{3.4}$$

where ρ_s is the scale density for a given altitude interval, while z and H_i are identical as defined earlier in Eq. (3.3)). A similar formulation for an isothermal model of

planetary atmospheres, often employed by planetary probes, is presented by Tewari [68]. Using this reduced variable dependence required re-calibration of the model to find ρ_s for each altitude interval. The calibration was performed so as to minimize discontinuities in the density between intervals. This model is hereafter referred to as the SPeAD-M86 model. Table A.2 in Appendix A presents the interval data employed during this calibration.

3.3 Validation

To validate the SPeAD-M86 model, it was necessary to compare the computed density with published empirical and analytic atmospheric density data. Empirical data was obtained from the MSIS-86 [5] and JB-2008, mean CIRA-86 [56], and NRLMSISE-00[1] at the average, maximum, and minimum $F_{10.7}$, 1976 US Standard Atmosphere [56] at equatorial latitude averaged over diurnal and seasonal variations. Data points from empirical sources were obtained by digitizing of the relevant graphics. The SPeAD-M86 model results were also compared against the CIRA72 piece-wise [57], simple exponential, and SPeAD-M86b analytical models. The altitude step size for all analytical models was 5 km and the modulus of the relative error was calculated using the equation below.

$$\%error = \frac{|\rho_a - \rho|}{\rho_a} * 100\% \tag{3.5}$$

where ρ is the atmospheric density calculated by SPeAD-M86, and ρ_a is the density from each of the other three analytical models. This equation quantifies the variation of the SPeAD-M86b, CIRA72, and simple exponential models from the SPeAD-M86

[1]COSPAR Working Group, COSPAR International Reference Atmosphere - 2012 Version: 1.0. 2012 (accessed May 12, 2014).http://sol.spacenvironment.net/CIRA-2012

model. In addition to direct density comparisons, the stability and accuracy of the model was verified in an orbit and attitude propagator implemented using the MATLAB-Simulink environment. A 3U CubeSat in LEO with no active control system was considered for this study. The initial classical orbital elements of the satellite with respect to Earth were defined as follows.

$$[a_0 \quad e_0 \quad i_0 \quad \Omega_0 \quad \omega_0 \quad \nu_0] = [6878 \text{ km} \quad 0.05 \quad 0.1^\circ \quad 270^\circ \quad 90^\circ \quad 0^\circ]$$

These elements correspond to a 156.1 km $\times$ 843.9 km orbit having a period of 94.6 minutes, with a low inclination of 0.1° to the equatorial plane of the Earth. Such an "atmosphere grazing" orbit could result from a failed circularization burn by the launch/deployment vehicle. Though unlikely for a CubeSat, it provides a scenario in which the atmospheric drag has a significant effect on the spacecraft near perigee. Furthermore, it allows the SPeAD-M86 model to be evaluated over a wide range of altitudes. To obtain consistent and unbiased results, it was necessary to implement the model in a variety of simulation environments. For each environment, the time variation of the orbital elements using the SPeAD-M86 model was compared with the variation computed using the NRLMSISE-00 model available in Simulink. The NRLMSISE-00 model was selected as a baseline as it has been recommended by NASA[2] to become the standard for use in satellite orbit prediction [66]. Table 3.1 presents the simulation environments utilized for validation, along with the variable parameters. All of the environments included the effects of gravity gradient torque, atmospheric drag, and atmospheric torque, while utilizing the ode3 (Bogacki-Shampine) integrator for orbit propagation over a simulation time of 86,400 seconds.

[2]Picone, J. M., Drob, D. P., Meier, R. R., and Hedin, A. E., NRLMSISE-00: A New Empirical Model of the Atmosphere. NRL Review, 2003 (accessed May 26, 2014). http://www.nrl.navy.mil/research/nrl-review/2003/atmospheric-science/picone/

Parameter	Env. A	Env. B	Env. C	Env. D
Time step, Δt (sec)	0.1	0.1	1.0	1.0
Gravity model	Spherical	Zonal	Spherical	Zonal
Order of gravity model	8	2	8	2
Attitude propagation	Quaternion	STM	Quaternion	STM
Integrator	ode3	Forward Euler	ode3	Forward Euler

Table 3.1: Summary of Simulation Environments

Environment A models the perturbations in LEO to a high degree of accuracy, while B is a simplified model designed to reduce CPU run-time. A State Transition Matrix (STM) technique, as presented by Whitmore [69], was implemented (see Sec. 5.4.2) for attitude propagation in B and D, thereby reducing the number of continuous integrations necessary. This was expected to reduce CPU run time, the results of which discussed further in Sec. 3.4.3. The forward Euler integration was not implemented into the orbit propagator due to observed instability and sensitivity to initial conditions. Environments C and D are identical to A and B respectively, differing only by the use of a larger time step. All were run on a single logical thread of an Intel®Core™ i7 2700K processor with 8 GB available memory. Geomagnetic index data, required as an input for the NRLMSISE-00 model, was obtained from NOAA[3] for January 4, 2014. Average solar flux index $F_{10.7a}$ and daily solar flux index $F_{10.7}$ data was obtained for May 15, 2014 as observed and derived from the GPS IONO model[4]. In addition, anomalous oxygen calculations were switched on for the NRLMSISE-00 model.

[3]NOAA, NOAA National Geophysical Data Center FTP Site.2014 (accessed May 26, 2014). ftp://ftp.ngdc.noaa.gov/STP/GEOMAGNETIC_DATA/INDICES/KP_AP/2014

[4]NorthWest Research Associates, Inc. Space Weather Services, 10.7 cm Solar Radio Flux. 2014 (accessed May 15, 2014). http://www.nwra.com/spawx/f10.html

The orbital element error for the first five orbital elements (a, e, i, Ω, ω) was calculated as a percentage error using the NRLMSISE-00 data as the baseline. However, due to the cyclic nature of ν from 0 to 2π rad., such a quantification would not provide a clear representation of the actual error. Instead, the smallest difference in ν at each time step was calculated, and normalized against a full orbit (i.e. 2π rad). The MATLAB code for this error formulation is presented in Listing A.1 in Appendix A. In addition to examining the accuracy and sensitivity of the SPeAD-M86 model, it was essential to quantify and compare the CPU run time against a NRLMSISE-00 baseline. The run times were obtained using the `tic` - `toc` functions in MATLAB, and averaged over 10 simulation runs per model in each environment.

3.4 Results

This section summarizes the results of the validation routines outlined in Sec. 3.3. Note that the actual orbital elements for each density model and simulation environment are not directly relevant towards the validation of the SPeAD-M86 model, but are included in the appendix for completeness.

3.4.1 Comparison with Empirical Data

Figure 3.1 presents the SPeAD-M86 model alongside published density data from several empirical models.

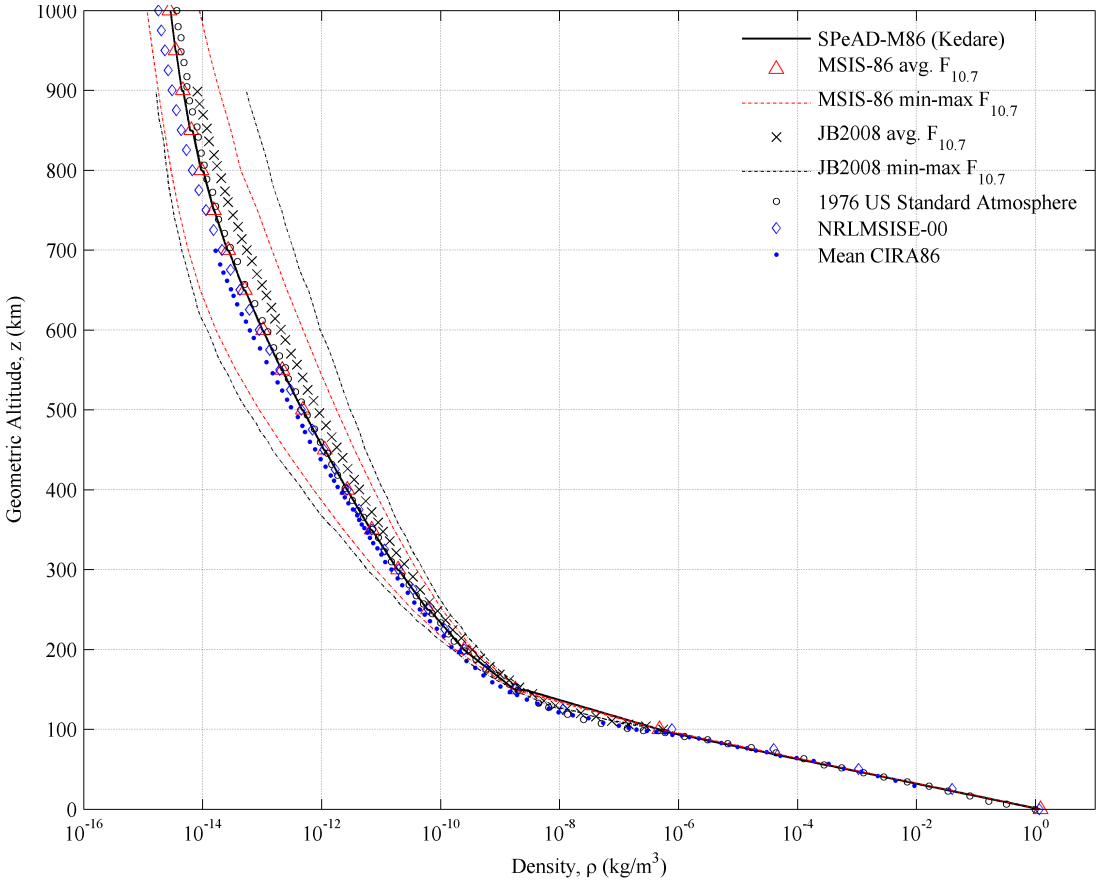


Figure 3.1: Comparison of SPeAD-M86 data against published empirical atmospheric density data.

As illustrated, the SPeAD-M86 data closely matches published empirical data from a number of models. The SPeAD-M86 data passes approximately halfway between the minimum and maximum density estimate boundaries provided by the MSIS-86 and JB2008 models. It therefore provides a good one-to-one approximation of the mean density. The even scatter of empirical data points about the SPeAD-M86 dataset is an indicator of a good correlation with previously observed atmospheric conditions. Between 100 km and 150 km, a minor deviation from published data is observed, with the SPeAD-M86 model slightly overpredicting the density. However, without continuous propulsive maneuvers below an altitude of 150 km, atmospheric drag rapidly decays the orbit of a satellite. Useful CubeSat orbits are typically higher

than this altitude, and thus the departure from published data is not considered to diminish the overall usefulness of the model. However, as a result of this discrepancy, care should be taken to ensure the model is not utilized for atmospheric re-entry predictions.

Figure 3.2 presents the density variation between the SPeAD-M86 model and other analytical models. Due to the close empirical agreement of the SPeAD-M86 model presented in Figure 3.1, the error graph provides an estimate of the error between these models and published empirical data.

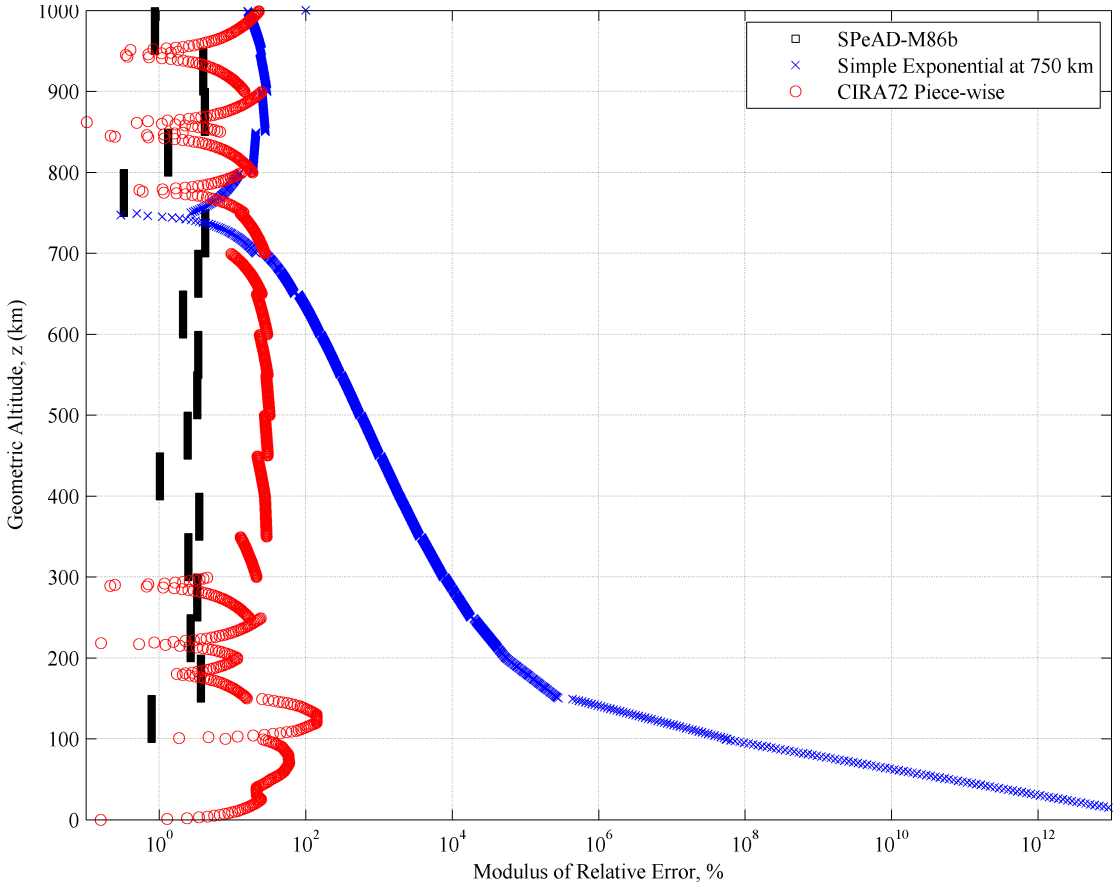


Figure 3.2: Relative error of select analytical models against SPeAD-M86 data.

The variation of the SPeAD-M86b model is less than 10% for the entire altitude range under consideration, which was anticipated as it used the same calibration

data set as the final SPeAD-M86 model. The CIRA72 piece-wise model deviates significantly from the SPeAD-M86 model, with the error varying between 1% and 60% for the majority of altitudes. Between 300 km and 700 km, the CIRA72 model exhibits a severe deviation from the SPeAD-M86 baseline. It is only at altitudes above 750 km that it provides an acceptable estimate of density. The simple exponential model herein designed specifically for an altitude of $\approx$ 750 km, exhibits a very large relative errors for altitudes below 725 km and above 775 km. Therefore, this model, though extremely simple, would not provide useful estimates of atmospheric density in orbits with significant eccentricity.

3.4.2 Orbital Element Error

The orbital elements were obtained from each simulation environment, and a comparison made between the SPeAD-M86 and NRLMSISE-00 implementations, with the latter used as the baseline. Figure 3.3 presents the error between the orbital elements for SPeAD-M86 and NRLMSISE-00 implementations in Environment A.

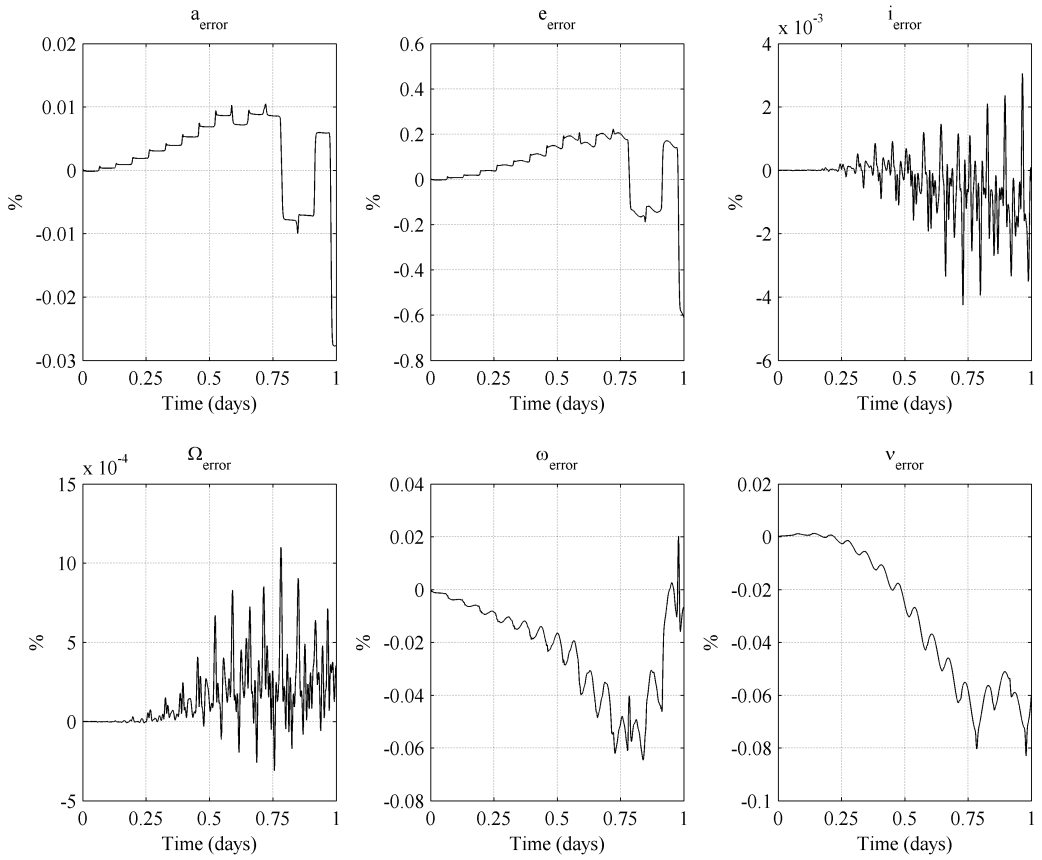


Figure 3.3: Environment A: Orbital element error between SPeAD-M86 and NRLMSISE-00 implementation.

The largest orbital element error is less than 1% for the duration of the simulation, and the elements relating to the shape of the orbit (a, e) show good correlation with the NRLMSISE-00 implementation. The elements relating to the orientation of the orbit (i, Ω, ω) also appear to closely follow the NRLMSISE-00 baseline. The error appears to increase in an oscillatory manner over time, particularly for i and Ω. Finally, the location of the spacecraft in the orbit, ν, shows an error of less than 0.1%. An examination of the time variant density from both atmospheric models, presented in Figure 3.4 and 3.5, provides further insight into the source of these small errors.

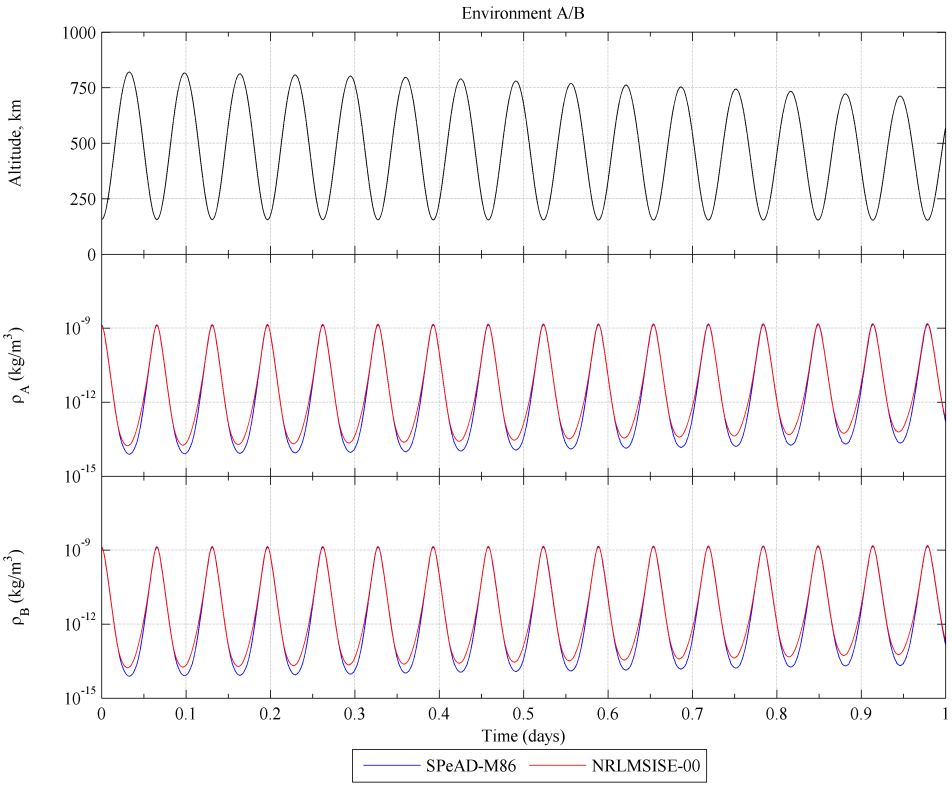


Figure 3.4: Density variations in Environments A and B with time; $\Delta t = 0.1$ second. *Altitude data from NRLMSISE-00 implementation in Environment A.*

Figure 3.4 presents the density variations for Environments A and B, alongside altitude data from the NRLMSISE-00 implementation in Environment A. This environment is considered the most accurate representation of the orbit and attitude dynamics, as it includes perturbation effects due to spherical gravitational harmonics. The variation in altitude over the course of an orbit significantly affects the atmospheric density which the spacecraft experiences during an orbit, as seen in Figure 3.4. Over one orbit, the density varies by five orders of magnitude, from 10^{-9} to 10^{-14} kg/m^3. The SPeAD-M86 model slightly under-predicts the density at higher altitudes, resulting in the a, e and ν errors. This underprediction is a result of the SPeAD-M86 model not accounting for the interaction of solar flux and geomagnetic field with the upper atmosphere. Over the simulation time of one day, the apogee

is reduced from 843.9 km to 712.2 km, while the perigee is reduced from 156.1 km to 153.0 km. Figure 3.5 presents the density variations for Environments C and D alongside altitude data from the NRLMSISE-00 implementation in Environment C.

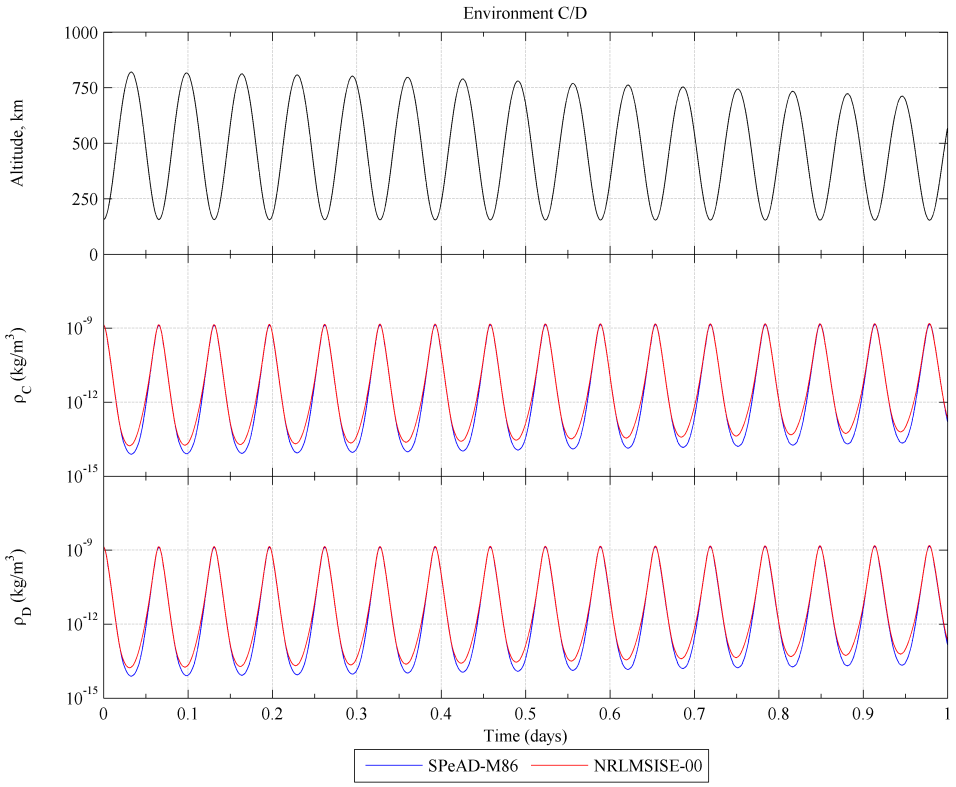


Figure 3.5: Density variations in Environments C and D with time; $\Delta t = 1.0$ second. *Altitude data from NRLMSISE-00 implementation in Environment C.*

From Figure 3.5, the apogee is reduced from 843.9 km to 711.7 km, while the perigee is reduced from 156.1 km to 153.0 km. The results are similar to those presented in Figure 3.4, with the SPeAD-M86 model slightly underpredicting the density at altitudes above 500 km. The increased Δt in Environments C and D does not appear to significantly affect the orbit propagation, indicating that the results are independent of the time steps considered in this study. Figure 3.6 presents the orbital element error for Environment B.

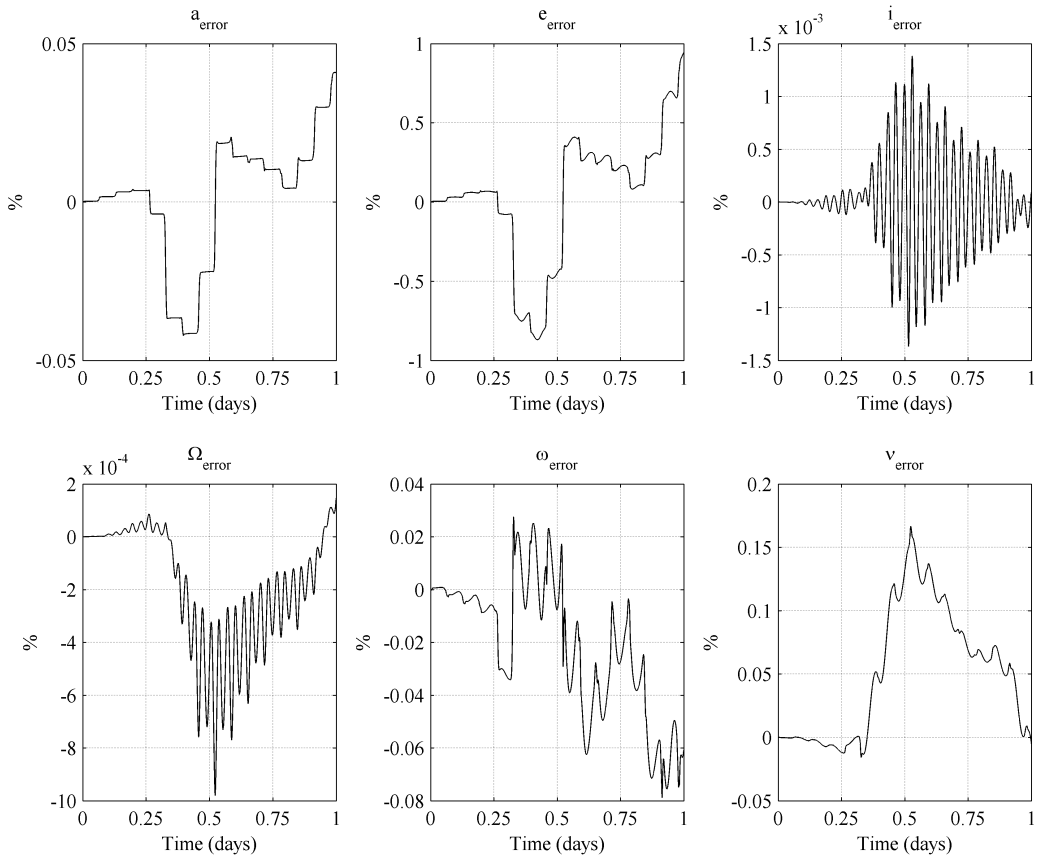


Figure 3.6: Environment B: Orbital element error between SPeAD-M86 and NRLMSISE-00 implementation.

Figure 3.6 illustrates that the errors in a, e, and ν are slightly larger than those observed for Environment A. However, all orbital element errors are less than 1% over the entire simulation period. The oscillatory nature of errors in i and Ω is visible, along with an interesting trend showing the error in ν increasing up to a maximum of 0.16% and then decreasing. Figure 3.7 presents the orbital element error for Environment C.

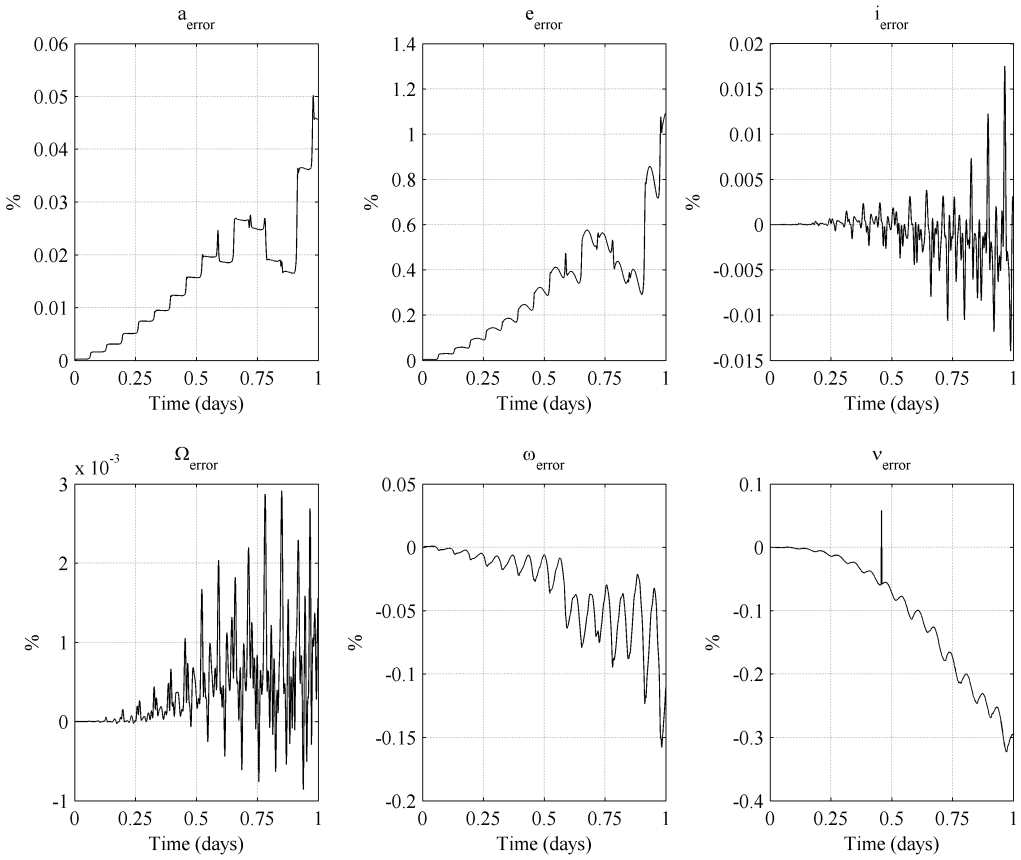


Figure 3.7: Environment C: Orbital element error between SPeAD-M86 and NRLMSISE-00 implementation.

Trends similar to those presented for Environment A and B are observed. However, a key distinction is the increase in i_{error} by an order of magnitude as compared to Environment A. In addition, the eccentricity error peaks at just over 1%. The spike in ν_{error} at ≈ 0.5 days is likely an anomaly caused by intermittent sensitivity of the propagation model to the time step, and does not affect the overall trend. Ω and i demonstrate oscillatory increasing errors. The underprediction of density presented in Figures 3.4 and 3.5 effectively reduces the circularization rate of the orbit as compared to the NRLMSISE-00 model, resulting in the positive error in e. The orbital element errors for Environment D are presented in Figure 3.7.

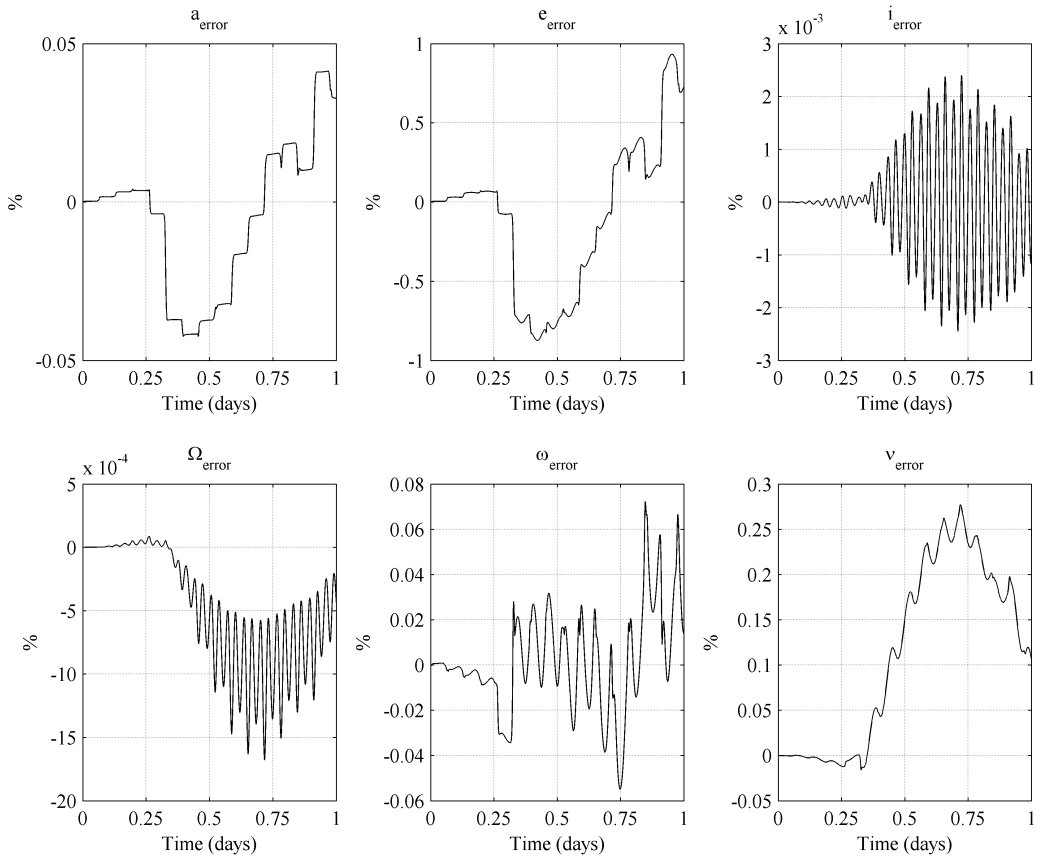


Figure 3.8: Environment D: Orbital element error between SPeAD-M86 and NRLMSISE-00 implementation.

Being the simplest and least computationally demanding simulation, Environment D was expected to exhibit the greatest effect on the accuracy of the SPeAD-M86 model. However, the results are similar to the values and trends presented for Environment B. The ability to propagate the orbit with reasonable accuracy while using a simplified simulation environment and a large time step verifies the stability of the SPeAD-M86 model.

3.4.3 CPU Run-time

In addition to examining the accuracy and sensitivity of the SPeAD-M86 model to orbit propagation environments, it was important to compare the CPU run time

against a NRLMSISE-00 baseline. These results are summarized in Table 3.2.

Environment	Atmospheric Model	Average Run Time (sec)	Run Time Difference
A	NRLMSISE-00	244.9	↓ 27.6%
	SPeAD-M86	177.2	
B	NRLMSISE-00	174.8	↓ 18.5%
	SPeAD-M86	142.8	
C	NRLMSISE-00	26.83	↓ 27.0%
	SPeAD-M86	19.58	
D	NRLMSISE-00	18.52	↓ 17.2%
	SPeAD-M86	15.33	

Table 3.2: Simulation CPU Run Times

For all four environments, the use of the SPeAD-M86 model results in a substantial decrease in run time over the NRLMSISE-00 model, ranging from 17.2% to 27.6%. The effect is particularly significant in Environments A and C, indicating that use of SPeAD-M86 model in high fidelity simulation environments could prove particularly beneficial. Furthermore, the run-time reduction appears to be independent of the simulation time step. Results indicate that Environments B and D, which were specifically designed to be computationally lightweight, require an average of 25% percent less CPU run-time compared to Environments A and C.

3.5 Summary of SPeAD-M86 Model

In this chapter, a low computation atmospheric density model for CubeSat applications was developed and validated using a piece-wise exponential approach. Unlike

existing atmospheric models, SPeAD-M86 presents a one-to-one relation between altitude and density for altitudes between 0 and 1000 km based on previously published empirical data. The proposed density correlation closely agrees with data from previously published empirical models. Propagation of spacecraft orbit and attitude agrees closely with a NRLMSISE-00 baseline, thus establishing the stability and validity of the SPeAD-M86 model. Furthermore, the proposed model reduces CPU run time by up to 27% while maintaining a daily propagation error of less than 1%, as compared with the NRLMSISE-00 model, making it suitable for on-board orbit and attitude propagation on space platforms with limited computational power, such as CubeSats.

Chapter 4

Passive Drag Sail Control

Having developed a simple, yet accurate, atmospheric model capable of significantly reducing the computational requirements for orbit propagation, we validate the stability and applicability of the SPeAD-M86 model in a complex real-world simulation environment similar to that described in Sec. 2.3. The validation builds upon the passive control techniques presented in Sec. 1.3.5, and focuses on the evaluation of undamped aerostabilization of a 3U CubeSat equipped with drag sails.

4.1 Previous Research

Early research on the effects of the atmosphere on the rotational dynamics of a spacecraft dates back to the late 1950s and early 1960s [52, 70]. These studies concluded that passive aerodynamic stability was achievable at around 300 miles altitude and below, and that aerodynamic stabilization resulted in oscillatory behavior. The first on-orbit demonstration of aerostabilization of satellites was performed by the Soviet Union in 1967 (Cosmos 149) and 1970 (Cosmos 320) [71]. These spacecraft utilized extended aerodynamic skirt stabilizers, and were nicknamed "space arrows". In 1996, the United States flight tested the Passive Aerodynamically Stabilized Magnetically

Damped Satellite (PAMS) on STS-77, demonstrating the feasibility of magnetically damped aerostabilization [72–74]. The PAMS experiment relied on offsetting the center of mass to one end of the satellite to achieve an aerodynamically stable configuration. Such an offset, however, is not allowed by the CubeSat Standard[1], and it was apparent that further research would be necessary to demonstrate the feasibility of aerostabilization for CubeSats.

Studies conducted at the University of Kentucky Space Systems Laboratory investigated the conditions required for stability of a 3U CubeSat with deployable side panels [75]. In 2010, the QbX spacecraft pair, developed by the US Naval Research Laboratory, successfully demonstrated the feasibility of aerodynamic stabilization and passive velocity-vector pointing for a 3U CubeSat at an altitude of 300 km [76][2]. However, the QbX satellites required a complex damping system utilizing active actuators. To date, there have been no flight demonstrations of completely passive solutions for CubeSat stability.

Building on this research, in 2012, Rawashdeh and Lumpp [77] presented a completely passive solution for 3U and 1U CubeSat stabilization. Their study focused on the use of two aerodynamically stable designs - one based on the Pumpkin Colony-1 bus by Pumpkin Inc. and the other incorporating flexible spring steel drag fins in a 1U CubeSat bus. Both solutions utilized magnetic methods for oscillation damping.

Significant research has also been conducted towards the development of rapid deorbit technology for the CubeSat form factor. The CanX-7 satellite, funded by DRDC-Ottawa, NSERC, and COM DEV Ltd., is currently under development at the University of Toronto's Space Flight Laboratory. The mission, expected to launch

[1]CubeSat Design Specifications, The CubeSat Project - CalPoly San Luis Obispo, 2014 (accessed Aug. 21, 2014), http://cubesat.org/index.php/documents/developers

[2]NRL Launches Nanosatellite Experimental Platforms, Naval Research Laboratory, 2010 (accessed Aug. 23, 2014), http://www.nrl.navy.mil/media/news-releases/2010/nrl-launches-nanosatellite-experimental-platforms

in late 2014, aims to deploy a 5 m^2 lightweight, modular sail capable of meeting the deorbit requirements set by the IADC [78]. DeOrbitSail, a similar mission headed by Surrey Space Center and also slated to launch in 2014 [79], proposes the use of four triangular drag sails with a combined area of 25 m^2.

4.2 Problem Statement

The passive drag focuses on the evaluation of the effect of varying the area of a CubeSat drag sail on the velocity-pointing (ram-facing) accuracy over seven days without the use of an active control system or magnetic hysteresis material for oscillation damping. Though previous research has established the oscillatory nature of aerostabilization [52,70] and examined its effectiveness with damping [77], to the best knowledge of the author, a quantitative study on the undamped stability characteristics for CubeSats has not been conducted. Four sail configurations are considered, consisting of drag sail pairs with areas of 5 m^2, 1 m^2 and 0.1 m^2, alongside a control configuration with no drag sails. The effect of these drag sails on the spacecraft orbit will also be examined for the duration of the simulation.

In addition, the effect of scaling the satellite inertia tensor on the oscillation frequency about the ram-facing direction will be evaluated. Such an inertia scaling would simulate the deployment of gravity gradient stabilization booms or scientific equipment. The original inertia tensor, $\mathbf{J}$, is scaled by factors of 0.5, 2.0 and 4.0 for a 3U CubeSat configured with a pair of 5 m^2 drag sails. Examining the ability of the drag sails to maintain the CubeSat in a ram-pointing orientation for a variety of inertia tensors will aid in evaluating the flexibility of such a stabilization technique.

Lastly, the study examines the ability of an active ACS with limited torque and

angular momentum capability to reorient the satellite from passive drag sail stabilization mode into a low drag configuration. The ACS must be capable of holding the satellite in such an orientation for a period of five days, and demonstrate a significant reduction in orbit decay compared to the passive stabilization mode. The selected maneuver is an Eigenaxis rotation to an attitude state of $[90^\circ\ 10^\circ\ 0^\circ]_{RPY}$ from passive stability mode.

4.3 Attitude Controller

To enable the spacecraft to maneuver into the Orbital Life Extension (OLE) mode, the control system must be capable of reorienting the spacecraft while continually overcoming external environmental torques. The proportional-derivative control law [13] is presented in Eq. (4.1).

$$\vec{u} = -k\mathbf{J}\tilde{q_e} - c\mathbf{J}\vec{\omega} + (\vec{\omega} \times \mathbf{J}\vec{\omega}) \tag{4.1}$$

where $\vec{u}$ is the commanded control torque, and $\tilde{q_e}$, representing the instantaneous Eigenaxis of the rotation, consists of the first three elements of the quaternion error. The quaternion error, $\mathbf{q_e}$, is calculated from the current quaternion state, $\mathbf{q}$, and a skew symmetric matrix assembled from the commanded quaternion, $\mathbf{q_c}$.

$$\mathbf{q_e} = \begin{bmatrix} q_{e_1} \\ q_{e_2} \\ q_{e_3} \\ q_{e_4} \end{bmatrix} = \begin{bmatrix} q_{c_4} & q_{c_3} & -q_{c_2} & -q_{c_1} \\ -q_{c_3} & q_{c_4} & q_{c_1} & -q_{c_2} \\ q_{c_2} & -q_{c_1} & q_{c_4} & -q_{c_3} \\ q_{c_1} & q_{c_2} & q_{c_3} & q_{c_4} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{4.2}$$

This algorithm presented in Eq. (4.1) scales the proportional and derivative gain matrices based on the inertia tensor, reducing the overshoot of the controller, and making $\mathbf{q_e}$ globally asymptotically stable. The last term, $\vec{\omega} \times \mathbf{J}\vec{\omega}$, corrects for the gyroscopic moment generated by the inertia tensor during a rotation. The gain coefficient, k, and damping coefficient, c, are functions of the desired system natural frequency and damping ratio defined as follows.

$$k = \omega_n^2 \tag{4.3a}$$

$$c = 2\omega_n\zeta \tag{4.3b}$$

In order to verify the ability of a commercially available ACS to perform the OLE maneuver, realistic torque and angular momentum limits had to be enforced upon the control system. These limits correspond to those of the MAI-201 Miniature 3-Axis Reaction Wheel system[3]. The angular momentum of the reaction wheels was calculated through propagation of the reaction wheel state equation, presented in Eq. (4.4) [80].

$$\dot{\vec{H}}_{RW} = -\vec{u} - \left(\vec{\omega} \times \vec{H}_{RW}\right) \tag{4.4}$$

where $\vec{u}$ is the applied control torque, $\vec{\omega}$ is the inertial angular velocity of the spacecraft body frame and $\vec{H}_{RW}$ denotes the reaction wheel angular momentum, which saturates at $\pm$10.8E-3 N·m·s based on the specifications of the MAI-201 system. $\vec{\omega} \times \vec{H}_{RW}$ is the gyroscopic torque generated by the total reaction wheel momentum. Furthermore, $\vec{u}$ was limited to $\pm$0.1 mN·m, 16% of the maximum torque of $\pm$0.625 mN·m that the MAI-201 can provide. Operating the reaction wheels at a lower torque could decrease wear and tear of the bearings and contact brushes, and reduce the possibility of equipment failure. The decreased torque also demonstrates the ability of the spacecraft to

[3]MAI-201 Miniature 3-Axis Reaction Wheel, CubeSatShop.com, 2014 (accessed Aug. 24, 2014)

maneuver with severely degraded reaction wheels.

4.4 Drag Sail Modelling

Figure 4.1 illustrates the spacecraft configuration for CubeSat DS-1 (CSDS-1), a hypothetical nanosatellite with deployed drag sails and gravity gradient boom.

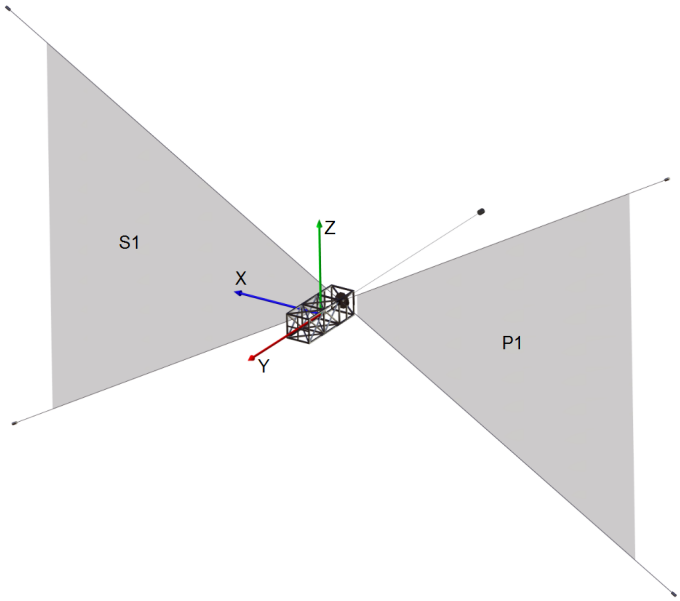


Figure 4.1: Configuration of CubeSat DS-1 with deployed drag sails and gravity gradient boom

Positive roll, pitch, and yaw correspond to counterclockwise rotations about the x-, y-, and z-axis respectively. The drag sails are labeled P1, port ($-\hat{x}$), and S1, starboard ($+\hat{x}$), in accordance with nautical convention. The drag sails, located in the aft section of the CubeSat, are deployed symmetrically about the body y-axis, providing weathervane stability to the satellite.

From Eqs. (2.2), (2.3), and (2.10), accurate modelling of aerodynamic perturbations requires knowledge of the variation in the projected area, as well as information regarding the location of the center of pressure location of each individual panel in

the body frame. The 10 faces of CSDS-1 were individually defined in the body coordinate frame as area vectors. The center of pressure for each of the panels was similarly specified. Table 4.1 summarizes the mathematical definition of the satellite geometry.

Table 4.1: CubeSat DS-1 Geometry Definition

Panel ID	Label	Area, A (m^2)	Area normal vector, $\hat{A}$	C_P location, $\vec{r}_{cp}$
1	Starboard panel	0.03	$+\hat{x}$	$+(b/2)\hat{x} - (l/3000)\hat{y}$
2	Fore panel	0.01	$+\hat{y}$	$+(l/2)\hat{y}$
3	Zenith panel	0.03	$+\hat{z}$	$-(l/3000)\hat{y} + (h/2)\hat{z}$
4	Port panel	0.03	$-\hat{x}$	$-(b/2)\hat{x} - (l/3000)\hat{y}$
5	Aft panel	0.01	$-\hat{y}$	$-(l/2)\hat{y}$
6	Nadir panel	0.03	$-\hat{z}$	$-(l/3000)\hat{y} - (h/2)\hat{z}$
7	S1 forward face	0.0 - 5.0*	$+\hat{y}$	$+d_x\hat{x} - (l/2)\hat{y}$
8	P1 forward face	0.0 - 5.0*	$+\hat{y}$	$-d_x\hat{x} - (l/2)\hat{y}$
9	S1 aft face	0.0 - 5.0*	$-\hat{y}$	$+d_x\hat{x} - (l/2)\hat{y}$
10	P1 aft face	0.0 - 5.0*	$-\hat{y}$	$-d_x\hat{x} - (l/2)\hat{y}$

*Simulation Variable

The dimensions of CSDS-1 are l, b, and h, while d_x represents the C_P location of the drag sails in the $\hat{x}$ direction. The values of these spacecraft parameters are presented later in Sec. 4.5.2.

4.5 Drag Sail Simulation Setup

Simulink-MATLAB was selected for the creation of a simulation environment due to its customizability, modularity, and ease of debugging. The Agilent STK software,

though popular for orbit and attitude propagation, does not allow for straightforward implementation of time-variant projected areas. The Smart Nanosatellite Attitude Propagator (SNAP) tool [75] was considered as an option, as it had been extensively verified and utilized in related CubeSat simulations [46]. However, closer examination of the SNAP environment indicated that the effects of atmospheric drag forces on the spacecraft position and velocity were not considered[4]. Figure 4.2 visualizes the data flow between each of the simulation environment components described in the previous subsections and illustrates the interdependence of the various modules.

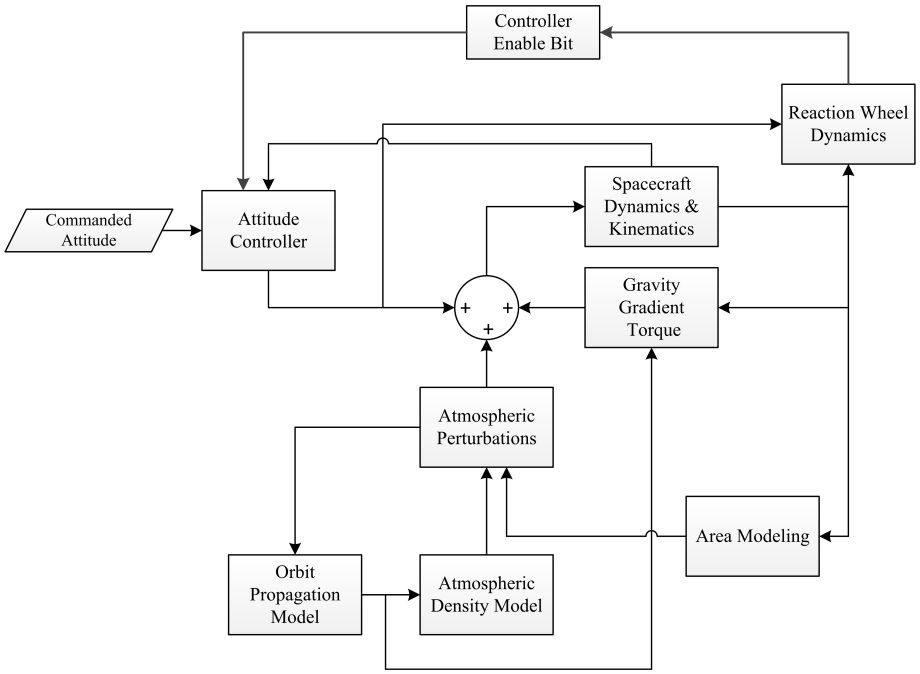


Figure 4.2: Overview of Simulation Environment Structure

The spacecraft dynamics and kinematics block forms the core of the simulation environment, alongside the orbit propagator. The output states from these blocks drive the environmental perturbations which in turn affect the states. The reaction wheel dynamics block models momentum storage, limiting the maximum torque and

[4]Simulation Tools: Smart Nanosatellite Attitude Propagator (SNAP), University of Kentucky Space Systems Laboratory. http://ssl.engineering.uky.edu/simulationtools/snap/

relaying an interrupt to the controller through an Enable Bit if the maximum momentum capacity is exceeded. All simulations for this study utilized the real-world environment as defined in Sec. 2.3 and were executed on parallel threads of an Intel®Core™i7 2700K processor with 16 GB available memory.

4.5.1 Initial Conditions

The initial classical orbital elements of CubeSat DS-1 with respect to Earth were defined as follows.

$$[a \quad e \quad i \quad \Omega \quad \omega \quad \nu]_0 = [6978 \text{ km} \quad 0.005 \quad 0.1^\circ \quad 270^\circ \quad 90^\circ \quad 0^\circ]$$

These elements correspond to a 565.1 km $\times$ 634.9 km orbit having a period of 96.7 minutes, with a low inclination of 0.1° to the equatorial plane of the Earth. Such a near-circular orbit is typical for CubeSats, though at a higher altitude compared to prior research. The initial attitude states, $\mathbf{q_0}$ and $\boldsymbol{\omega_0}$, of the satellite are presented below.

$$\mathbf{q_0} = [0 \quad 0 \quad 0 \quad 1]^T \qquad \boldsymbol{\omega_0} = [0 \quad 0 \quad 0] \text{ rad}\cdot\text{s}^{-1}$$

This corresponds to a satellite with no angular rates aligned with the Earth Centered Inertial (ECI) reference frame.

4.5.2 Satellite Properties

CSDS-1 was defined as having a mass of 4.0 kilograms in LEO, and equipped with an on-board 3-axis active attitude control system alongside deployable drag sails. The satellite dimensions were defined as $l = 0.03$ m, $b = 0.01$ m, and $h = 0.01$ m. The C_P x-location for each drag sail was set as $d_x = 1.0$ m. For rotational stability, $I_{zz} > I_{xx}, I_{yy}$ or $I_{zz} < I_{xx}, I_{yy}$. Such an inertia matrix allows for rotational stability about the axis of maximum or minimum moment of inertia respectively [54]. The

satellite inertia tensor **J**, obtained using the *Mechanical Properties* function in CATIA v5, is now presented for the satellite with drag sails and a single deployable mass along the y-axis.

$$\mathbf{J} = \begin{bmatrix} 0.059 & 0 & 0 \\ 0 & 0.114 & 0 \\ 0 & 0 & 0.126 \end{bmatrix} \text{ kg} \cdot \text{m}^2$$

In addition to the geometric and inertial properties of the satellite, it was necessary to define the controller characteristics. For the purposes of this work, a damping ratio, $\zeta = 1$ (the critically damped case), and an arbitrary natural frequency $\omega_n = 0.35$, were selected.

4.6 Drag Sail Simulation Results

Having defined the satellite properties and initial conditions, the simulation results based on the criteria outlined in Sec. 4.2 can be presented.

4.6.1 Effect of Drag-sail Area on Ram-pointing Behavior

Figure 4.3 presents the variation of ram-pointing error over seven days for four drag sail areas. Γ_{RAM} is the angle between the velocity vector and forward face of CSDS-1.

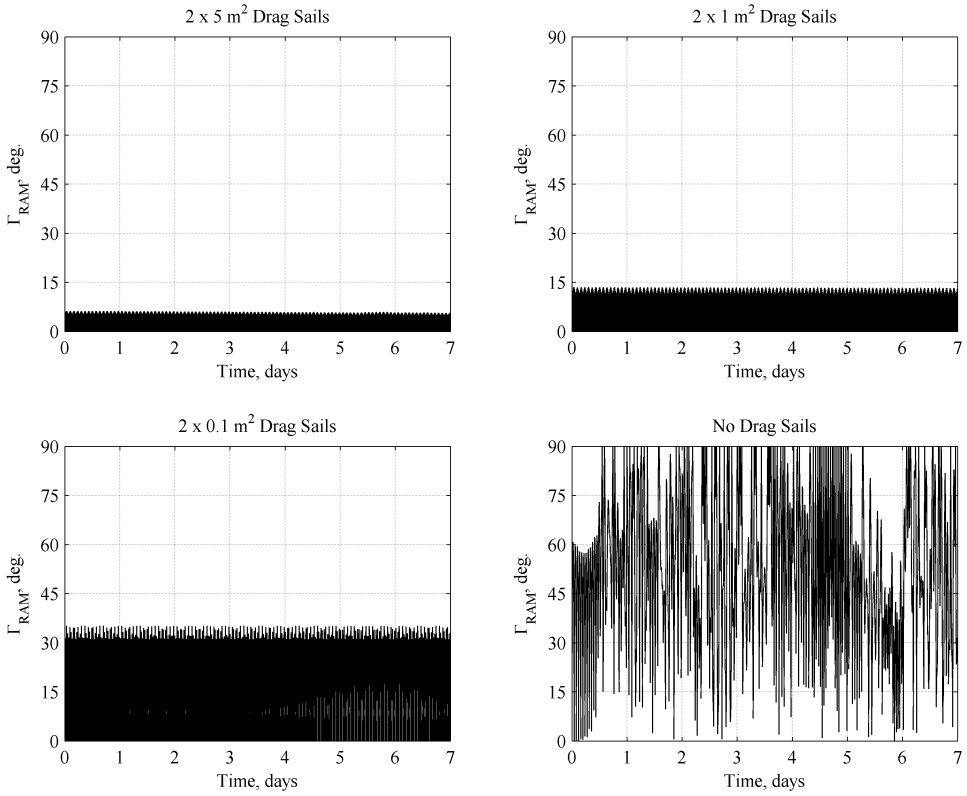


Figure 4.3: Effect of drag sail area on ram-pointing accuracy of CSDS-1

The presence of a pair of drag sails symmetric about the y-axis of CSDS-1 prevents the development of uncontrolled oscillations about the ram-pointing orientation. Without drag sails, the spacecraft enters uncontrolled oscillations after approximately 12 hours. The 0.1 m^2 drag sails provide a steady pointing accuracy of $\pm 32^\circ$ over seven days. 1 m^2 drag sails provide an improved pointing accuracy of $\pm 12^\circ$. As in the case of the 0.1 m^2 sails, this pointing accuracy is maintained over seven days. The largest sails considered for this study, with an area of 5 m^2, provide a pointing accuracy of $\pm 6^\circ$. The larger area exerts a greater restoring moment on the spacecraft, allowing for greater pointing accuracy.

Having established the effect of the drag sail area on pointing accuracy, it is

of interest to examine the oscillation frequency to determine the feasibility of such a system for ram-facing observations. Figure 4.4 presents the ram-pointing error for the four considered drag sail configurations over the first 240 minutes of the simulation.

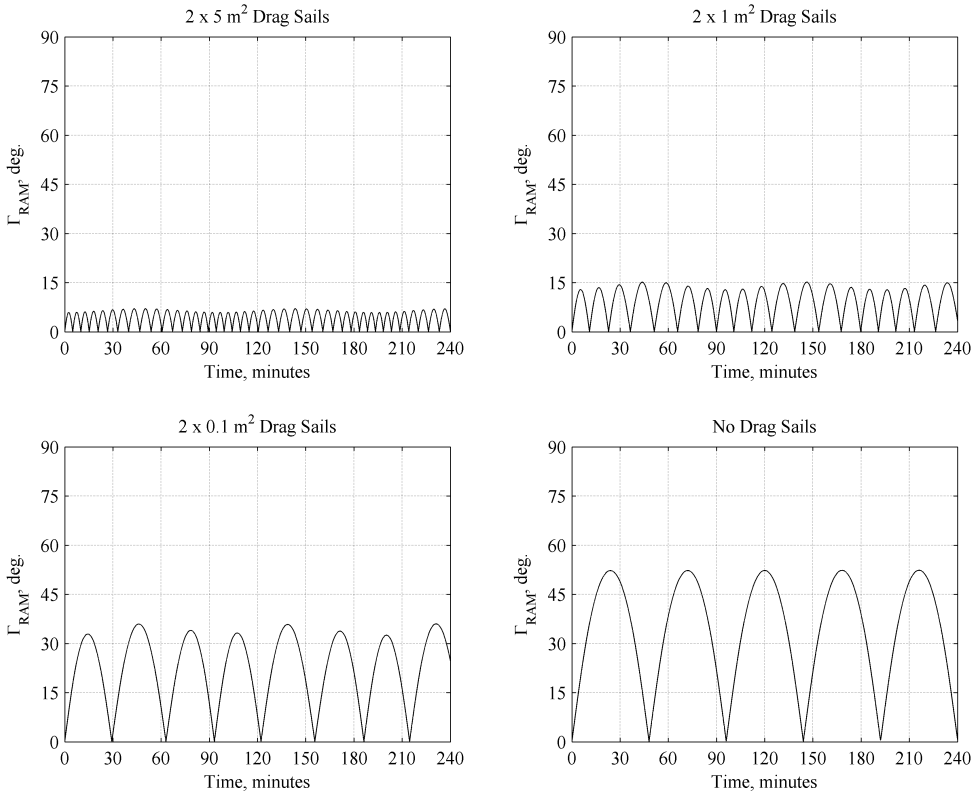


Figure 4.4: Effect of drag sail area on short-term oscillations for CSDS-1

Two oscillatory responses are observed for configurations utilizing drag sails - one short term, and one long term. It should be noted that as Γ_{RAM}, the angle between the forward face of CSDS-1 and $\vec{v}$, is calculated as an absolute value, one full oscillation is considered to be two consecutive cycles, corresponding to the spacecraft slewing to the left and right of the ram-pointing vector. The short term oscillations are due to the undamped response of the satellite to the environmental torques, while the long term variations, with a period similar to the orbital period, could be the result

of spatial variations in the atmospheric density. However, data from the current study is insufficient to conclusively support such a correlation. Interestingly, the control configuration does not appear to exhibit a well-defined long-term oscillation, supporting the possibility that the long-term oscillations are driven by ρ, a function of $\vec{r}$. Table 4.2 summarizes the effect of drag sail area on the oscillatory characteristics of aerostabilized ram-pointing.

Table 4.2: Summary of Drag Sail Area Effects on Oscillations

Drag Sail Area (m^2)	Mean Γ_{RAM} (deg.)	Mean Period (min.)	Mean Slew Rate (deg·min^{-1})
5.0	5.7	10.3	1.11
1.0	12	22.4	1.07
0.1	32	60.3	1.06
0.0	52	115	0.90

The results indicate that larger sail areas increase pointing accuracy, and reduce the period of each oscillation. The mean slew rate demonstrates a marginal increase with increasing drag sail area, but generally remains within $\pm 10\%$ of 1 deg·min^{-1} for the considered sail areas.

4.6.2 Effect of Drag-sail Area on Orbit Altitude

The atmospheric drag which this passive control technique relies upon also affects the orbit of the satellite, and it is therefore impractical to simply increase drag sail areas to achieve greater ram-pointing accuracy. Figure 4.5 presents the variation in apoapsis and periapsis over seven days for the four sail areas.

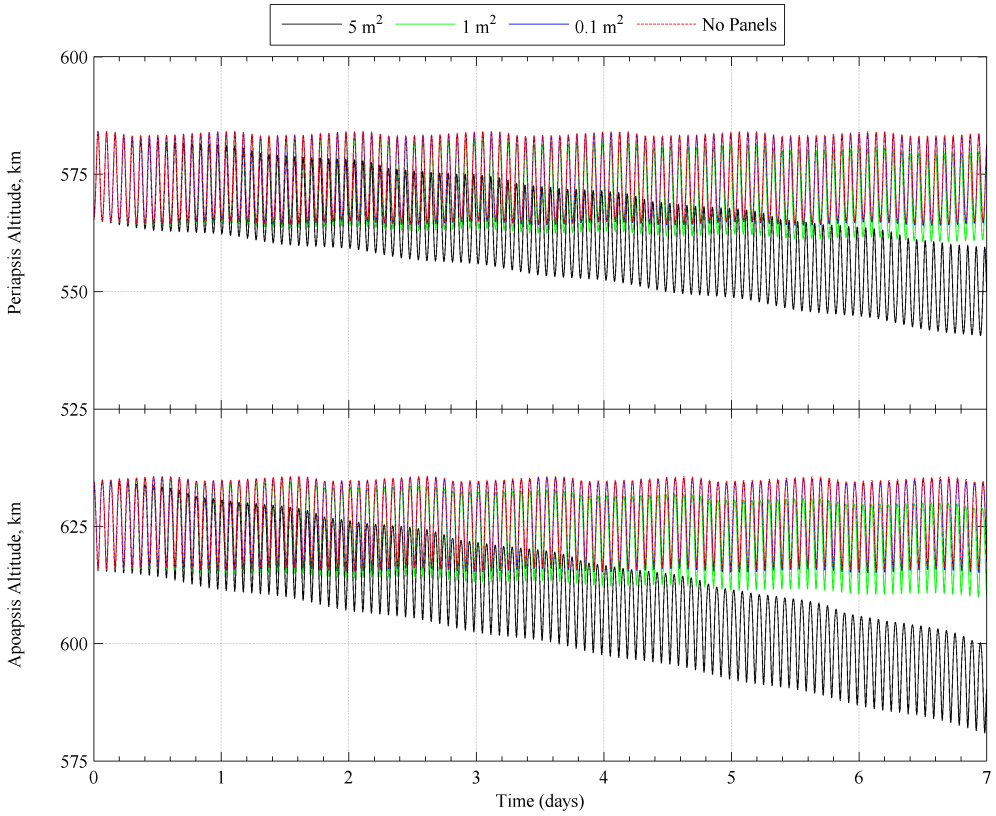


Figure 4.5: Effect of drag sail area on orbit altitude of CSDS-1

As expected, the 5 m^2 sails produce the largest drop in orbit altitude with time due to the greater drag force experienced by the satellite. Over seven days, the mean periapsis is reduced by ≈25 km, while the mean apoapsis drops by ≈35 km. Such sails would be useful if rapid de-orbit of a spacecraft is a requirement. Meanwhile, over seven days, the 1 m^2 sail reduces the mean periapsis and apoapsis by ≈5 km and 6 km respectively. In comparison, the 0.1 m^2 sails reduce the mean periapsis and apoapsis by less than 200 m. The information and methodology presented here is sufficient to allow mission planners to perform a preliminary trade study based on the pointing accuracy, oscillation period, and orbital decay rate.

4.6.3 Effect of Satellite Inertia on Ram-pointing Behavior

Having examined the effects of the sail areas on satellite pointing accuracy, we now investigate the adaptability of undamped aerostabilization to different inertia tensors. Figure 4.6 presents the ram-pointing error of four inertia configurations over a period of 240 minutes for CSDS-1 equipped with a pair of 5 m^2 drag sails.

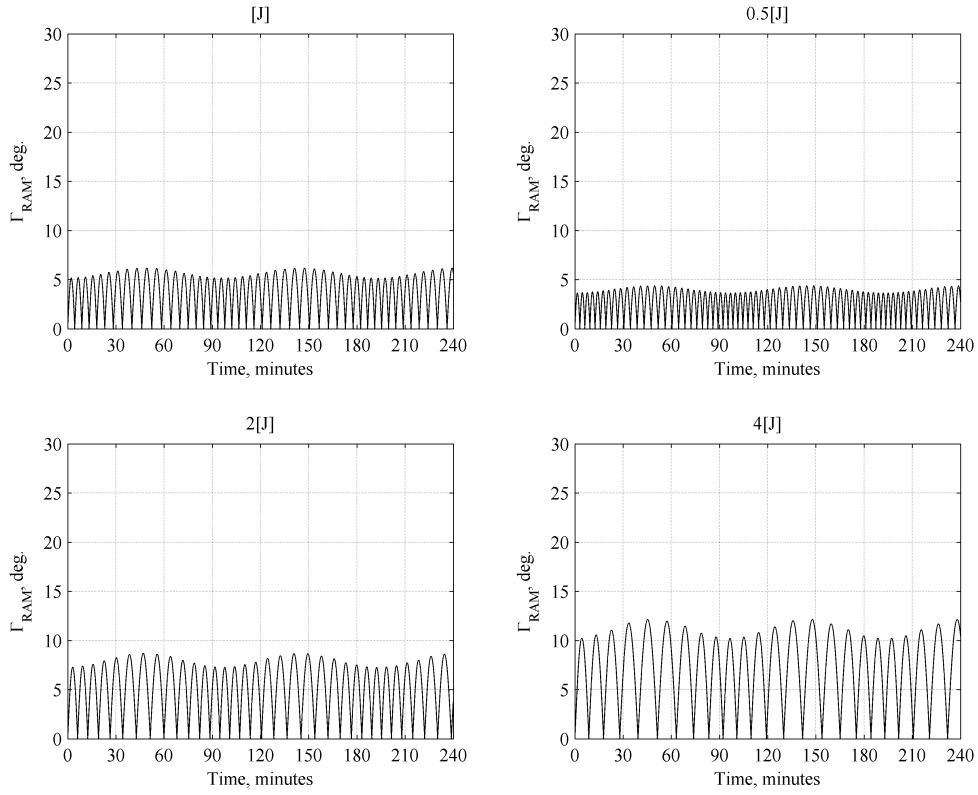


Figure 4.6: Effect of CubeSat inertia scaling on Oscillation Frequency of CSDS-1

Scaling the inertia tensor upwards results in a reduction in pointing accuracy, and an increase in the oscillatory period. Downscaling of **J** produces the opposite effect, i.e., an increase in pointing accuracy, and a decrease in oscillatory period. This agrees with the expected response, as an increased inertia offers greater resistance to external torques. Interestingly, inertia tensor scaling appears to affect the short-term

oscillatory response without influencing the long term response. Table 4.3 summarizes the results.

Table 4.3: Summary of Inertia Scaling Effects for CSDS-1

Inertia Tensor	Mean Γ_{RAM} (deg.)	Mean Period (min.)	Mean Slew Rate (deg·min^{-1})
1.0**J**	5.7	10.3	1.11
0.5**J**	4.04	7.25	1.11
2.0**J**	7.99	14.4	1.11
4.0**J**	11.2	20.1	1.11

Results are for CSDS-1 with 5 m^2 drag sails

4.6.4 Evaluation of Orbit Life Extension (OLE) Mode

Figure 4.7 presents the variation in orbit altitude over 7 days, with the implementation of the Orbit Life Extension (OLE) mode from Day 2 onwards.

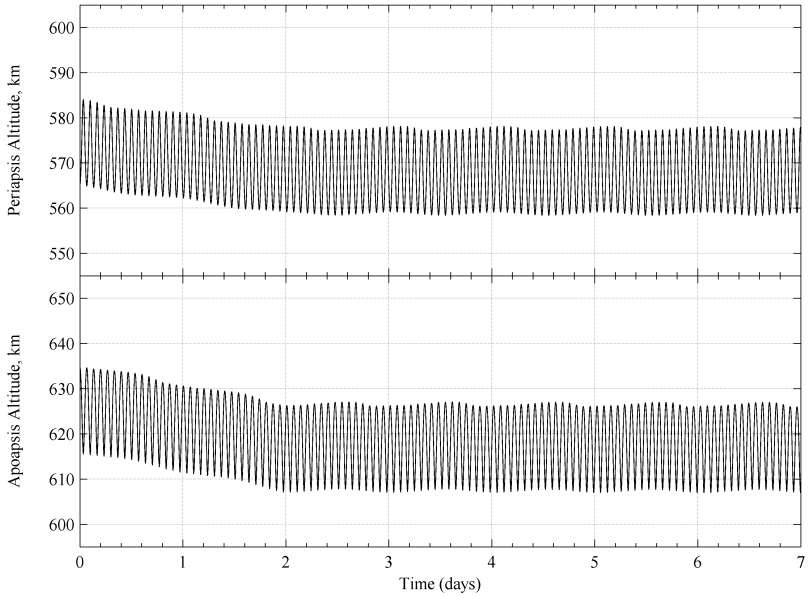
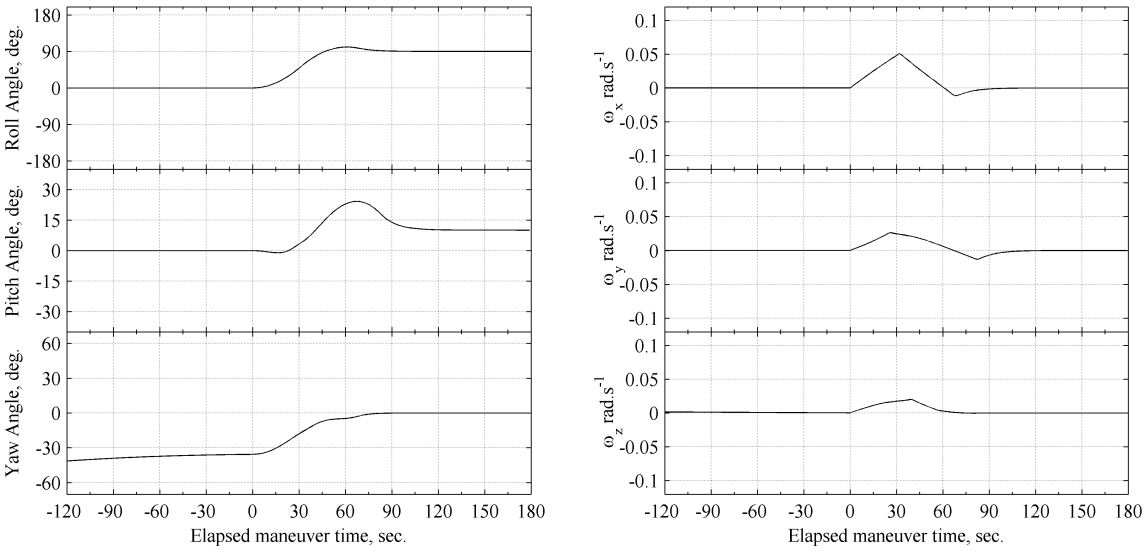


Figure 4.7: Effect of OLE Maneuver on orbit altitude of CSDS-1

Over the first two days of the simulation, the mean periapsis and apoapsis drop by 5 km. The satellite maintains a ram-pointing orientation during this time frame, as established in the results presented earlier. Following the execution of the OLE maneuver, the satellite stabilizes in a 570 km $\times$ 618 km orbit. The orbital elements for the seven day period presented in Figure 4.7 are included in Figure B.1 in Appendix A. The results demonstrate that the OLE maneuver can successfully reduce the orbital decay rate of CSDS-1. Figure 4.8 presents the attitude states for a time window spanning T-120 and T+180 seconds from the activation of the control system. The spacecraft successfully performs an Eigenaxis maneuver from a passive aerostabilized orientation to $\begin{bmatrix}90^\circ & 10^\circ & 0^\circ\end{bmatrix}_{RPY}$ in 135 seconds.



(a) RPY angles

(b) Angular rates

Figure 4.8: Attitude states: OLE maneuver

Figure 4.9 presents the applied torque and angular momentum of the x-, y-, and z-axis reaction wheels of the spacecraft.

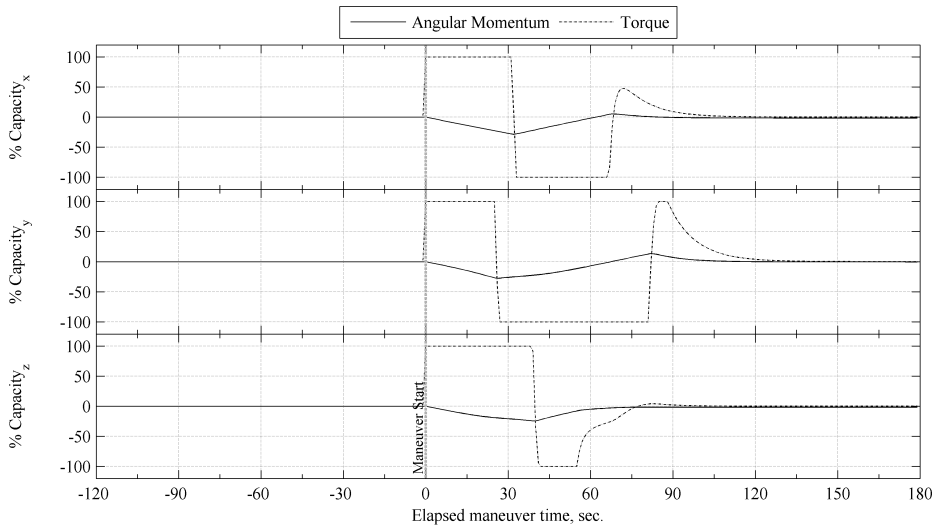


Figure 4.9: Actuator usage: OLE Maneuver

The allowable torque and angular momentum of the attitude control system remain within the allowable envelope during OLE maneuver. With the torque limited to ± 0.1 mN·m, the controller is capable of overcoming environmental torques, and maneuvers the spacecraft to the commanded attitude. The maximum instantaneous angular momentum capacity utilized during the maneuver was 28.7%, 27.7%, and 24.1% in the x-, y-, and z-axis respectively. Over five days, the spacecraft utilized 12.0%, 0.11%, and 5.74% of the momentum capacity to counteract environmental disturbance torques and maintain a $[90^\circ\ 10^\circ\ 0^\circ]_{RPY}$ orientation. Figure B.2 in Appendix B provides the angular momentum usage for the entirety of the 7-day period.

4.7 Summary of Drag-Sail Study

This chapter examined the undamped passive attitude characteristics of a 3U CubeSat, referred to within the text as CubeSat DS-1, equipped with deployable drag sails. Two variables – the drag sail area and the satellite inertia tensor **J** – were

varied separately to investigate their effects on the ram-pointing accuracy and associated oscillatory response. Larger drag sails demonstrated tighter pointing accuracies, up to a maximum of 5.7° from the ram-direction, and decreased oscillation periods, at the cost of faster orbital decay. Upscaling of the inertia tensor decreased the pointing accuracy and increased the oscillation period. However, the reduction in pointing accuracy was minimal, with a 4× inertia scaling corresponding to a pointing accuracy 11.2°, a decrease of 6° from the nominal inertia. In addition to aerostabilization characteristics, an Orbital Life Extension maneuver was analyzed. The simulations indicate that the spacecraft is capable of entering and maintaining a low-drag configuration for five days while utilizing a maximum of 12.0% of available angular momentum capacity to counter environmental disturbances. Furthermore, these results establish the validity and stability of the SPeAD-M86 model and the real-world simulation environment.

Chapter 5

Optimal Guidance Generation

The research in Chapter 3 and Chapter 4 presents a technique to reduce the computational cost for on-board orbit propagation, and verifies that environmental disturbance torques can significantly affect the attitude of a CubeSat. From these findings, we now investigate the formulation of a torque-optimal guidance algorithm as applicable to CubeSats.

5.1 Approach Overview

The torque-optimal guidance problem was defined and solved within the MATLAB-Simulink environement, coupled with GPOPS and a Sparse Nonlinear Optimizer (SNOPT). After benchmarking GPOPS to validate its trajectory optimization capabilities, a cost function based on the applied torques was implemented. An on-board orbit propagator provided the initial and final states required by GPOPS, while the controller bounding box was defined based on actuator limits, i.e., torque and angular momentum capacity.

5.2 GPOPS

The General Pseudospectral Optimal Software (GPOPS) tool was implemented alongside a spacecraft dynamics model to obtain benchmark results, and validate the capability of the MATLAB-Simulink environment to accurately optimize guidance trajectories. Following validation, GPOPS was utilized for the generation of the torque-optimal guidance trajectory. The software, which is integrated into the MATLAB environment, requires the user to define the dynamics of the problem using differential equations, an associated cost function, connections between phases (if present), and finally the limits and initial guesses for each of the states and controls [39]. Details regarding the setup and structure of the GPOPS environment are provided later in Sec. 5.2.3. The generated torque-optimal trajectory was validated using the real-world environment outlined in Sec. 2.3.

5.2.1 GPOPS Algorithm

In its general form, any optimal guidance/control problem can be formulated so as to minimize the cost function

$$J = \Phi\left(\mathbf{x}\left(t_f\right), t_f\right) + \int_{t_0}^{t_f} g\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right), t\right) dt \tag{5.1}$$

In Eq. (5.1), Φ is the Mayer and g is the Lagrange component. The system is subject to the dynamic constraints given by

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{5.2}$$

and the associated boundary conditions

$$\phi\left(\mathbf{x}\left(t_0\right), t_0, \mathbf{x}\left(t_f\right), t_f\right) = 0 \tag{5.3}$$

In this research, the optimal guidance problem defined in Eq. (5.1) - Eq. (5.3) is solved using a direct transcription method called the *Gauss pseudospectral method* [33]. Though a detailed explanation regarding the development and mathematics behind the Gauss pseudospectral method is beyond the scope of this thesis, an overview shall be provided to familiarize the reader with the associated algorithms. For a single-phase optimal control problem, the Gauss pseudospectral method can be summarized as follows. First, the original time interval $t \in [t_0, t_f]$, is transformed to the non-dimensionalized time interval $\tau \in [-1, 1]$ as

$$t = \frac{(t_f - t_0)\tau + (t_f + t_0)}{2} \tag{5.4}$$

Following the temporal transformation in Eq. (5.4), the cost function in Eq. (5.1) can be rewritten in terms of τ as

$$J = \Phi(\mathbf{x}(1), t_f) + \frac{(t_f - t_0)}{2} \int_{-1}^{1} g(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) d\tau \tag{5.5}$$

In the same manner, the dynamic constraints of Eq. (5.2) are given in terms of τ

$$\frac{2}{(t_f - t_0)} \frac{d\mathbf{x}}{d\tau} = \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) \tag{5.6}$$

The boundary conditions of Eq. (5.3) in terms of τ are given as

$$\phi(\mathbf{x}(-1), t_0, \mathbf{x}(1), t_f) = 0 \tag{5.7}$$

We approximate the state, $\mathbf{x}(t)$, in terms of a basis $N+1$ Lagrange interpolating polynomial on the interval from $[-1,1]$ as

$$\mathbf{x}(t) \approx \mathbf{X}(t) = \sum_{k=0}^{N} \mathbf{X}(t_k) L_k(t) \tag{5.8}$$

Furthermore, suppose we select the following $N+1$ points on $\tau \in [-1,1]$ for discretization of the continuous-time problem: the initial point $\tau = -1$ and N Gauss points, τ_k, $k = 1, \cdots, N$, which lie within the interval $[-1,1]$. The approximation of the state derivative at the Gauss points is then expressed as

$$\left[\frac{d\mathbf{x}}{dt}\right]_{t_i} \approx \left[\frac{d\mathbf{X}^N}{dt}\right]_{t_i} = \sum_{k=0}^{N} \mathbf{X}_k \left(\frac{dL_k}{dt}\right)_{t_i} = \sum_{k=0}^{N} D_{ik}\mathbf{X}_k = \mathbf{f}(\mathbf{X}_i, \mathbf{U}_i, t_i) \tag{5.9}$$

where D_{ik} is called the *Differentiation Matrix*. The differential operators $D \in \mathbb{R}^{N\times N}$ and $\bar{D} \in \mathbb{R}^N$ are computed through the exact derivative of the Lagrange interpolating polynomial, $L_k(t)$ as

$$\left[\frac{d\mathbf{x}}{dt}\right]_{t_i} = \dot{\mathbf{x}}(t_i) \approx \dot{\mathbf{X}}(t_i) = \mathbf{X}(t_0)\cdot \bar{D}_i + \sum_{k=1}^{N} \mathbf{X}(t_k)\cdot D_{ik} \tag{5.10}$$

where $D_{ik} = \dot{L}_k(t_i)$ and $\bar{D}_{ik} = \dot{L}_0(t_i)$ are the differential operators. The continuous-time optimal control problem can then discretized into a nonlinear programming problem (NLP) using the variables $\mathbf{X}_k \in \mathbb{R}^n$ and $\mathbf{U}_k \in \mathbb{R}^m$ for the states and control at the Gauss points respectively, for $k = 1, \cdots, N$. The initial and final states, $\mathbf{X}(t_0) \in \mathbb{R}^n$ and $\mathbf{X}(t_f) \in \mathbb{R}^n$ respectively, are also included as variables.

The continuous-time cost function in Eq. (5.5) is first approximated using a Gauss quadrature.

$$J = \Phi(\mathbf{X}(t_f), t_f) + \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k g(\mathbf{X}_k, \mathbf{U}_k, t_k) \tag{5.11}$$

where w_k, the Gauss weight at each Gauss point, is given by

$$w_k = \frac{2}{(1-\tau_k^2)[P'_N]^2} \tag{5.12}$$

In Eq. (5.12), P'_N represents the N$^{\text{th}}$-degree Legendre polynomial defining the Gauss points. The differential equation constraints in Eq. (5.6) are also discretized at the Gauss points as

$$\frac{2}{t_f - t_0}\bar{D}_i \mathbf{X}(t_0) + \frac{2}{t_f - t_0}\sum_{k=1}^{N} D_{ik}\mathbf{X}_k = \mathbf{f}(\mathbf{X}_i, \mathbf{U}_i, t_i), \quad i = 1, \cdots, N \tag{5.13}$$

where

$$\begin{bmatrix} \bar{D}_i & D_{ik} \end{bmatrix} = \begin{bmatrix} (dL_0/d\tau)_{\tau_i} & (dL_k/d\tau)_{\tau_i} \end{bmatrix} \in \mathbb{R}^{(N+1)\times(N+1)} \tag{5.14}$$

Rao and Huntington emphasize that unlike previously developed pseudospectral methods [31, 81], the differential equations are collocated only at the Gauss points, and not at the boundary points. From this, the discretized boundary conditions are given as

$$\phi(\mathbf{X}(t_0), t_0, \mathbf{X}(t_f), t_f) = 0 \tag{5.15}$$

Lastly, the terminal state, $\mathbf{X}(t_f)$, is defined using a quadrature approximation to the dynamics as

$$\mathbf{X}(t_f) = \mathbf{X}(t_0) + \frac{t_f - t_0}{2}\sum_{k=1}^{N} w_k \mathbf{f}(\mathbf{X}_k, \mathbf{U}_k, t_k) \tag{5.16}$$

The cost function of Eq. (5.11) along with the constraints of Eq. (5.13) and Eq. (5.15) define an NLP. The solution of this NLP is an approximate solution to the continuous-time optimal control problem.

5.2.2 Sparse Nonlinear Optimizer (SNOPT)

The NLP described in Sec. 5.2.1 was solved with the MATLAB `mex` interface of the NLP solver SNOPT using analytic first-order derivatives for the constraint Jacobian and the gradient of the objective function. A large number of NLP solvers, such as IPOPT [82], KNITRO [83], and MINOS [84], are available to academic and industrial users. However, SNOPT was selected based on availability and prior implementation in the GPOPS environment.

SNOPT is a general purpose system for constrained optimization. It minimizes a linear or nonlinear function subject to bounds on the variables and sparse linear or nonlinear constraints [85]. SNOPT uses a sequential quadratic programming (SQP) algorithm to solve an NLP. The search direction are obtained through QP subproblems which minimize a quadratic model of a Lagrangian function subject to linearized constraints. In order to ensure that convergence is independent of the starting point, an augmented Lagrangian merit function is reduced along each search direction.

An in-depth explanation of the SNOPT algorithm is beyond the scope of this thesis, and details regarding the implementation of SNOPT may be found in the SNOPT User's Guide [85]. However, it suffices to state that SNOPT is suitable for solving nonlinear programs of the form

$$\text{minimize } \mathbf{x} \text{ for } f_0(\mathbf{x}) \text{ subject to } l \leqslant \begin{pmatrix} \mathbf{x} \\ f(\mathbf{x}) \\ A_L\mathbf{x} \end{pmatrix} \leqslant u$$

where $\mathbf{x}$ is an n-vector of variables, l and u are constant lower and upper bounds, $f_0(\mathbf{x})$ is a smooth scalar objective function, A_L is a sparse matrix, and $f(\mathbf{x})$ is a vector of smooth nonlinear constraint functions $f_i(\mathbf{x})$. In the ideal case, the first derivatives of $f_0(\mathbf{x})$ and $f_i(\mathbf{x})$ should be coded by the user. Upper and lower bounds are specified

for all variables and constraints.

5.2.3 GPOPS Setup

Four scripts form the core of GPOPS: a `main` script; a `DAE` script, which defines the dynamics; a `connect` script; and a `cost` script. The role of each script in the GPOPS software shall now be briefly described [86].

1. `main` script: This script assembles the framework for an optimal control problem. An optimal control problem may be divided into multiple phases, with each phase having a distinctive set of states, controls, equations of motion and/or cost functions. The `main` script assembles these phases, and allows the user to specify the number of nodes (Gauss points) per phase. These nodes are then passed to an NLP solver such as SNOPT. The node count is critical as it must be high enough to capture the optimal solution, but not so high as to trigger numerical divergence or unnecessary computation.

   Each phase includes a definition for the minimum and maximum values of the time, states, and controls. To obtain an optimal solution to the dynamics defined in the `DAE` script, an educated initial guess of the solution, within the bounds previously defined, is necessary. The run time of the code is highly sensitive to these guesses.

   Finally, the `main` script calls and sets the maximum/minimum values for the equations defined in the `connect` script (if present) to link the states and controls of subsequent phases.

2. `DAE` script: This script defines the dynamics for the optimization problem. The NLP solver requires each state to be defined as a first order differential equation. The dynamics may be defined as global or phase-specific, depending on the

control optimization under consideration. In addition to the dynamics, the `DAE` script allows the user to define any number of path constraints in terms of state and control variables. The attitude dynamics and kinematics were presented in Eq. (2.6) and Eq. (2.7), while the Keplerian orbit propagator within the optimization environment shall be presented in Sec. 5.4.5. The perturbing torque formulation was presented in Sec. 2.2.3.

3. `cost` script: This script defines the cost function for the optimal guidance/control problem. It consists of two terms - the Lagrange and Mayer functions, which were presented earlier in Eq. (5.1). The cost function may be phase-specific, or a phase-weighted sum. The cost formulations for the time- and torque-optimal routines are subsequently provided in Eq. (5.17) and Eq. (5.18) respectively.

4. `connect` script: This script defines the states at the temporal interfaces between phases. An example regarding the use of the `connect` script is a multi-stage rocket launch optimization, where the mass of the rocket would vary between control phases due to staging events. For the optimal guidance problem under consideration however, a single phase is sufficient, and the `connect` script is not utilized.

5.3 GPOPS Validation

Having developed an understanding of the algorithms, structure, and implementation of GPOPS, it was necessary to demonstrate the capability of the MATLAB-Simulink environment to generate valid solutions to the optimal guidance problem. An unperturbed time-optimal reorientation maneuver, originally simulated by Bilimoria and Wie [18], was selected as a benchmark. Subsequent studies by Fleming [87], and

Boyarko et al. [43] have reproduced these results, making them ideal for establishing the validity of an optimization environment.

The simulation variables and conditions for the benchmark were matched to those presented by Boyarko et al. for their GPOPS generated solution [43]. A 180° rest-to-rest yaw maneuver was performed for a symmetric spacecraft with inertia properties of $I_{xx} = I_{yy} = I_{zz} = 1 \text{ kg} \cdot \text{m}^2$, with a control authority torque limit of ± 1 N·m about each axis. The results of the time-optimal maneuver for this scenario according to Boyarko et al. are presented in Fig. 5.1 and Fig. 5.2.

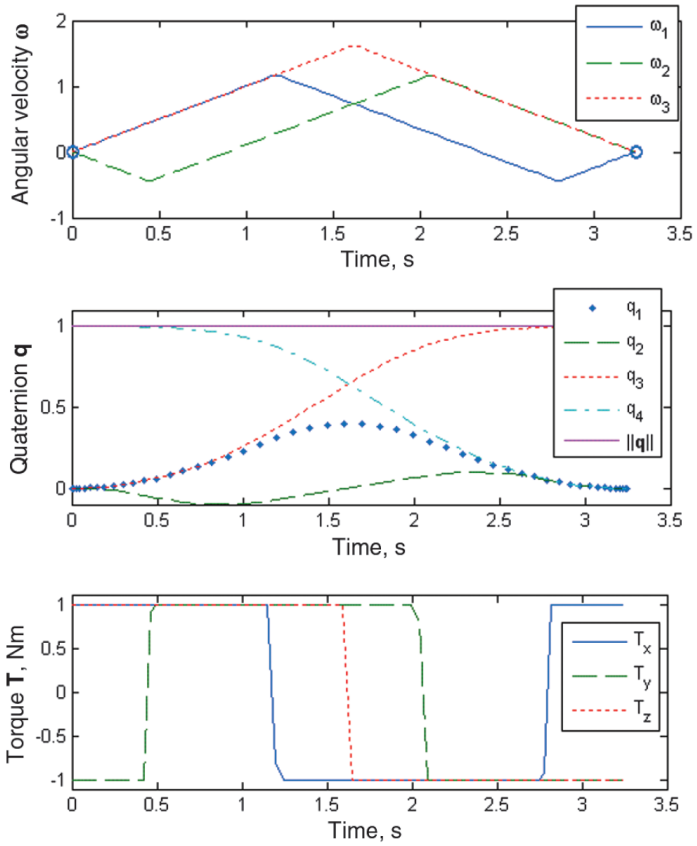


Figure 5.1: Time history: 180° slew maneuver about the z-axis, 100 node time-optimal GPOPS solution (Boyarko et al., 2011) [43]

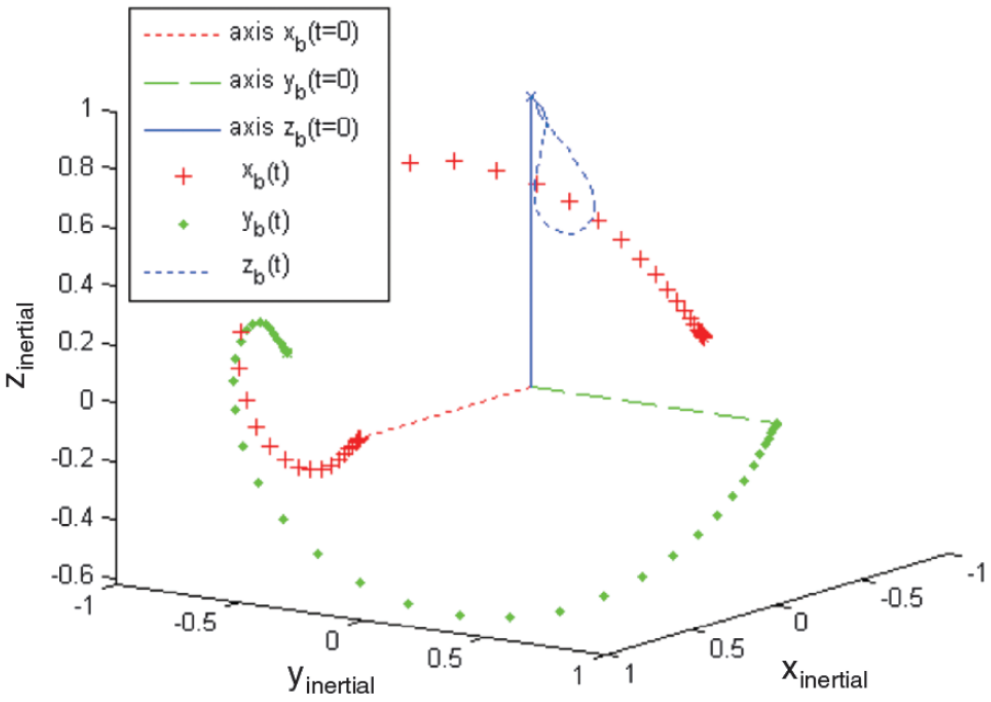


Figure 5.2: 3D Representation: 180° slew maneuver about the z-axis, 100 node time-optimal GPOPS solution (Boyarko et al., 2011) [43]

Figure 5.2 clearly illustrates that the time-optimal solution is not an Eigenaxis maneuver, with z_b being inclined during a rotation in $x_b y_b$ plane. The results indicate that the calculated optimal maneuver time, t_f, is 3.2430 seconds for this reorientation. It should be noted that the presented maneuver is technically a $-180°$ rotation, based on the direction about the z-axis. The authors state that 100 nodes, and two hours of CPU time, were required to obtain the stable converging GPOPS solution presented. However, given the sensitivity of any NLP system to the node spacing and initial conditions, it is difficult to ascertain if this suggested node count was an absolute minimum. The study suggests that reducing the node count could lead to a computationally more robust result [88]. However, in the published study, a 50-node setup produced a non-optimal solution. Further reduction to a 25-node setup resulted in an optimal solution symmetric to the 100 node case.

Given the focus throughout this thesis on minimizing computational requirements, reproducing the published results was approached in a different manner. The initial guesses in the `main` script were selected such that only 50 nodes were required to reproduce the published results. The reduction in node count, as well as careful choice of initial guesses resulted in a substantial reduction in run time. The time-optimal cost function in Lagrangian form is given by

$$J = \int_{t_0}^{t_f} \mathrm{d}t. \tag{5.17}$$

The reproduced time-optimal results, in a form similar to that employed by Boyarko et al. are presented Fig. 5.3 and Fig. 5.4.

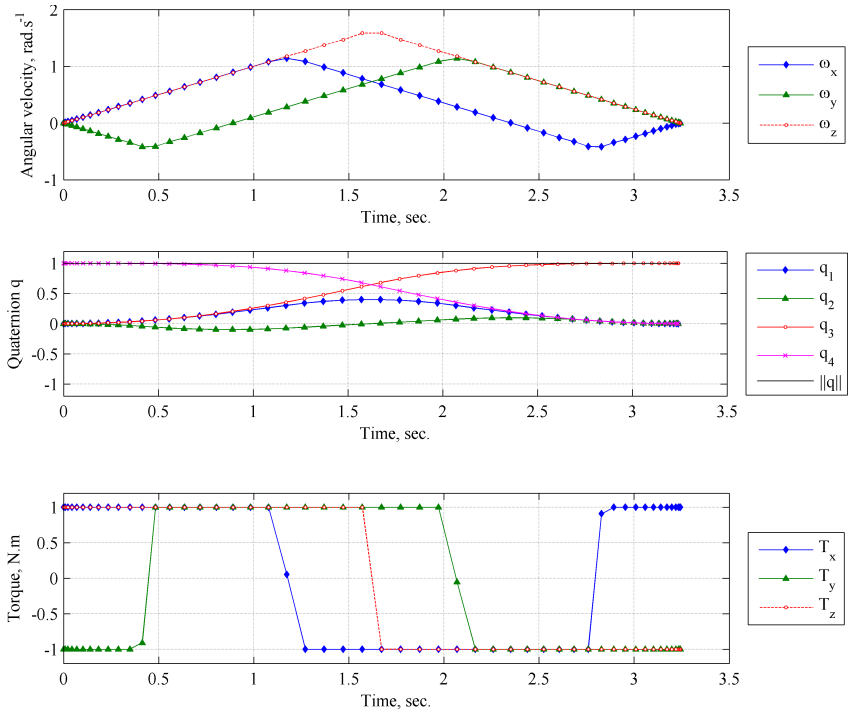


Figure 5.3: Time history: 180° slew maneuver about the z-axis, time-optimal GPOPS solution, reproduced using 50 nodes

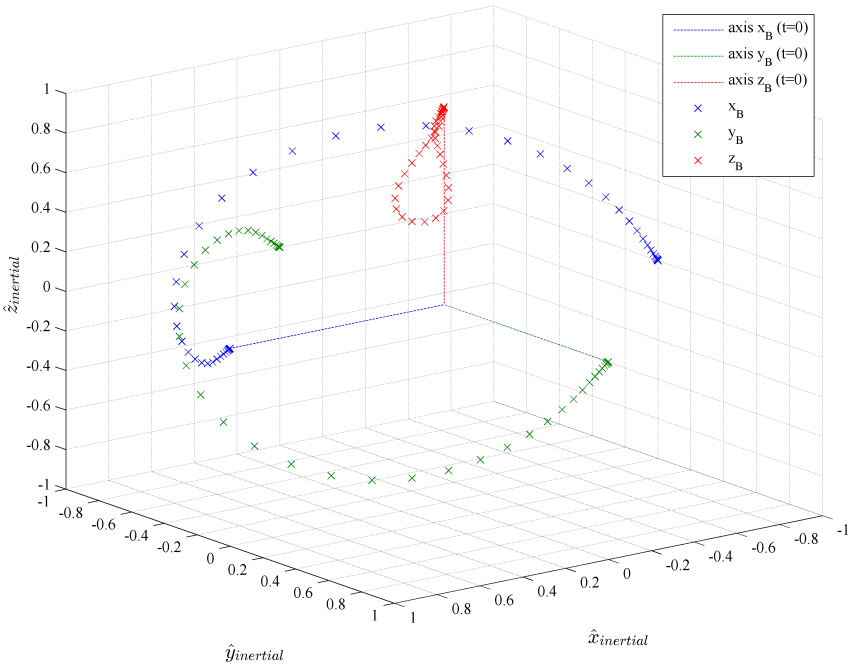


Figure 5.4: 3D Representation: 180° slew maneuver about the z-axis, time-optimal GPOPS solution, reproduced using 50 nodes

These results correspond closely to the benchmarks presented in Fig. 5.1 and Fig. 5.2. Furthermore, the calculated optimal maneuver time, t_f, is 3.2430 seconds, which corresponds exactly with the results presented by Boyarko et al. [43]. Table 5.1 compares the computation time for the 50 node Fast-GPOPS implementation against similar simulation results from Boyarko et al. [43] All improvements are calculated based on an Eigenaxis baseline.

Table 5.1: 180° rest-to-rest time-optimal slew maneuver about the z-axis

Trajectory generation method	Nodes	CPU Time, s	Cost, t_f	% Improvement
Eigenaxis	—	—	3.5449	—
Optimal (Bilimoria & Wie)	—	—	3.2431	8.51
GPOPS (Boyarko et al.)	25	15.5	3.2573	8.11
GPOPS (Boyarko et al.)	50	200.5	3.2851	7.31
GPOPS (Boyarko et al.)	100	5962.8	3.2430	8.52
Fast-GPOPS, Core2Duo (Kedare)	50	35.2	3.2430	8.52
Fast-GPOPS, i7 (Kedare)	50	10.5	3.2430	8.52
IDVD 7^{th}-order (Boyarko et al.)	50	69.8	3.3769	4.74
IDVD 7^{th}-order (Boyarko et al.)	100	121.2	3.3776	4.72

The Fast-GPOPS case was executed on a Hewlett-Packard 8710W notebook workstation equipped with an Intel®Core2Duo™T7700 processor clocked at 2.4 GHz with 4 GB available memory. This configuration is comparable to that utilized by Boyarko et al. [43]. Furthermore, the use of an Intel®i7™2700K processor clocked at 3.9 GHz with 16 GB available memory required CPU time on the same order of magnitude. Hence, the computational gain cannot be attributed primarily to computational advances. Instead, it appears that careful selection of the initial guesses in GPOPS contributed significantly towards reducing the required computational time.

Following the demonstration of an unperturbed time-optimal maneuver, matching published results using 50 nodes, an unperturbed torque-optimal implementation was analyzed for the spacecraft from Boyarko et al. The maneuver duration, which must be specified for a torque-optimal maneuver, was set to 20 seconds – corresponding to approximately 5× the duration for an Eigenaxis maneuver. The simulation aimed to

demonstrate that the MATLAB-GPOPS environment could generate a valid torque-optimal trajectory. The torque cost function was defined in Lagrangian form as

$$J = \int_{t_0}^{t_f} \sqrt{u_x^2 + u_y^2 + u_z^2}\, \mathrm{d}t. \tag{5.18}$$

which is representative of the sum of the control torque vector magnitude over time. This formulation of the torque cost shall be used for all subsequent torque-optimal studies within this thesis. The generated torque-optimal trajectory and associated attitude parameters are presented in Fig. 5.5 and Fig. 5.6 respectively.

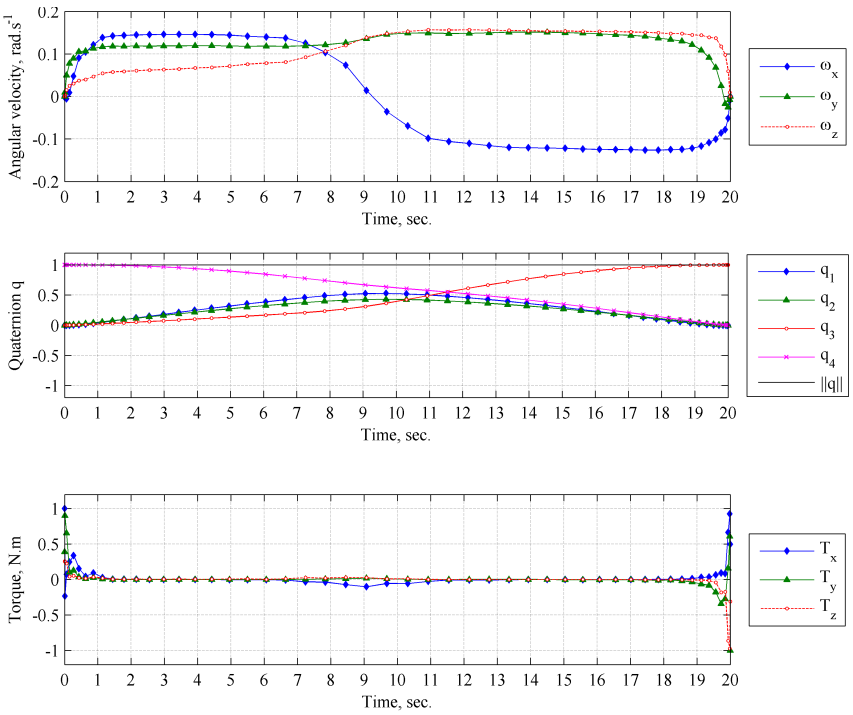


Figure 5.5: Time History: 180° slew maneuver about the z-axis, 50 node torque-optimal GPOPS solution

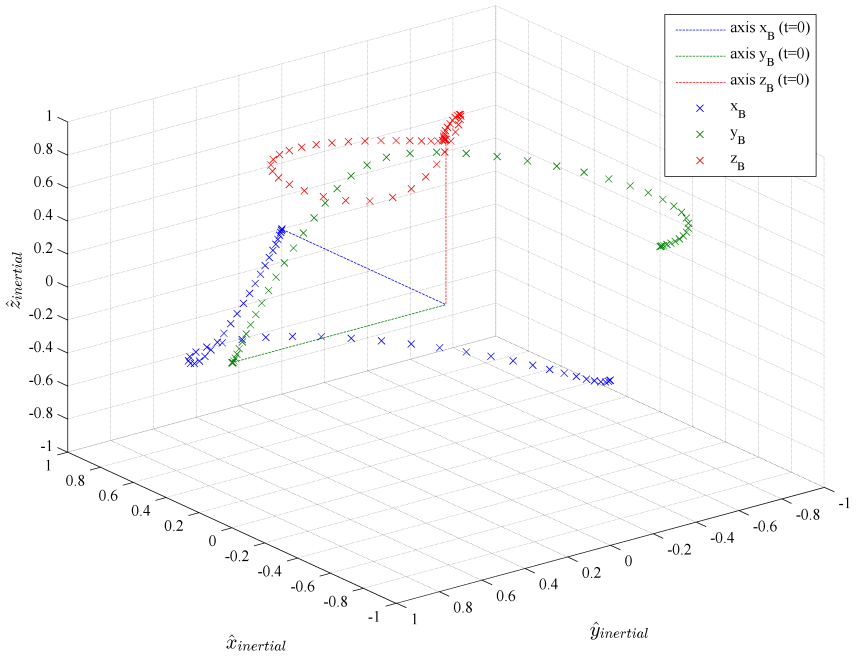


Figure 5.6: 3D Representation: 180° slew maneuver about the z-axis, 50 node torque-optimal GPOPS solution

The time-history of the torque-optimal guidance trajectory presented in Fig. 5.5 clearly illustrates the minimization of torque usage. The guidance routine applies a torque at the start and terminal phases of the maneuver, in addition to a mid-trajectory correction.

5.4 Implementation Details

This section presents the simplifications, and associated justifications, made to the dynamics model within the on-board propagator and optimization routine in an effort to reduce computational requirements. Also described are the properties of the satellite under consideration, and justification regarding the selected maneuver duration.

5.4.1 Guidance Algorithm Outline

Figure 5.7 illustrates the flow of information within the torque-optimal guidance algorithm.

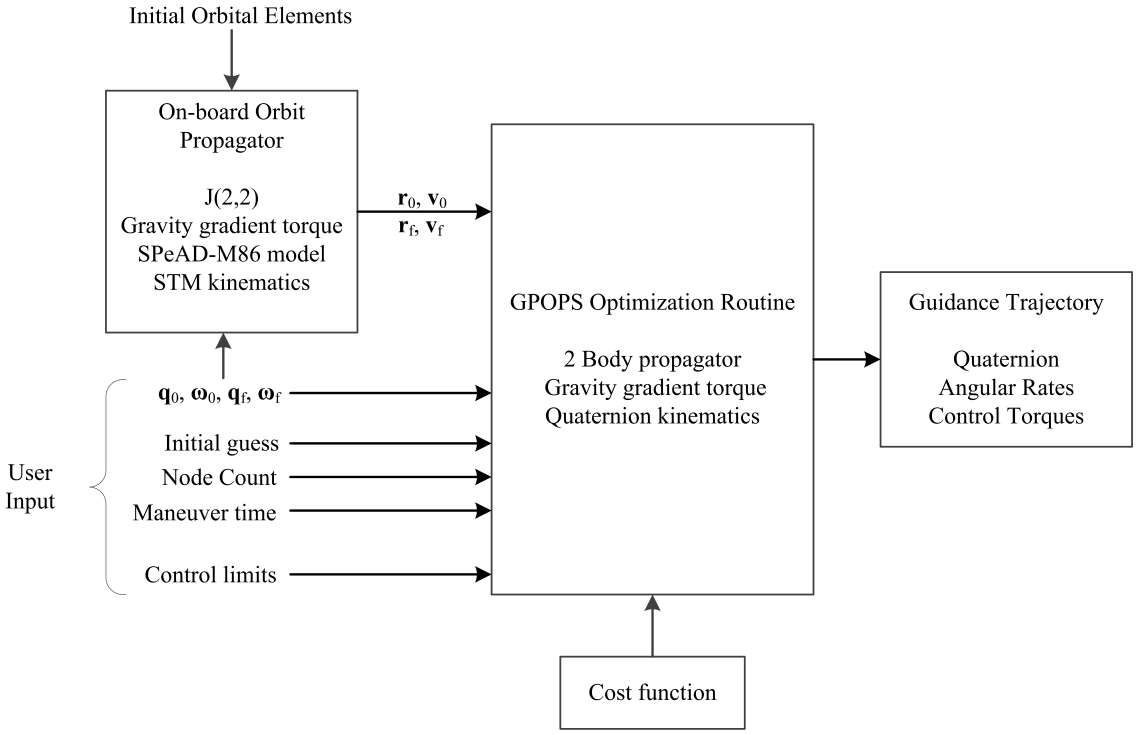


Figure 5.7: Guidance Algorithm Overview

The core of the guidance algorithm is the GPOPS optimization routine, which generates an optimal trajectory based on the user-defined cost function. The on-board propagator provides the initial and final orbital states required by the optimization routine. Additional user inputs include the desired initial and final attitude states, node count, maneuver time, control torque limits, and an initial guess for the states. The properties of the on-board and optimization propagation environments introduced in Figure 5.7 are summarized in Table 5.2.

Table 5.2: Orbit and Attitude Propagators within the Guidance Algorithm

Parameter	On-board	Optimization
Time step, Δt (sec)	1	Gauss points
Gravity model	Spherical harmonic	Keplerian
Order of gravity model	2	N.A.
Atmospheric Model	SPeAD-M86	None
Orbit Integrator	ode3	Gaussian quadrature
Attitude propagation	STM	Quaternion
Attitude Integrator	Forward Euler	Gaussian quadrature

Figure C.2 in Appendix C presents the Simulink block diagram of the on-board propagation environment. The development and justification of specific aspects of the on-board and optimization propagation environments shall now be discussed.

5.4.2 State Transition Matrix

In an effort to reduce the computational power required by the on-board model, a state transition matrix (STM), as presented by Whitmore [69], was implemented for attitude propagation. Details regarding the formulation of this technique can be found in U.S. Patent 6,061,611 "Closed-form integrator for the quaternion (Euler angle) kinematics equations". An abridged derivation of the STM shall now be presented, along with a discrete-time implementation technique. From Eq. (2.7), the rate of change of the quaternion state, $\dot{\mathbf{q}}$, is given by

$$\dot{\mathbf{q}} = \Psi\mathbf{q} \quad \text{where} \quad \Psi = \frac{1}{2}\boldsymbol{\Omega} \tag{5.19}$$

As Eq. (5.19) is of the form $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$, it can be integrated to obtain an expression for the continuous state.

$$\mathbf{x}(t) = e^{\int_{t_0}^{t} \mathbf{A}\, \mathrm{d}t}\mathbf{x}_0 \tag{5.20}$$

Next, let

$$\frac{U}{2} = \int_{t_0}^{t} \frac{\omega_x}{2}\, \mathrm{d}t, \quad \frac{V}{2} = \int_{t_0}^{t} \frac{\omega_y}{2}\, \mathrm{d}t, \quad \frac{W}{2} = \int_{t_0}^{t} \frac{\omega_z}{2}\, \mathrm{d}t \tag{5.21}$$

Substituting the variables from Eq. (5.21) into the skew-symmetric matrix $\mathbf{\Omega}$, and applying Eq. (5.20), we obtain a modified expression for the continuous quaternion state.

$$\mathbf{q}(t) = e^{\boldsymbol{\rho}}\mathbf{q}_0 \quad \text{where} \quad \boldsymbol{\rho} = \begin{bmatrix} 0 & \frac{W}{2} & -\frac{V}{2} & \frac{U}{2} \\ -\frac{W}{2} & 0 & \frac{U}{2} & \frac{V}{2} \\ \frac{V}{2} & -\frac{U}{2} & 0 & \frac{W}{2} \\ -\frac{U}{2} & -\frac{V}{2} & -\frac{W}{2} & 0 \end{bmatrix} \tag{5.22}$$

The state transition matrix, $\mathbf{\Phi}(t, t_0)$, is obtained as a function of the angular velocity states by expanding $e^{\boldsymbol{\rho}}$ using the Maclaurin series.

$$\mathbf{\Phi}(t, t_0) = \mathbf{I}_4 cos\left(\frac{\|\omega\|}{2}\right) + \boldsymbol{\rho}\frac{2}{\|\omega\|} sin\left(\frac{\|\omega\|}{2}\right) \quad \text{where} \quad \|\omega\| = \sqrt{U^2 + V^2 + W^2} \tag{5.23}$$

The STM is then substituted into Eq. (5.22) to obtain the quaternion state change equation.

$$\mathbf{q}(t) = \mathbf{\Phi}(t, t_0)\mathbf{q}_0 \tag{5.24}$$

With the STM and associated equation of state defined, we now construct an implementation technique by discretization of the variables presented in Eqs. (5.21) -

(5.24) at n temporal nodes, with $1 < k < n$.

$$U = \int_{t_0}^{t} \omega_x \, \mathrm{d}t = \frac{\omega_{x_k} + \omega_{x_{k+1}}}{2} \Delta t \tag{5.25a}$$

$$V = \int_{t_0}^{t} \omega_y \, \mathrm{d}t = \frac{\omega_{y_k} + \omega_{y_{k+1}}}{2} \Delta t \tag{5.25b}$$

$$W = \int_{t_0}^{t} \omega_z \, \mathrm{d}t = \frac{\omega_{z_k} + \omega_{z_{k+1}}}{2} \Delta t \tag{5.25c}$$

$$\boldsymbol{\rho}_{k+1,k} = \frac{1}{2} \begin{bmatrix} 0 & W & -V & U \\ -W & 0 & U & V \\ V & -U & 0 & W \\ -U & -V & -W & 0 \end{bmatrix} \tag{5.25d}$$

$$\|\omega_{k+1,k}\| = \sqrt{U^2 + V^2 + W^2} \tag{5.25e}$$

The state transition matrix can now be defined in the discrete-time domain as

$$\boldsymbol{\Phi}_{k+1,k} = \mathbf{I}_4 cos\left(\frac{\|\omega_{k+1,k}\|}{2}\right) + \boldsymbol{\rho}_{k+1,k} \frac{2}{\|\omega_{k+1,k}\|} sin\left(\frac{\|\omega_{k+1,k}\|}{2}\right) \tag{5.26}$$

Finally, the discrete time STM formulation for quaternion propagation becomes

$$\mathbf{q}_{k+1} = \boldsymbol{\Phi}_{k+1,k} \mathbf{q}_k \tag{5.27}$$

The quaternion states obtained through the STM propagation will tend to denormalize over time as a result of errors in evaluating trignometric terms. As a result, it is recommended that the quaternion be normalized periodically to avoid round-off error accumulation. Note that for the case of $\|\omega\| = 0$, i.e., no rotation, the STM is singular, and the algorithm assigns $\boldsymbol{\Phi}_{k+1,k} = \mathbf{I}_4$ to prevent numerical instability.

5.4.3 Environmental Torque Evaluation for a 3U CubeSat

From the requirement to minimize computational overhead, it was essential to determine the extent to which perturbation torques had to be modelled. Though prior studies appeared to have established gravity gradient and atmospheric effect as the dominant perturbing torques, these studies focused on generic spacecraft and not specifically on the CubeSat geometry. A 3U CubeSat with an inertia tensor identical to that presented in Sec. 4.5.2 was simulated in the real-world environment while the on-board controller maintained an orientation of $[45^\circ \ \ 45^\circ \ \ 45^\circ]_B^{RPY}$ for a period of one day. The ratio of the perturbing torque magnitudes, T_{gg} and T_{aero}, was calculated to allow for an order of magnitude analysis. Figure 5.8 presents the results of this study.

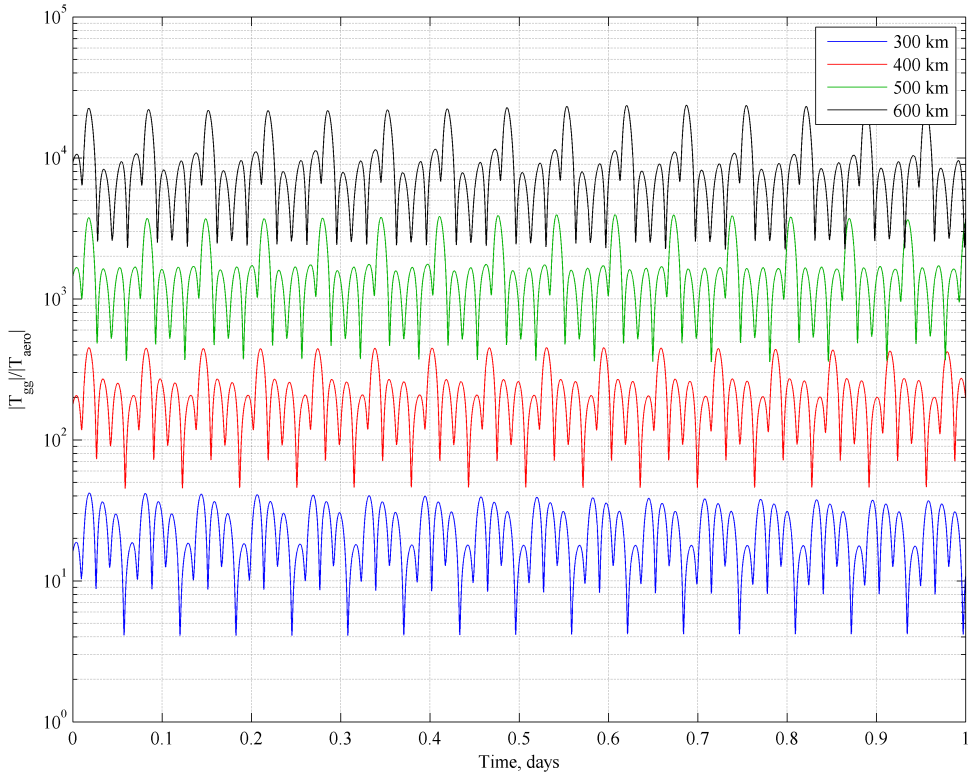


Figure 5.8: Magnitude comparison of dominant perturbing torques with varying orbit altitude for a 3U CubeSat in LEO

The order of magnitude analysis illustrates that for orbit altitudes of 400 km and higher, the gravity gradient torque $\vec{T}_{gg}$ is the dominant environmental perturbation affecting the spacecraft attitude. Therefore, in order to minimize computational cost, $\vec{T}_{gg}$ was the solitary environmental perturbation modeled within the optimization routine.

5.4.4 Quaternion Formulation of Gravity Gradient Torque

Initial attempts to implement gravity gradient perturbation torque, as presented in Eq. (2.8), into the optimization routine demonstrated severe numerical instability and non-convergent solutions. The trigonometric functions within the quaternion-to-DCM function were suspected to trigger these instabilities. An alternative formulation was therefore necessary. Junfeng et al. [89] utilize a technique to calculate the gravity gradient torque using the quaternion state and spacecraft position. This formulation is specifically for a spacecraft with principal moments of inertia, but could be easily extended to accommodate any inertia tensor.

$$T_{gg_x} = 3\frac{\mu_\oplus}{r^3}(I_{zz} - I_{yy})(2q_2q_1 - 2q_4q_3)(2q_3q_1 + 2q_4q_2) \tag{5.28a}$$

$$T_{gg_y} = 3\frac{\mu_\oplus}{r^3}(I_{xx} - I_{zz})(2q_3q_1 + 2q_4q_2)(2q_4^2 + 2q_1^2 - 1) \tag{5.28b}$$

$$T_{gg_z} = 3\frac{\mu_\oplus}{r^3}(I_{yy} - I_{xx})(2q_2q_1 - 2q_4q_3)(2q_4^2 + 2q_1^2 - 1) \tag{5.28c}$$

A similar formulation is incorporated by Bender for the calculation of gravity gradient torque [90]. However, Junfeng et al. and Bender both approximate the $\mu_\oplus/r^3$ term by the mean motion of the orbit, which is only a valid assumption for perfectly circular orbits. The formulation in Eq. (5.28) is the general case, where r is the magnitude of the position vector $\vec{r}_G$.

5.4.5 Orbit Propagator Evaluation for a 3U CubeSat

Due to the iterative nature of PS methods, it was essential to simplify the orbit propagator integrated within the optimization routine. Two simple propagation techniques were selected for evaluation - Keplerian 2-body, and Keplerian + J_2. As the on-board propagator accounts for atmospheric drag, and supplies the initial and final orbital states to the optimization routine, it was considered unnecessary to include atmospheric drag effects for the short duration of attitude maneuvers. The rate of change of the translational state vectors, $\vec{r}$ and $\vec{v}$, can be expressed by

$$\dot{\vec{r}}_G = \vec{v} \qquad \text{and} \qquad \ddot{\vec{r}}_G = \ddot{\vec{r}}_{Kep} + \ddot{\vec{r}}_{J2} \tag{5.29}$$

The two-body problem represents the perturbation-free motion of a satellite about a central body under the influence of gravity [5]. The locations of two masses, m_1 (Earth) and m_2 (spacecraft), are defined in an inertial reference frame as $\vec{r}_1$ and $\vec{r}_2$ respectively. The position of m_2 relative to m_1 is then given by

$$\vec{r}_{21} = \vec{r}_2 - \vec{r}_1 \tag{5.30}$$

From Newton's Law of Gravitation, the gravitational force exerted on m_2 by m_1 can be expressed as

$$\vec{F}_{21} = \frac{-Gm_1m_2}{{r_{21}}^2}\frac{\vec{r}_{21}}{r_{21}} \tag{5.31}$$

Similarly, the gravitational force exerted on m_1 by m_2 is given by

$$\vec{F}_{12} = \frac{Gm_1m_2}{{r_{21}}^2}\frac{\vec{r}_{21}}{r_{21}} \tag{5.32}$$

From Eq. (5.31) and Eq. (5.32), the acceleration of each mass in the inertial reference frame can be expressed as

$$\ddot{\vec{r}}_1 = \frac{Gm_2}{{r_{21}}^3}\vec{r}_{21} \tag{5.33}$$

$$\ddot{\vec{r}}_2 = -\frac{Gm_1}{{r_{21}}^3}\vec{r}_{21} \tag{5.34}$$

The relative acceleration of m_2 w.r.t. m_1 can then be written as

$$\ddot{\vec{r}}_{21} = \ddot{\vec{r}}_2 - \ddot{\vec{r}}_1 \tag{5.35}$$

Substituting the expression from Eq. (5.33) and Eq. (5.34) into Eq. (5.35), we obtain the general equation of motion of m_2 relative to m_1 in an inertial reference frame as governed by Newton's Law of Gravitation.

$$\ddot{\vec{r}}_{21} = \frac{-G(m_1 + m_2)}{{r_{21}}^3}\vec{r}_{21} \tag{5.36}$$

As the Earth is orders of magnitude more massive than the satellite, by assuming that $m_1 >> m_2$, fixing the inertial reference frame to the center of gravity of the central body i.e., Earth, and representing Gm_1 by the Earth's gravitational parameter, $\mu_\oplus$, we obtain the simplified equation of motion describing the unperturbed motion of a satellite about the Earth.

$$\ddot{\vec{r}}_{Kep} = \frac{-\mu_\oplus}{{r_G}^3}\vec{r}_G \tag{5.37}$$

The J_2 term of the gravitational potential function U, as given by Bate [91], is presented in Eq. (5.38).

$$U_{J2} = \frac{-\mu_\oplus}{r^3} J_2 R_\oplus^2 \left(\frac{3}{2} sin^2\delta - \frac{1}{2}\right) \tag{5.38}$$

The perturbing acceleration in the ECI reference frame due to the J_2 term can then be obtained by finding the scalar potential of U_{J2}.

$$\ddot{\vec{r}}_{J2} = \nabla U_{J2} = \frac{\partial}{\partial x} U_{J2} \hat{\imath}_G + \frac{\partial}{\partial y} U_{J2} \hat{\jmath}_G + \frac{\partial}{\partial z} U_{J2} \hat{k}_G \tag{5.39}$$

Through use of a spherical coordinate system to express $sin^2\delta$ from Eq. (5.38) in Cartesian coordinates, and subsequent application of the ∇ operator, we obtain the perturbing acceleration due to the J_2 zonal harmonic.

$$\ddot{\vec{r}}_{J2} = \frac{3\mu J_2 {R_\oplus}^2}{2r_G^5} \left(5\frac{z_G^2}{r_G^2} - 1 \right) \vec{r}_G - 2z_G \hat{k}_G \tag{5.40}$$

Having defined $\ddot{\vec{r}}_{Kep}$ and $\ddot{\vec{r}}_{J2}$, we may now evaluate the error introduced in $\vec{r}$ and $\vec{T}_{gg}$ by neglecting the J_2 term. These errors are calculated for a 3U CubeSat in a near-circular orbit at an altitude 400 km is actively held in a $[45^\circ \ \ 45^\circ \ \ 45^\circ]_B^{RPY}$ orientation. The results of this evaluation are presented in Figure 5.9.

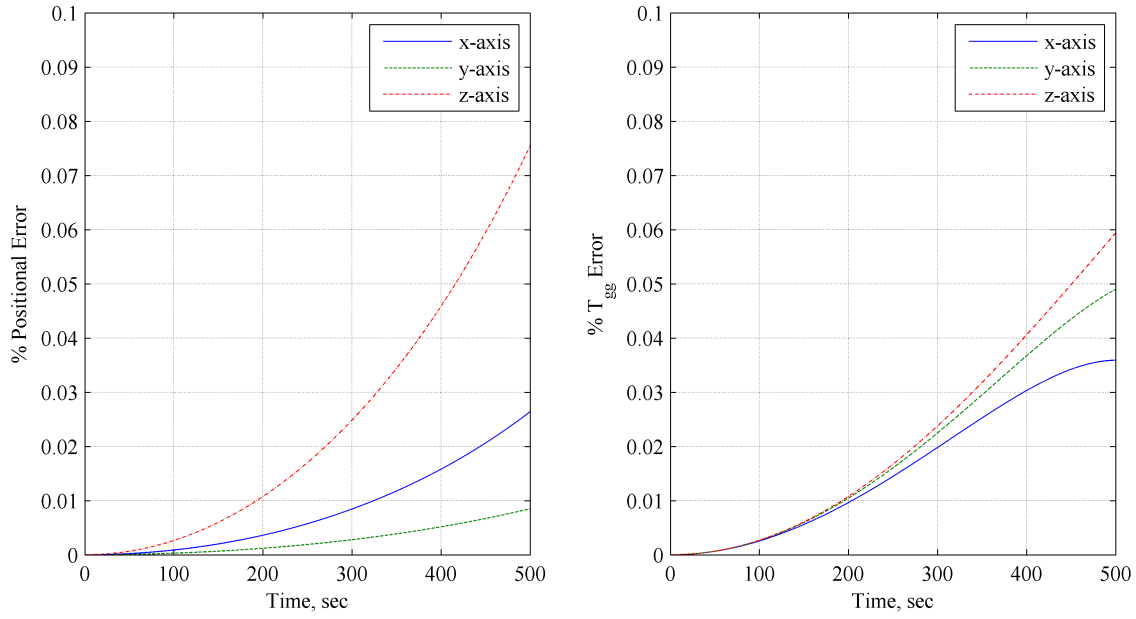


Figure 5.9: Positional and T_{gg} errors for a Keplerian propagator ($GM + J_2$ baseline)

For the short simulation periods of up to 500 seconds, the use of a Keplerian propagator does not introduce significant error in the position of the spacecraft. As a result, the propagated error in the gravity gradient torque is also minimal, with a maximum of 0.06% about the z-axis. For the maneuver time of 100 seconds being considered for optimal guidance, the maximum error for both $\vec{r}_G$ and $\vec{T}_{gg}$ is 0.003%.

5.4.6 Simulation Properties

A 3U CubeSat having a mass of 4.0 kg and an inertia tensor $\mathbf{J}$ identical to that presented in Sec. 4.5.2 with initial Keplerian orbital elements

$$[a \quad e \quad i \quad \Omega \quad \omega \quad \nu]_0 = [6778 \text{ km} \quad 0.005 \quad 0.1^\circ \quad 270^\circ \quad 90^\circ \quad 0^\circ]$$

was considered for this study. This corresponds to a 366.1 km $\times$ 433.9 km orbit having a period of 92.6 minutes, with a low inclination of 0.1° to the equatorial plane of the Earth. The spacecraft configuration resembles that presented in Fig. 4.1 minus the drag sails. Table 5.3 presents the initial and final states for the guidance algorithm.

Table 5.3: Summary of Initial and Final States for Optimization Routine

Parameter	Value	Unit
$\mathbf{r_0}$	$[6744.1 \quad -1.652\text{E-}12 \quad 11.77]_G$	km
$\mathbf{r_f}$	$[6700.3 \quad 769.03 \quad 11.69]_G$	km
$\mathbf{v_0}$	$[1.88\text{E-}15 \quad 7.707 \quad 8.237\text{E-}19]_G$	km·s^{-1}
$\mathbf{v_f}$	$[-8.757\text{E-}1 \quad 7.657 \quad 1.533\text{E-}3]_G$	km·s^{-1}
$\boldsymbol{\omega_0}$	$[0 \quad 0 \quad 0]_B$	rad·s^{-1}
$\boldsymbol{\omega_f}$	$[0 \quad 0 \quad 0]_B$	rad·s^{-1}
$\mathbf{q}_0$	$[0 \quad 0 \quad 0 \quad 1]$	–
$\mathbf{q}_f$	Case 1: $[0 \quad 0 \quad 1 \quad 0]$	–
	Case 2: $[0 \quad 0 \quad \sqrt{2}/2 \quad \sqrt{2}/2]$	–
Nodes	Case 1: 60	–
	Case 2: 20	–
t_0	0	sec
t_f	100	sec

Case 1 and Case 2 in $\mathbf{q}_f$ correspond to a 90° and 180° rotation about the body z-axis respectively, while $\mathbf{q}_0$ represents an initial orientation aligned with the geocentric inertial frame. The node count for each case was selected iteratively based on numerical stability and the generation of a valid solution. The control torques were limited to 0.1 mN·m and the angular momentum limit set to 10.3 mN·m·, which correspond to 16% and 100% of the relevant limits of the MAI-201 Miniature 3-Axis Reaction Wheel system[1]. The reduced torque capability was enforced to simulate a worst-case scenario with degraded reaction wheels. The selection of t_0 and t_f was based on a preliminary scaling of time-optimal and Eigenaxis results using the maneuver duration

[1]MAI-201 Miniature 3-Axis Reaction Wheel, CubeSatShop.com, 2014 (accessed Aug. 24, 2014)

correlation in Eq. (5.41) as outlined by Bilimoria and Wie [18].

$$t = \sqrt{\frac{\mathbf{J}_{ii}}{u_{max}}}\hat{t} \tag{5.41}$$

where $\hat{t}$ is the non-dimensional maneuver time and u_{max} is the maximum allowable control torque. A range of maneuver durations, presented in Table 5.4 was computed based on the minimum and maximum satellite inertia components.

Table 5.4: Estimated Maneuver Duration

Yaw angle	Maneuver type	t_{min} (sec.)	t_{max} (sec.)
90°	Eigenaxis	60.88	88.90
	Time-optimal	52.80	85.94
180°	Eigenaxis	86.11	125.8
	Time-optimal	78.77	115.1

Based on these estimates, a maneuver duration of 100 seconds was selected for the 90° and 180° yaw maneuvers. The possibility that the satellite may be incapable of performing a 180° rotation about the yaw axis within the enforced time limit is acknowledged. However, such a scenario tests the robustness of the optimal guidance routine to generate a valid trajectory when faced with maneuvers at or beyond its dynamic capabilities. The MATLAB code of the torque-optimal guidance algorithm can be found in Listings C.1–C.3 in Appendix C.

5.5 Torque-optimal Trajectory Validation

Following the generation of torque-optimal trajectories, it was necessary to validate them in a real-world propagation environment. To accomplish this, the time history

of torques, as generated by GPOPS, was passed to an open-loop controller running within the real-world propagator. Specifically, the open-loop control law is given by

$$\vec{u} = \vec{T}_{GPOPS} \tag{5.42}$$

This open loop controller is not provided any information regarding the target attitude state - it simply applies the commanded torque profile. Though a PID controller could be designed to follow the generated $\mathbf{q}$ and $\vec{\omega}$ profiles, such an implementation is beyond the scope of this thesis.

As the optimal trajectory is discretized at the Gauss nodes, the time intervals between the guidance 'waypoints' are not constant. However, the real-world simulator and typical OBC modules operate at a constant time-step. It was therefore necessary to sample the guidance trajectory at a constant frequency of 10 Hz using the MATLAB `resample` function with linear interpolation.

Chapter 6

Results and Discussion

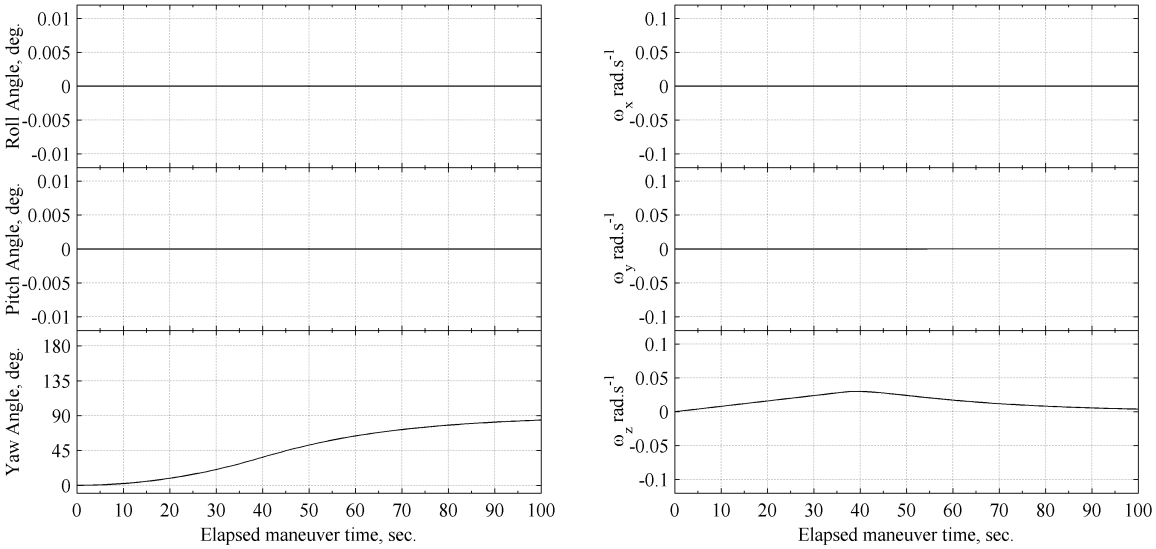
The trajectories generated by the torque-optimal guidance algorithm and associated validation results shall now be presented, along with the performance for Eigenaxis maneuvers in identical scenarios.

6.1 Large-Angle Eigenaxis Maneuvers

Having defined the spacecraft maneuver parameters, the first step was to examine large-angle rest-to-rest Eigenaxis maneuvers for a satellite with the properties presented in Sec. 5.4.6. This study evaluates the performance of an Eigenaxis controller for a 3U CubeSat equipped with realistic torque and angular momentum constraints. The real-world model was used alongside a PD controller similar to that described in Sec. 4.3, with $\zeta = 1$ and $\omega_n = 0.13$.

6.1.1 90° Rest-to-rest Yaw Maneuver

Figure 6.1 presents the spacecraft orientation during a 90° rest-to-rest yaw maneuver. In 100 seconds, the Eigenaxis guidance and control law is able to yaw the spacecraft to an angle of 84.21°, undershooting the commanded attitude by 5.79°.



(a) RPY angles

(b) Angular rates

Figure 6.1: Attitude states: 90° Eigenaxis yaw maneuver, $\Delta t_{mv} = 100$ sec.

During this maneuver, the roll and pitch angles remain at 0.00°, indicating that the z-axis is indeed the Eigenaxis. At the end of the maneuver, the spacecraft has a small residual ω_z of 0.22 deg·s^{-1}, indicating that the Eigenaxis maneuver was unable to fully satisfy the rest-to-rest constraint within the allotted maneuver duration. Figure 6.2 and Figure 6.3 present the actuator usage and trajectory followed by the body axis during the maneuver respectively.

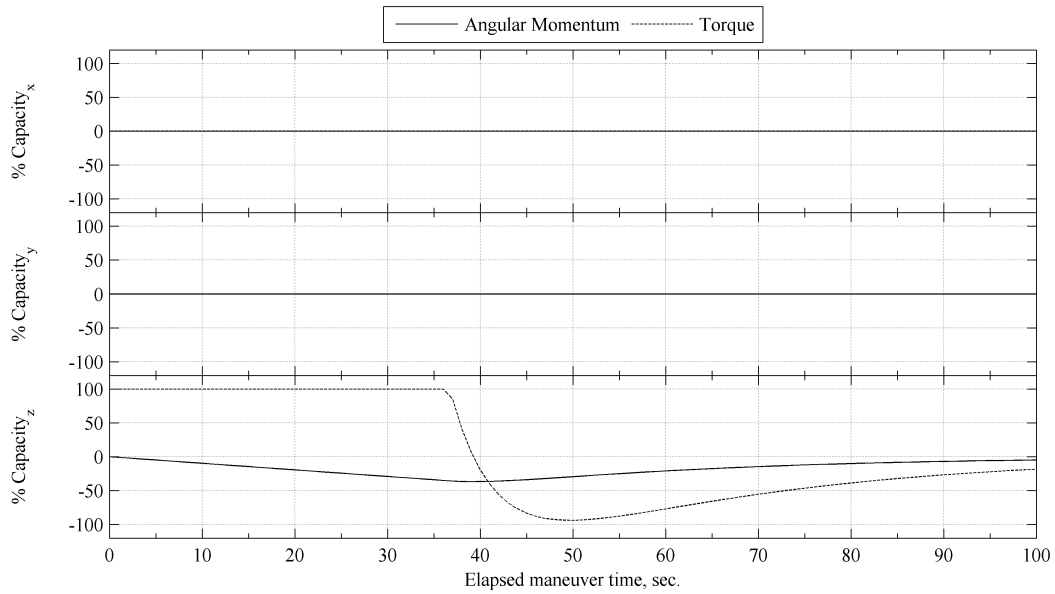


Figure 6.2: Actuator usage: 90° Eigenaxis yaw maneuver, $\Delta t_{mv} = 100$ sec.

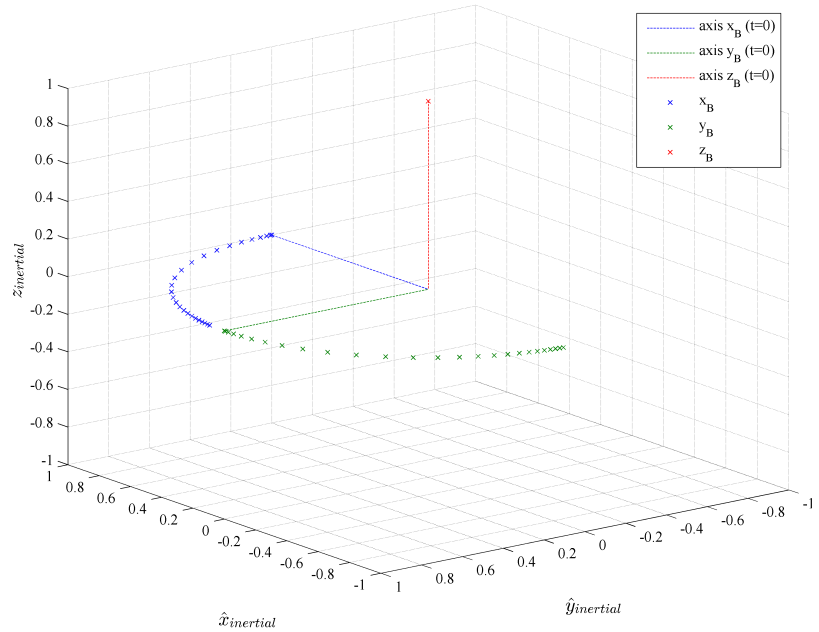


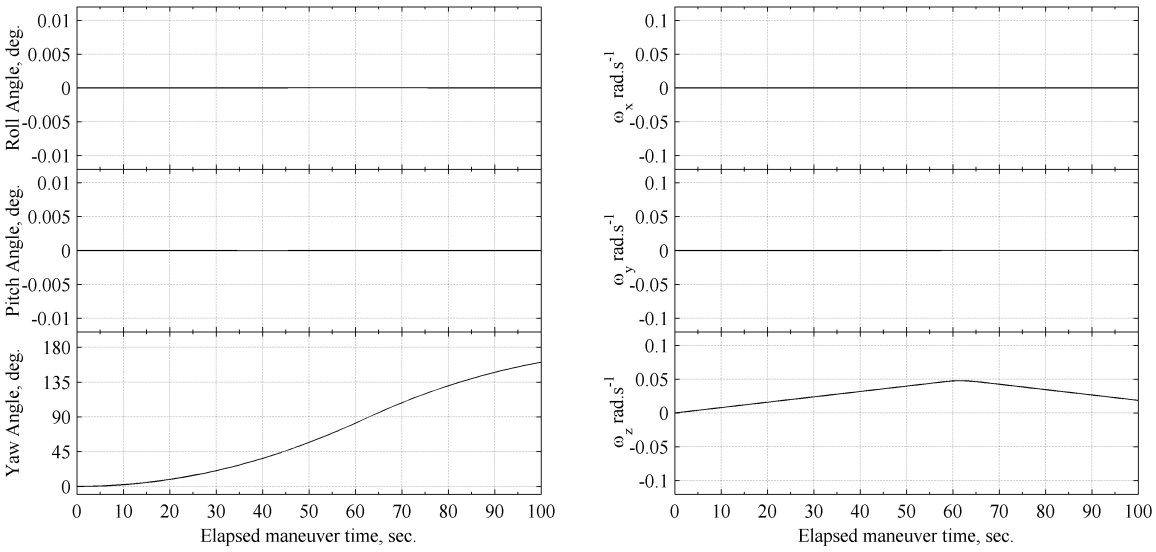
Figure 6.3: 3D Representation: 90° Eigenaxis yaw maneuver, $\Delta t_{mv} = 100$ sec.

From Fig. 6.2, we observe that rotating about the z-axis results in sole usage of the z-axis actuator. The maximum angular momentum usage during the maneuver

was 38.9%. At the end of the maneuver, momentum usage in the z-axis accumulates to 4.76%. The trajectory in Fig. 6.3 verifies the use of the z-axis as the Eigenaxis, and illustrates that the spacecraft did not achieve the commanded 90° yaw angle.

6.1.2 180° Rest-to-rest Yaw Maneuver

Figure 6.4 presents the spacecraft orientation during a 180° yaw maneuver.



(a) RPY angles

(b) Angular rates

Figure 6.4: Attitude states: 180° Eigenaxis yaw maneuver, $\Delta t_{mv} = 100$ sec.

In 100 seconds, the Eigenaxis guidance and control law is able to yaw the spacecraft to an angle of 160.6°, undershooting the commanded attitude by 19.4°. Similar to the 90° Eigenaxis maneuver, the roll and pitch angles remain at 0.00°. At the end of the maneuver, the spacecraft has a residual ω_z of 1.07 deg·s^{-1}, which indicates that the Eigenaxis maneuver did not satisfy the rest-to-rest constraint. Figure 6.5 and Figure 6.6 present the actuator usage and trajectory followed by the body axis during the maneuver respectively.

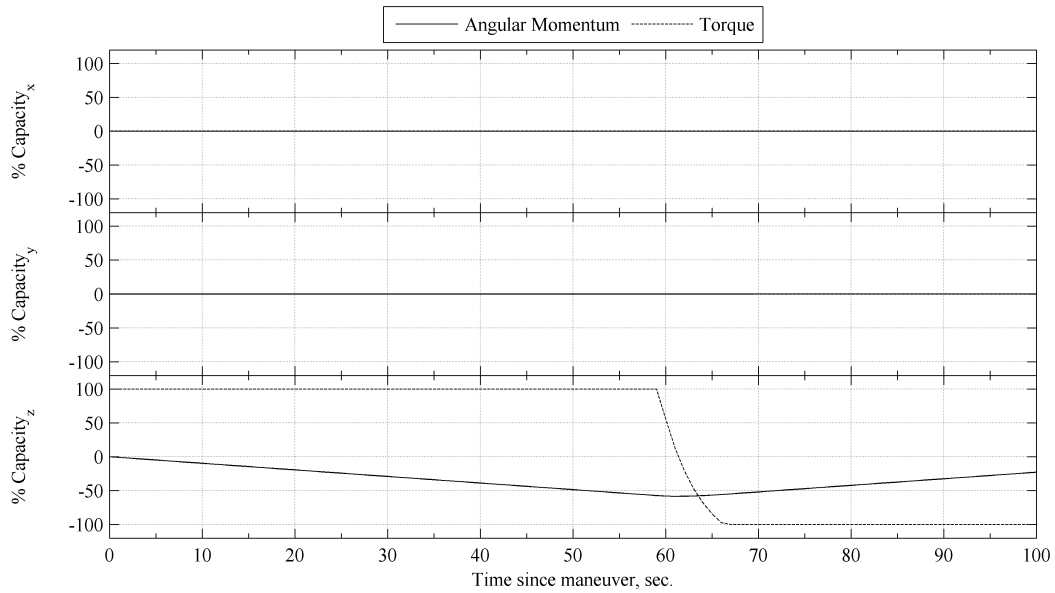


Figure 6.5: Actuator usage: 180° Eigenaxis yaw maneuver, $\Delta t_{mv} = 100$ sec.

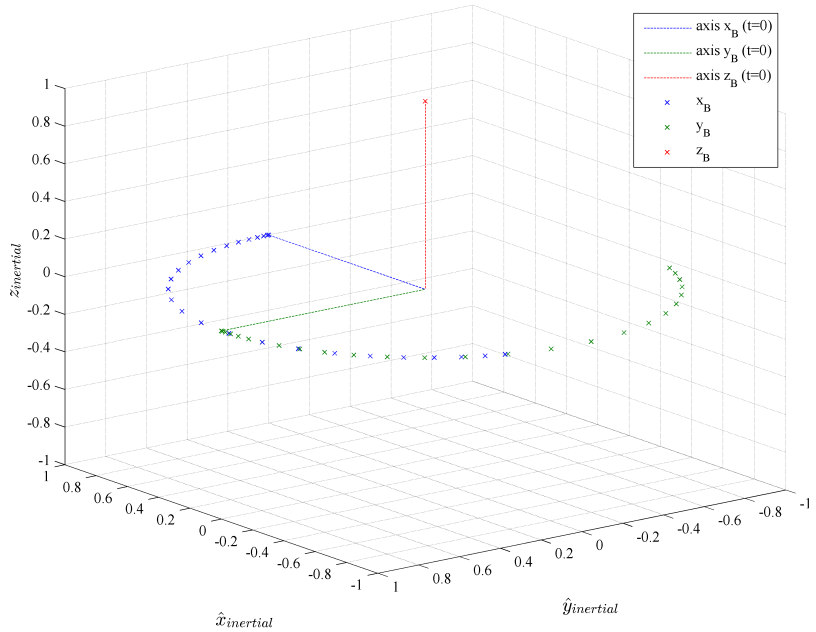


Figure 6.6: 3D Representation: 180° Eigenaxis yaw maneuver, $\Delta t_{mv} = 100$ sec.

Figure 6.6 confirms the use of the z-body axis as the Eigenaxis during the maneuver. The maximum angular momentum usage during the maneuver was 58.4%.

At the end of the maneuver, momentum usage about the z-axis was 22.8%. The trajectory indicates that the spacecraft did not achieve the commanded attitude in the specified maneuver duration, despite significant utilization of the available torque and angular momentum.

6.2 Torque-Optimal Solutions

After examining the performance of an Eigenaxis guidance law, we now present the torque-optimal guidance formulation for the 90° and 180° yaw maneuvers. It should be noted that the trajectories presented in this section are not representative of global minima for the associated torque cost-function, but rather of local minima for the specified maneuver duration.

6.2.1 90° Rest-to-rest Yaw Maneuver

Figure 6.7 presents the attitude states and control variables generated by the torque-optimal guidance routine for a 90° yaw maneuver.

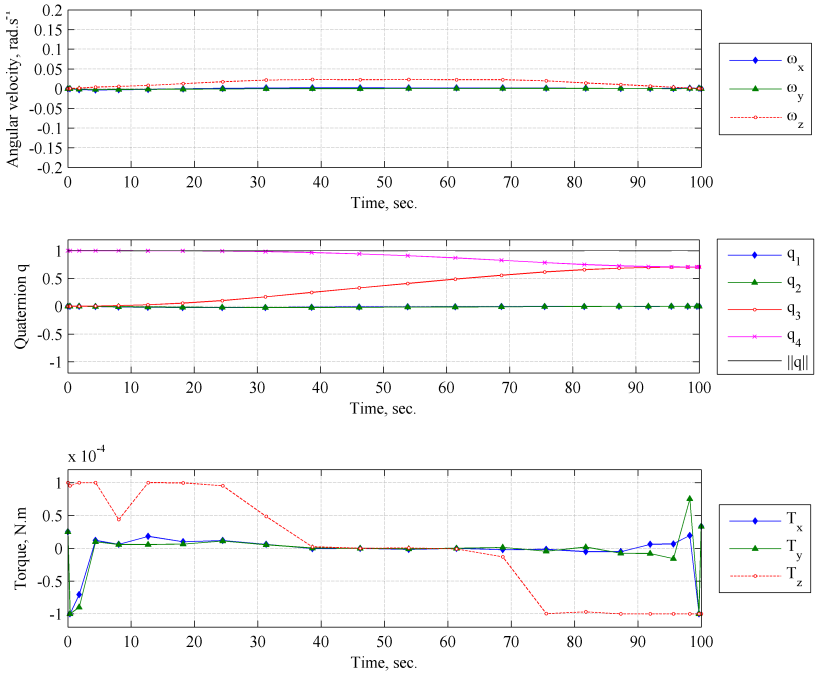


Figure 6.7: Time history: 90° yaw torque-optimal trajectory

The torque-optimal guidance routine generates a non-linear path from the initial to the final attitude states in quaternion space, using internal and external disturbance torques to minimize the torque-cost function. The generated torque profile resembles that presented in Fig. 5.5 during GPOPS validation. By utilizing internal and external torques produced during the rotation, and implementing a "look-ahead" approach, the algorithm reduces actuator usage. Figure 6.8 illustrates the associated trajectory in 3D space.

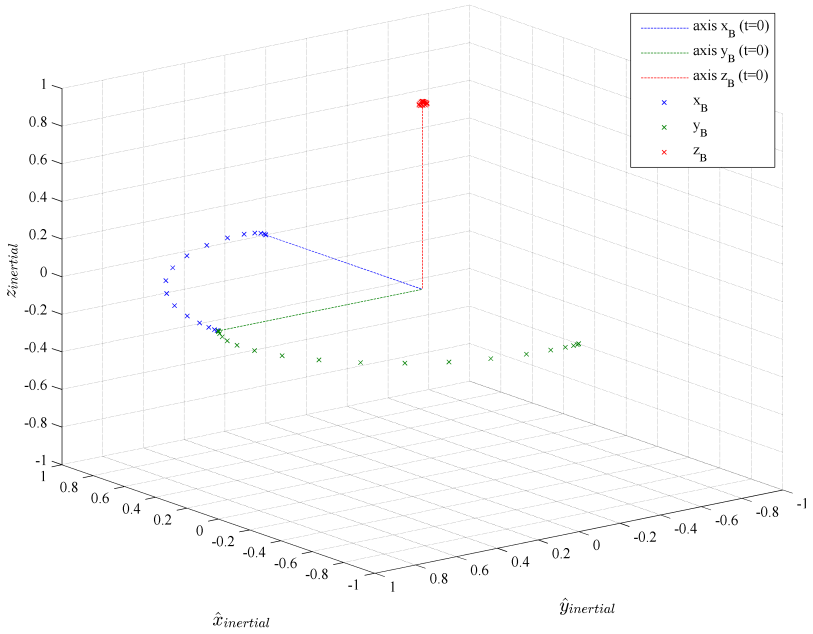


Figure 6.8: 3D Representation: 90° yaw torque-optimal trajectory

For a 90° yaw maneuver within the specified duration, the torque-optimal solution is near-Eigenaxis. The minor deviations add nutational components based on the inertia tensor of an asymmetric spacecraft, effectively increasing the torque available about each axis.

6.2.2 180° Rest-to-rest Yaw Maneuver

Figure 6.9 presents the attitude states and control variables generated by the torque-optimal guidance routine for a 180° yaw maneuver.

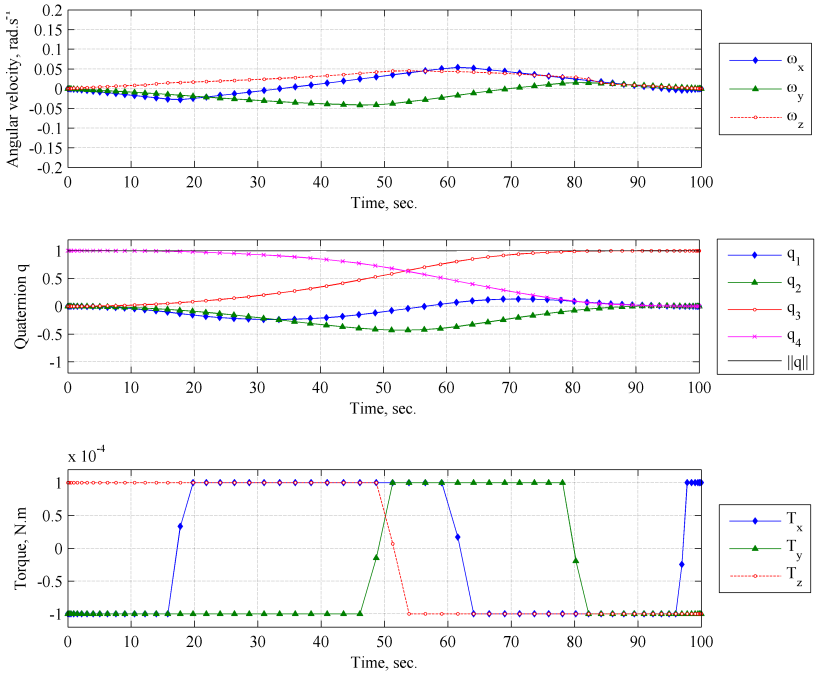


Figure 6.9: Time history: 180° yaw torque-optimal trajectory

As in the case of the 90° yaw maneuver, the torque-optimal guidance routine generates a non-linear path from the initial to the final attitude states in quaternion space. However, unlike the 90° case, the actuators continuously apply torque about each axis for the duration of the maneuver. This is due to the larger angular rates required to accomplish the maneuver within the specified time. Figure 6.8 illustrates the generated trajectory in 3D space.

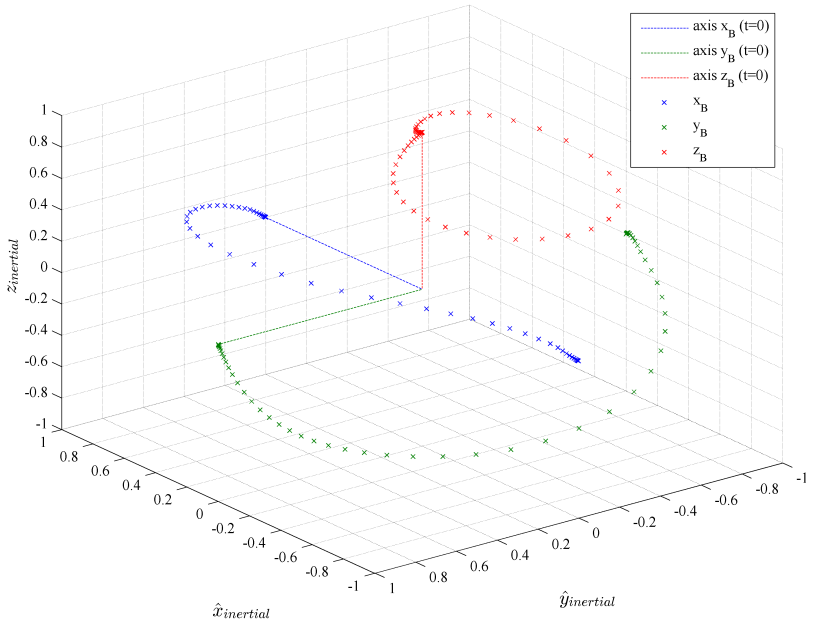


Figure 6.10: 3D Representation: 180° yaw torque-optimal trajectory

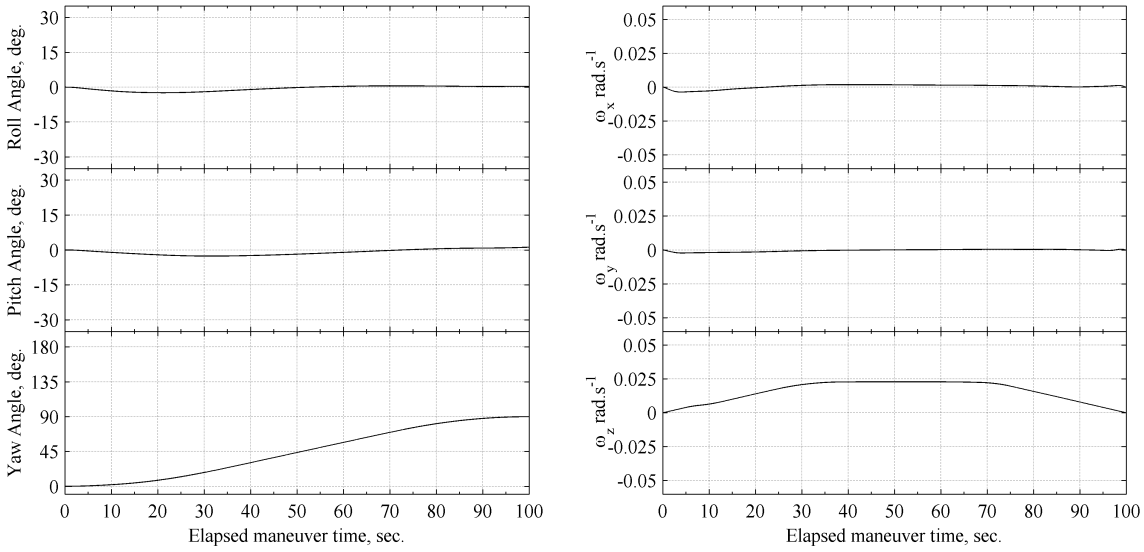
For a 180° yaw maneuver within the specified duration, the torque-optimal solution is clearly non-Eigenaxis. The z-axis deviates significantly from the vertical, tracing a path resembling a cardioid. As a result, both the x- and y-axis deviate from the perfect arcs seen for the Eigenaxis maneuver in Fig. 6.6. It can be deduced that the axis of rotation is not constant – a characteristic typical of both time- and torque-optimal guidance. The algorithm clusters the Gauss points at the beginning and end of the trajectory in an effort to accurately capture the large rates of change near these temporal nodes.

6.3 Torque-Optimal Guidance Validation

Having presented the generated torque-optimal trajectories for the 90° and 180° yaw maneuvers as generated by GPOPS, we examine the validation results in an open-loop control scenario.

6.3.1 90° Rest-to-rest yaw Maneuver

Figure 6.11 presents the spacecraft orientation while following the guidance torque profile for the 90° yaw maneuver.



(a) RPY angles

(b) Angular rates

Figure 6.11: Attitude states: 90° torque-optimal yaw maneuver validation

By following the trajectory described by the torque-optimal guidance algorithm, the spacecraft achieves a yaw angle of 89.88° within the specified maneuver duration. The associated roll and pitch angles are 0.33° and 1.16° respectively. The torque-optimal guidance routine therefore achieves greater yaw-pointing accuracy than an Eigenaxis controller, but introduces small errors in roll and pitch. At the end of the maneuver, the angular rates about all three axis are <0.01 deg·s^{-1}. The torque-optimal guidance law thus satisfies the rest-to-rest constraint for the maneuver, unlike an Eigenaxis controller. Figure 6.12 and Figure 6.13 present the actuator usage and maneuver trajectory respectively.

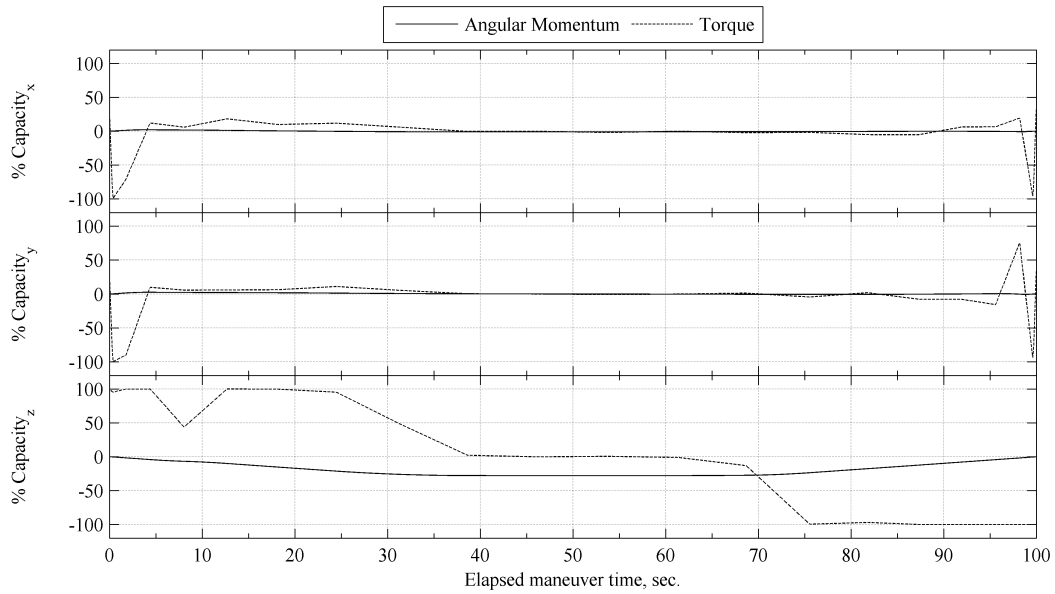


Figure 6.12: Actuator usage: 90° torque-optimal yaw maneuver validation

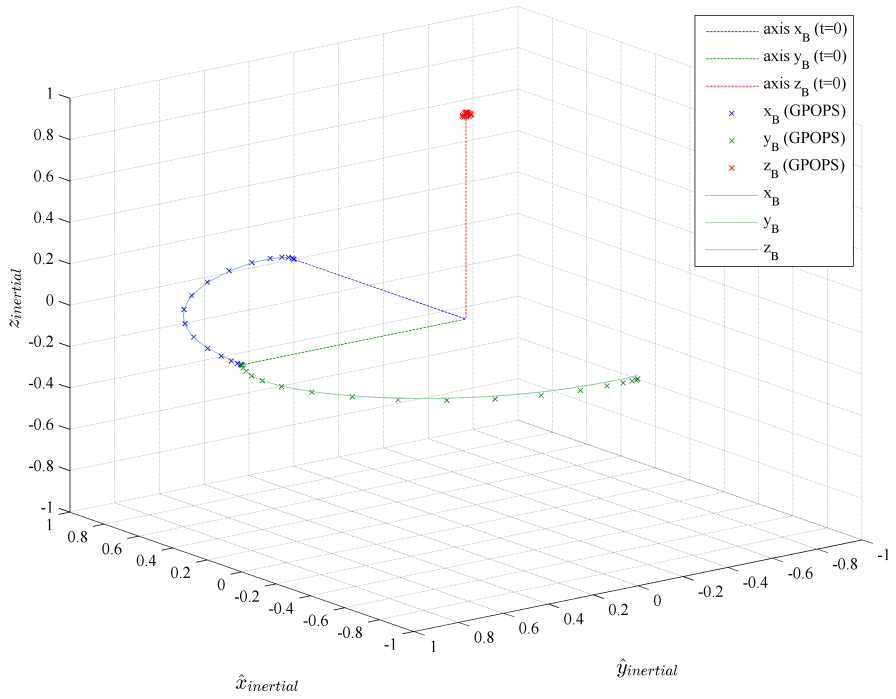


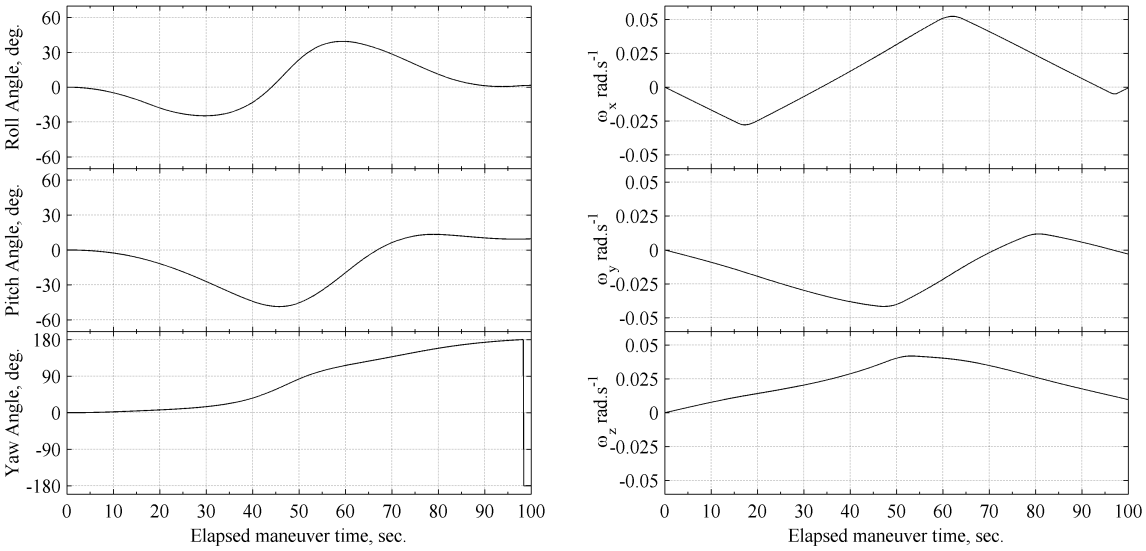
Figure 6.13: 3D Representation: 90° torque-optimal yaw maneuver validation

From Fig. 6.12, we observe the utilization of actuators about all three axes. The

maximum angular momentum usage during the maneuver is 2.08%, 2.49%, and 27.9% in the x-, y-, and z-axis respectively. At the end of the maneuver, accumulated momentum usage is <0.1% about each axis. This is a significant improvement over the actuator usage for the equivalent Eigenaxis maneuver, presented in Fig. 6.2. By utilizing the nutational torque components, the guidance algorithm significantly reduces torque usage about the z-axis, with minimal additional momentum usage in the x- and y-axis. The spacecraft closely follows the torque-optimal guidance waypoints, as seen in Fig. 6.13.

6.3.2 180° Rest-to-rest yaw Maneuver

Figure 6.14 presents the spacecraft orientation while following the guidance waypoints for the 180° yaw maneuver.



(a) RPY angles

(b) Angular rates

Figure 6.14: Attitude states: 180° torque-optimal yaw maneuver validation

Following the trajectory described by the torque-optimal guidance algorithm, the

spacecraft achieves a yaw angle of -179.1° within the specified maneuver duration. This corresponds to a yaw-pointing error of 0.9°, which is significantly lower than that achieved in the Eigenaxis case. The associated roll and pitch angles are 1.56° and 9.56° respectively. The torque-optimal guidance routine therefore achieves greater yaw-pointing accuracy than an Eigenaxis controller, but introduces moderate errors in roll and pitch. At the end of the maneuver, ω_x, ω_y, and ω_z are 0.04, 0.17 and 0.56 deg·s^{-1} respectively. Though the rest-to-rest constraint is not satisfied, ω_z is nearly half that achieved by the Eigenaxis controller, underlining the robustness of the guidance algorithm. Figure 6.15 and Figure 6.16 present the actuator usage and maneuver trajectory respectively.

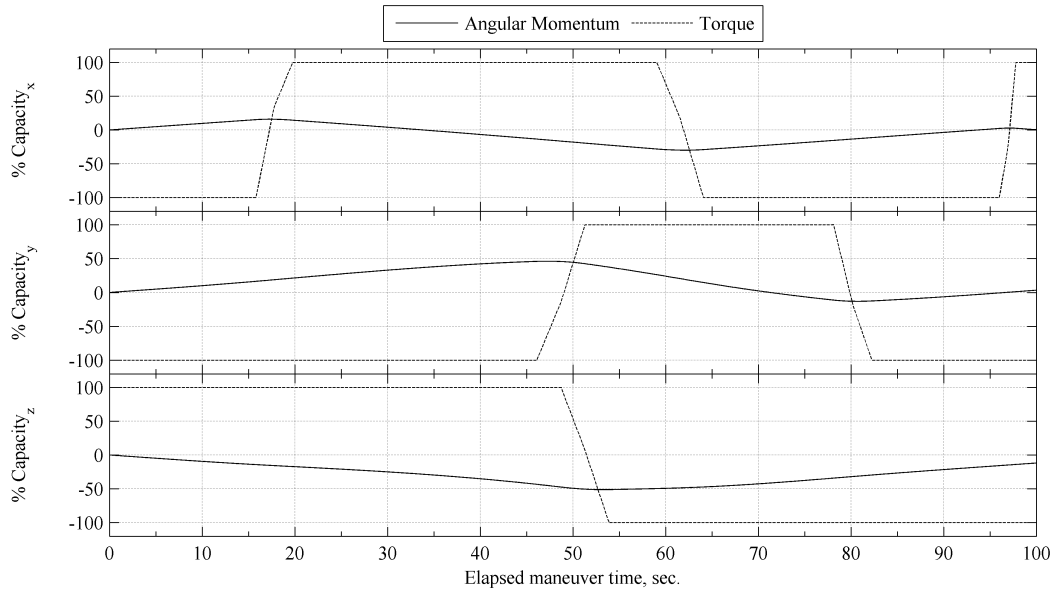


Figure 6.15: Actuator usage: 180° torque-optimal yaw maneuver validation

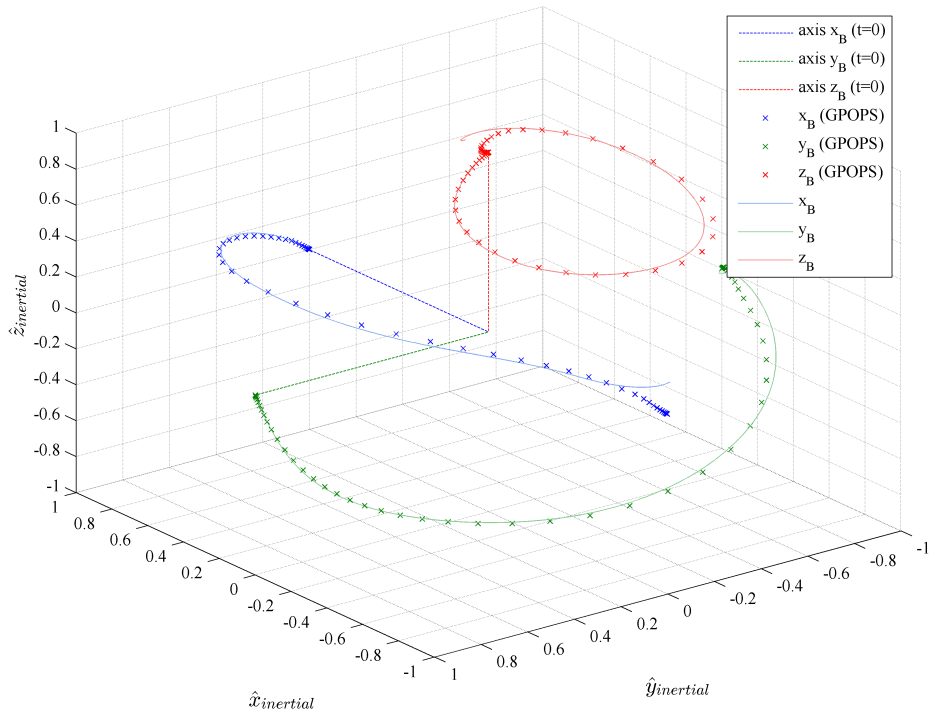


Figure 6.16: 3D Representation: 180° torque-optimal yaw maneuver validation

In Fig. 6.15, we observe the utilization of actuators about all three axes. The maximum angular momentum usage during the maneuver is 29.9%, 46.1%, and 51.2% in the x-, y-, and z-axis respectively. By the end of the maneuver, accumulated momentum usage is 0.35%, 3.32%, and 11.9% about these axes respectively. By utilizing the nutational torque components, the guidance algorithm significantly reduces torque usage about the z-axis, with moderate additional momentum usage in the x- and y-axis. The spacecraft closely follows the torque-optimal guidance waypoints for the majority of the maneuver, as seen in Fig. 6.16. However, as the spacecraft nears the endpoint, the control system is unable to fully counter the gyroscopic torques, resulting in a deviation from the predicted trajectory.

6.4 Computational Cost

The results presented in Sec. 6.3 indicate that the trajectory generated by the torque-optimal guidance algorithm can be closely followed by an open-loop controller in a real-world environment. However, it is necessary to consider the CPU run-time required to generate these trajectories. Also important is the time required to generate the orbital states required by the guidance algorithm using the on-board propagator. Table 6.1 summarizes the algorithm run time on two different processors, averaged over ten executions. Processor specifications were obtained from official Intel documentation[1,2].

					Run Time (sec.)		
Yaw Angle	Nodes	Processor	GFLOP	RAM	*Propagator*	*Guidance*	*Total*
90°	20	i7 2700K	112	16 GB	0.267	5.134	5.401
		Core2Duo T7700	19.2	4 GB	0.690	9.531	10.22
180°	60	i7 2700K	112	16 GB	0.267	63.72	63.99
		Core2Duo T7700	19.2	4 GB	0.690	114.4	115.1

Table 6.1: CPU Run Times for Torque-optimal Trajectory Generation

The on-board propagator requires <1 second to compute the initial and final states required by the guidance algorithm. The CPU time required for on-board propagation is independent of the node count, and scales with maneuver duration - in this case, 100 seconds. From Table 6.1, the guidance routine is clearly the dominant computational load, utilizing 92% - 99.6% of the total run time. Computation times are similar to those presented by Boyarko et al. for time-optimal trajectory generation using

[1]Intel Inc., Intel®Core2Duo™Processor T7000 Series, 2011 (accessed Sept. 12, 2014), http://download.intel.com/support/processors/core2duo/sb/core_T7000.pdf

[2]Intel Inc., Intel®i7™2700 Desktop Processor Series, 2012 (accessed Sept. 19, 2014), http://download.intel.com/support/processors/corei7/sb/core_i7-2700_d.pdf

fifth-order and seventh-order IDVD methods [43], further supporting the validity of the results.

In order to put the run times presented in Table 6.1 into perspective, the computational power available on a Snapdragon 800 System on Chip (SoC), which powers the Google Nexus 5[3], is outlined. Equipped with a 2.26 GHz Quad-core Krait 400 CPU and a Qualcomm® Adreno™ 330 GPU, this SoC is capable of 129.6 GFLOPS. It uses the ARM architecture, in contrast to the x86-64 architecture in modern-day laptops and desktops. ARM does not support hyperthreading, but has the advantage of operating at lower temperatures and with lower power consumption than mainstream processors. Precedence regarding the integration of a smartphone SoC as an auxiliary processing unit on board a CubeSat has been established earlier in Sec. 1.2.

As optimal guidance is a predictive procedure, not real-time, the algorithm would be executed off-line prior to the planned maneuver, and the resulting trajectory data stored until it is required by the controller. Therefore, the guidance algorithm would not continuously utilize valuable computational power which is necessary for scientific observations.

[3]Google Inc., Nexus 5 Technical Specifications. 2014 (accessed Sept. 21, 2014), http://www.google.ca/nexus/5

Chapter 7

Conclusions

The primary research within this thesis focused on the formulation, development, and validation of a low-computation torque-optimal guidance algorithm within the MATLAB-Simulink environment. Simulation results indicate that the torque-optimal guidance algorithm outperforms an Eigenaxis control law in both yaw-pointing accuracy and angular momentum usage for identical maneuvers, with low to moderate computational overhead. Auxiliary research in support of the primary research objective included the development of an accurate, low computational cost atmospheric density model (SPeAD-M86), and the evaluation of drag sails for CubeSat aerostabilization using an in-house dynamics model.

The SPeAD-M86 model proved to be a promising development toward the implementation of precision orbit determination on-board CubeSats. Validation results indicated a reduction in CPU run time of up to 27% while maintaining a daily propagation error of less than 1% compared to an NRLMSISE-00 baseline.

Further validation of the SPeAD-M86 model through examination of passive CubeSat aerostabilization established the use of drag sails on a 3U CubeSat as a viable technique for maintaining a ram-facing orientation. 5 m^2 drag sails maintained a pointing accuracy of 5.7° from the velocity vector over seven days. Furthermore, a

realistic ACS was found to successfully reorient the spacecraft into a low-drag configuration to extend spacecraft orbital life.

7.1 Significance

The significance of this research work is evidenced by the acceptance of publications and presentations in spaceflight and aerospace simulation conferences, as listed below:

Kedare, S. S. and Ulrich, S., Undamped Passive Attitude Stabilization and Orbit Management of a 3U CubeSat with Drag Sails, *25th AAS/AIAA Space Flight Mechanics Meeting*, Williamsburg, VA, 11-15 Jan. 2015, accepted.

Kedare, S. S. and Ulrich, S., Design and Evaluation of a Semi-Empirical Piece-wise Exponential Atmospheric Density Model for CubeSat Applications, *AIAA Modeling and Simulation Technologies Conference*, Kissimmee, FL, 5-9 Jan. 2015, accepted.

Furthermore, the research presented in Chapter 5 is currently being prepared into a conference paper for submission to the 2015 AAS/AIAA Astrodynamics Specialist Conference in Vail, CO.

7.2 Future Work

Future work regarding the SPeAD-M86 atmospheric model in Chapter 3 could include calibration against the MSIS-86 minimum and maximum $F_{10.7}$ density estimates. These additional lookup tables could provide the model with the capability to account for the sinusoidal temporal variation of the solar flux within each solar cycle. An analogous sinusoidal correlation for geomagnetic activity could further improve the accuracy of the model.

Meanwhile, the research on passive drag-sail stability in Chapter 4 opens up a range of potential applications from high-efficiency passive de-orbit modules to science payloads with rapid de-orbit capability. Variable geometry drag-sails could be

utilized to rapidly desaturate or augment momentum storage devices through intentional application of environmental torques. The effects of geometric scaling on aerostabilization could be assessed to determine if similar techniques could be implemented on future resupply vehicles and crewed shuttlecraft.

The torque-optimal guidance algorithm in Chapter 5 presents a significant step towards the development of intelligent GN&C systems for small satellites. For larger spacecraft, the increased size and the presence of structural booms for antennae and instrumentation would necessitate the inclusion of atmospheric and magnetic torques within the guidance routine. Given the modular structure of the algorithm, these additional perturbations could be integrated within the optimization code.

List of References

[1] Bowen, J. A., "On-board orbit determination and 3-axis attitude determination for picosatellite applications," Master's thesis, California Polytechnic State University, San Luis Obispo, California, 2009.

[2] Zimmerman, J. V., *Agenda for IAF Symposium.* International Astronautical Federation, 2007.

[3] Nagatomo, M., Sasaki, S., and Naruo, Y., "Conceptual study of a solar power satellite, SPS 2000," in *Proceedings of the 19th International Symposium on Space Technology and Science (ISTS1994)*, (Annapolis, MD), pp. 469–476, May 1994.

[4] Brown, W. C. and Microwave Power Transmission Systems, "The Equatorial Plane - The International Gateway to Space," in *Space Manufacturing 8 Proceedings of the 10th Princeton/AIAA/SSIConference*, pp. 25–31, May 1991.

[5] Wertz, J. R. and Larson, W. J., *Space Mission Analysis and Design, 2nd ed.* Hawthorne, CA: Microcosm and Springer, 2008.

[6] Proper, I., "Reaction Wheel Design, Construction and Qualification Testing," Master's thesis, York University, Toronto, Canada, 2010.

[7] Jan, Y. W., and Chiou, J. C., "Attitude control system for ROCSAT-3 microsatellite: A conceptual design," *Acta Astronautica*, vol. 56, no. 4, pp. 439–452, 2005.

[8] Candini, G. P., Piergentili, F., and Santoni, F., "Miniaturized attitude control system for Nanosatellites," *Acta Astronautica*, vol. 81, pp. 325–334, 2005.

[9] Ovchinnikov, M., Penkov, V., Norberg, O., and Barabash, S., "Attitude control system for the first Swedish nanosatellite MUNIN," *Acta Astronautica*, vol. 46, no. 2, pp. 319–326, 2000.

[10] Martinelli, M. I., and Peña, R. S. S., "Passive 3 axis attitude control of MSU-1 pico-satellite," *Acta Astronautica*, vol. 56, no. 5, pp. 507–517, 2005.

[11] Grassi, M. and Pastena, M., "Minimum Power Optimum Control of Microsatellite Attitude Dynamics," *Journal of Guidance, Control and Dynamics*, vol. 23, no. 5, pp. 798–804, 2000.

[12] Hughes, P. C., *Spacecraft Attitude Dynamics*. New York: John Wiley and Sons, 1986.

[13] Wie, B., *Space Vehicle Dynamics and Control*. Ohio: AIAA Education Series, 1998.

[14] Platt, D., "A Monopropellant Milli-Newton Thruster System for Attitude Control of Nanosatellites," in *16th Annual AIAA/USU Conference on Small Satellites, SSC02-III-5*, (Logan, Utah), August 2002.

[15] Bedrossian, N., Bhatt, S., Lammers, M., Nguyen, L., and Zhang, Y., "First ever flight demonstration of Zero Propellant Maneuver attitude control concept," in *Proc. 2007 AIAA GN&C Conf.*, (Hilton Head, SC), pp. 1–12, Aug. 20–23 2007.

[16] Bedrossian, N., Bhatt, S., Lammers, M., and Nguyen, L., "Zero Propellant Maneuver flight results for 180° ISS rotation," in *Proc. 2007 Int. Symp. Space Flight Dynamics*, (Annapolis, MD), pp. 1–10, Sept. 24–28 2007. NASA/CP-2007-214158.

[17] Bedrossian, N., Bhatt, S., and Ross, I. M., "Zero propellant maneuver guidance: Rotating the international space station," *IEEE Control Systems Magazine*, vol. 29, no. 5, pp. 53–73, 2009.

[18] Bilimoria, K. D., and Wie, B., "Time-optimal three-axis reorientation of rigid spacecraft," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 3, pp. 446–452, 1993.

[19] Proulx, R., and Ross, I. M., "Time-optimal reorientation of asymmetric rigid bodies," *Advances in the Astronautical Sciences*, vol. 109, pp. 1207–1227, 2001.

[20] Xiaoli, B., and Junkins, J. L., "New results for time-optimal three-axis reorientation of a rigid spacecraft," *Journal of Guidance, Control, and Dynamics*, vol. 32, pp. 1071–1076, July - Aug 2009.

[21] Karpenko, M., Bhatt, S., Bedrossian, N., Fleming, A., and Ross, I. M., "First Flight Results on Time-Optimal Spacecraft Slews," *Journal of Guidance, Control, and Dynamics*, vol. 35, pp. 367–376, March - April 2012.

[22] Coon, T. R., and Irby, J. E., "Skylab attitude control system," *IBM Journal of Research and Development*, vol. 20, no. 1, pp. 58–66, 1976.

[23] Chubb, W. B., and Seltzer, S. M., "Skylab attitude and pointing control system," tech. rep., Feb. 1971. NASA TN D-6068.

[24] Powell, B. K., "Gravity Gradient Desaturation of a Momentum Exchange Attitude Control System," in *AIAA Guidance, Control and Flight Mechanics Conference*, (Hempstead, NY), August 16-18 1971. AIAA Paper 71-940.

[25] Tong, D., "Spacecraft Momentum Dumping Using Gravity Gradient," *Journal of Spacecraft and Rockets*, vol. 35, pp. 714–717, Sept-Oct 1998.

[26] Chamitoff G. E., Dershowitz, A. L., and Bryson, A. L., "Command Level Maneuver Optimization for the International Space Station," *Advances in the Astronautical Sciences*, vol. 104, pp. 311–326, 2000. AAS Paper 00-027.

[27] Bedrossian, N., Metzinger, R., and Adams, N., "Centralized Momentum Management," in *Draper Lab Presentation*, January 31 1996.

[28] Pietz, J. and Bedrossian, N., "Momentum Dumping Using Only CMGs," in *AIAA GN&C Conference*, (Austin, TX), 2003.

[29] Euler, L., "Elementa Calculi variationum," in *Originally published in Novi Commentarii academiae scientiarum Petropolitanae 10*, pp. 51–93, 1766.

[30] Garg, D., Patterson, M. A., Hager, W. W.,Rao, A. V.,Benson, D. A., and Huntington, G. T., "An Overview of Three Pseudospectral Method for the Numerical Solution of Optimal Control Problems," in *Advances in the Astronautical Sciences*, (San Diego), pp. 475–487, Univelt Inc., 2010.

[31] Elnagar, G., Kazemi, M., and Razzaghi, M., "The Pseudospectral Legendre Method for Discretizing Optimal Control Problems," *IEEE Transactions on Automatic Control*, vol. 40, pp. 1793–1796, October 1995.

[32] Fahroo, F. and Ross, I. M., "Costate Estimation by a Legendre Pseudospectral Method," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 2, pp. 270–277, 2001.

[33] Benson, D. A., *A Gauss Pseudospectral Transcription for Optimal Control*. PhD thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, November 2004.

[34] Benson, D. A., Huntington, G. T., Thorvaldsen, T. P., and Rao, A. V., "Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 6, pp. 1435–1440, 2006.

[35] Huntington, G. T., *Advancement and Analysis of a Gauss Pseudospectral Transcription for Optimal Control*. PhD thesis, Department of Aeronautics and Astronautics,Massachusetts Institute of Technology, 2007.

[36] Mathworks Inc., *Optimization Toolbox™User's Guide for R2013b*, September 2013.

[37] Ross, M., *User's Manual For DIDO: A MATLAB^{TM} Application Package for Solving Optimal Control Problems*. Naval Postgraduate School, Monterey, CA, February 2004. Version PR.1.

[38] Holmström, K., Martinsen, F., and Edvall, M., *User's Guide for TOMLAB/-SOCS*. Tomlab Optimization Inc.

[39] Rao, A. V., *User's Manual for GPOPS Version 4.x: A MATLAB(R) Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Pseudospectral Methods*, 2011.

[40] Ross, I. M. and D'Souza, C. N., "A Hybrid Optimal Control Framework for Mission Planning," *Journal of Guidance, Control, and Dynamics*, vol. 28, pp. 686–697, July 2005.

[41] Gill, P. E., Murray, W., and Saunders, M. A., *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*. Department of Mathematics University of California, San Diego, La Jolla, CA.

[42] Gong, Q., Kang, W., Bedrossian, N., Fahroo, F., Sekhavat, P., and Bollino, K., "Pseudospectral Optimal Control for Military and Industrial Applications," in *46th IEEE Conference on Decision and Control*, (New Orleans, LA), pp. 4128–4142, Dec. 2007.

[43] Boyarko, G. A., Romano, M., and Yakimenko, O. A., "Time-Optimal Reorientation of a Spacecraft Using an Inverse Dynamics Optimization Method," *Journal of Guidance, Control, and Dynamics*, vol. 34, pp. 1197–1208, July-August 2011.

[44] Toorian, A., Diaz, K., and Lee, S. , "The CubeSat Approach to Space Access," in *Aerospace Conference, 2008 IEEE*, (Big Sky, MT), March 1-8 2008.

[45] Nader, R., "Ultra Thin, Deployable, Multi-panel Solar Arrays for 1U CubeSats," in *62nd International Astronautical Congress*, (Cape Town, South Africa), October 3-7 2011.

[46] Rawashdeh, S. A., and Lumpp Jr., J. E., "Aerodynamic Stability for CubeSats at ISS Orbit," *Journal of Small Satellites*, vol. 2, no. 1, 2013.

[47] Rawashdeh, S. A., "Passive attitude stabilization for small satellites," Master's thesis, College of Engineering, University of Kentucky, Lexington, KY, 2009.

[48] Montenbruck, O. and Gill, E., *Satellite Orbits: Models, Methods, Applications*. Berlin: Springer-Verlag, 2000.

[49] Gaposchkin, E. M., "Calculation of Satellite Drag Coefficients," Tech. Rep. 998, MIT Lincoln Laboratory, MA, 1994.

[50] Vallado, D. A., and Finkelman, D., "A critical assessment of satellite drag and atmospheric density modeling," *Acta Astronautica*, vol. 92, pp. 141–165, Feb. - March 2014.

[51] Milani, A., Nobili, A. M., and Farinella, P., *Non-gravitational Perturbations and Satellite Geodesy*. Bristol: Adam Higler, 1987.

[52] Schrello, D. M., "Passive Aerodynamic Attitude Stabilization of Near Earth Satellites," tech. rep., North American Aviation Inc., Columbus, OH, 1961.

[53] National Aeronautics and Space Administration, *NASA Space Vehicle Design Criteria (GN&C): Spacecraft Aerodynamic Torques*. NASA SP-8058.

[54] Sidi, M., *Spacecraft Dynamics & Control: A Practical Engineering Approach*. New York: Cambridge University Press, 2002.

[55] Sidi, M., *Spacecraft Dynamics & Control*. New York: Cambridge University Press, 1997.

[56] Champion, K.S.W., Cole, A.E., and Kantor, A.J., *Handbook of Geophysics and the Space Environment: Chapter 14.* Springfield, VA: National Technical Information Service, 1985. Document Accession Number: ADA 167000.

[57] Vallado, D. A. and McClain, W. D., *Fundamentals of Astrodynamics and Applications, 4th ed.* Hawthorne, CA: Microcosm Press, 2013.

[58] Walter, U., *Astronautics: The Physics of Space Flight, 2nd ed.* Weinheim, Germany: Wiley-VCH, 2012.

[59] Jacchia, L. G., "Revised Static Models for the Thermosphere and Exosphere with Empirical Temperature Profiles," in *SAO Special Report No. 332*, (Cambridge, MA), Smithsonian Institution Astrophysical Observatory, 1971.

[60] Roberts Jr., C. E., "An Analytic Model for Upper Atmosphere Densities Based upon Jacchia's 1970 Models," *Celestial Mechanics*, vol. 4, pp. 368–377, December 1971.

[61] COSPAR Working Group IV, *COSPAR International Reference Atmosphere.* Berlin: Akademie-Verlag, 1972.

[62] National Technical Information Service, *U.S. Standard Atmosphere.* Springfield, Virginia, 1976. Product Number: ADA-035-6000.

[63] Bowman, B. R., Tobiska, W. K, Marcos, F., and Huang, C., "The Thermospheric Density Model JB2008 using New EUV Solar and Geomagnetic Indices," in *37th COSPAR Scientific Assembly*, (Annapolis, MD), p. 367, July 13–20 2008. Symposium C, session 42 (oral). Paper number: C42-0004-08.

[64] Hedin, A. E. and Salah, J. E. and Evans, J. V. and Reber, C. A. and Newton, G. P. and Spencer, N. W. and Kayser, D. C. and Alcayde, D. and Bauer, P. and Cogger, L. and McClure J. P., "A Global Thermospheric Model Based on Mass Spectrometer and Incoherent Scatter Data," *Journal of Geophysical Research*, vol. 82, pp. 2139–2156, 1977.

[65] Hedin, A. E., "Extension of the MSIS Thermosphere Model into the Middle and Lower Atmosphere," *Journal of Geophysical Research*, vol. 96, pp. 1159–1172, 1991.

[66] Picone, J. M., Hedin, A. E., Drob, D. P. and Aikin, A. C., "NRLMSISE-00 Empirical Model of the Atmosphere: Statistical Comparisons and Scientific Issues," *Journal of Geophysical Research*, vol. 107, no. A12, 2002.

[67] Akins, A., Healy, L., Coffey, S., and Picone, M. , "Comparison of MSIS and Jacchia Atmospheric Density Models for Orbit Determination and Propagation," in *13th AAS/AIAA Space Flight Mechanics Meeting*, (Ponce, Puerto Rico), February 9-13 2003. Paper AAS 03-165.

[68] Tewari, A., *Atmospheric and Space Flight Dynamics: Modeling and Simulation with MATLAB and Simulink*. Boston: Birkhauser, 2007.

[69] Whitmore, S. A., "Closed-form integrator for the quaternion (euler angle) kinematics equations," May 9 2000. US Patent 6,061,611.

[70] Wall, J. K., "The Feasibility of Aerodynamic Attitude Stabilization of a Satellite Vehicle," *American Rocket Society Preprints*, no. 787, 1959.

[71] Sarychev, V. A., and Ovchinnikov, M. Y., "Dynamics of a Satellite with a Passive Aerodynamic Attitude Control System," *Cosmic Research*, vol. 32, no. 6, pp. 561–575, 1994.

[72] Kumar, R. R., Mazanek, D. D., and Heck, M. L., "Parametric and Classical Resonance in Passive Satellite Aerostabilization," *Journal of Spacecraft and Rockets*, vol. 33, no. 2, pp. 228–234, 1996.

[73] Kumar, R. R., Mazanek, D. D., and Heck, M. L., "Simulation and Shuttle Hitchhiker Validation of Passive Satellite Aerostabilization," *Journal of Spacecraft and Rockets*, vol. 32, no. 5, pp. 806–811, 1995.

[74] Pacini, L., and Skillman, D. A, "A Passive Aerodynamically Stabilized Satellite for Low Earth Orbit," in *Spaceflight Mechanics 1995 : Proceedings of the AAS/ AIAA Spaceflight Mechanics Meeting*, (Albequerque, New Mexico), 1995.

[75] Rawashdeh, S., Jones, D., Erb, D., Karam, A., and Lumpp Jr., J. E., "Aerodynamic Attitude Stabilization for a Ram-Facing CubeSat ," in *AAS 32nd Annual Guidance and Control Conference*, (Breckenridge, Colorado), 2009.

[76] Armstrong, J., Casey, C., Creamer, G., and Dutchover, G., "Pointing Control for Low Altitude Triple Cubesat Space Darts ," in *23rd Annual AIAA/USU Conference on Small Satellites.*, (Logan, UT), 2009.

[77] Rawashdeh, S. A. and Lumpp Jr., J. E., "CubeSat Aerodynamic Stability at ISS Altitude and Inclination," in *26th Annual AIAA/USU Conference on Small Satellites.*, (Logan, UT), 2012. SSC12-VIII-6.

[78] Bonin, G., Hiemstra, J., Sears, T., and Zee, R. E., "The CanX-7 Drag Sail Demonstration Mission: Enabling Environmental Stewardship for Nano- and Microsatellites," in *27th Annual AIAA/USU Conference on Small Satellites.*, (Logan, UT), 2013. SSC13-XI-9.

[79] Lappas, V., Pelleggrino, S., Guenat, H., Straubel, M., Steyn, H., Kostopoulos, V., Sarris, E., Takinalp, O., Wokes, S., and Bonnema, A., "DeOrbitSail: De-Orbiting of Satellites using Solar Sails," in *2nd International Conference on Space Technology (ICST)*, (Athens, Greece), 15-17 Sept. 2011.

[80] Shi, J.-F., Ulrich, S., and A. Allen, "Spacecraft Adaptive Attitude Control with Application to Space Station Free-Flyer Robotic Capture," in *AIAA Guidance, Navigation, and Control Conference*, (Kissimmee, FL), 5-9 January 2015. accepted.

[81] Fahroo, F. and Ross, I. M., "Direct trajectory optimization by a chebyshev pseudospectral method," *Journal of Guidance, Control, and Dynamics*, vol. 25, pp. 160–166, January-February 2002.

[82] Kawajir, Y., Laird, C., Vigerske, S., and Wächter, A., *Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT*. Eclipse Public License, August 2014. Revision: 2500.

[83] Waltz, R. A. and Plantenga, T. D., *KNITRO User's Manual Version 7.0*. Ziena Optimization, Inc., September 2010.

[84] Murtagh, B. A. and Saunders, M. A., *MINOS 5.51 User's Guide*. Systems Optimization Laboratory, Stanford University, Stanford CA, September 2003.

[85] Gill, P. E., Murray, W., and Saunders, M. A., *User Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*, June 2008.

[86] Yaple, D. E., "Simulation and Application of GPOPS for a Trajectory Optimizaton and Mission Planning Tool," Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, March 2010. AFIT/GAE/ENY/10-M29.

[87] Fleming, A., "Real-time Optimal Slew Maneuver Design and Control," Master's thesis, U.S. Naval Postgraduate School, Monterey, CA, 2004.

[88] Yakimenko, O., "Direct method for rapid prototyping of near optimal aircraft trajectories," *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 5, pp. 865–875, 2000.

[89] Junfeng, L., Min, X., Zhaolin, W., and Shimin, W., "Minimum-torque Earth Off-nadir Pointing Control of Gravity Gradient Stabilized Small Satellites," *Tsinghua Science and Tenchnology*, vol. 5, pp. 31–33, March 2000.

[90] Bender, E., *An Analysis of Stabilizing 3U CubeSats Using Gravity Gradient Techniques and a Low Power Reaction Wheel*. California Polytechnic State University, San Luis Obispo, CA, 2011.

[91] Bate, R. R., Mueller, D. D., White, J. E., *Fundamentals of Astrodynamics*. New York: Dover Publications, Inc., 1971.

Appendix A

SPeAD-M86 Supporting Material

Altitude Interval, km	Scale Height, km	CIRA72 Base Density, kg·m^{-3}
0 - 25	7.249	1.225E+00
25 - 30	6.349	3.899E-02
30 - 40	6.682	1.774E-02
40 - 50	7.554	3.972E-03
50 - 60	8.382	1.057E-03
60 - 70	7.714	3.206E-03
70 - 80	6.549	8.770E-05
80 - 90	5.799	1.905E-05
90 - 100	5.382	3.396E-06
100 - 110	5.877	5.297E-07
110 - 120	7.263	9.661E-08
120 - 130	9.473	2.438E-08
130 - 140	12.636	8.484E-09
140 - 150	16.149	3.845E-09
150 - 180	22.523	2.070E-09
180 - 200	29.74	5.464E-10
200 - 250	37.105	2.789E-10
250 - 300	45.546	7.248E-11
300 - 350	53.628	2.418E-11
350 - 400	53.298	9.518E-12
400 - 450	58.515	3.725E-12
450 - 500	60.828	1.585E-12
500 - 600	63.822	6.967E-13
600 - 700	71.835	1.454E-13
700 - 800	88.667	3.614E-14
800 - 900	124.64	1.170E-14
900 - 1000	181.05	5.245E-15

Table A.1: CIRA72 Altitude Interval Data [57]

Altitude Interval, km	Scale Height, km	MSIS-86 Base Density, kg/m^3	Scale Density, kg/m^3
0 - 100	6.7	1.225	1.225
100 - 150	9.5	4.79E-07	1.30E-02
150 - 200	25.5	1.81E-09	5.70E-07
200 - 250	37.5	2.53E-10	4.80E-08
250 - 300	44.8	6.24E-11	1.60E-08
300 - 350	50.3	1.95E-11	7.40E-09
350 - 400	54.8	6.98E-12	4.00E-09
400 - 450	58.2	2.72E-12	2.60E-09
450 - 500	61.3	1.13E-12	1.70E-09
500 - 550	64.5	4.89E-13	1.10E-09
550 - 600	68.7	2.21E-13	6.30E-10
600 - 650	74.8	1.04E-13	3.10E-10
650 - 700	84.4	5.15E-14	1.10E-10
700 - 750	99.3	2.72E-14	3.00E-11
750 - 800	121	1.55E-14	7.10E-12
800 - 850	151	9.63E-15	1.90E-12
850 - 900	188	6.47E-15	5.90E-13
900 - 950	226	4.66E-15	2.40E-13
950 - 1000	263	3.54E-15	1.30E-13

Table A.2: MSIS-86 Altitude Interval Data

Listing A.1: True anomaly error computation

```
% Initialize v_error
v_error = [];
for i = 1:length(time)
% Calculate v_error from COE arrays, assign to temporaty variable
v_error_temp =  COE_SP(i,6) - COE_NRL(i,6);
% Process
    if  v_error_temp > pi
        v_error_temp =  2*pi - (COE_SP(i,6) - COE_NRL(i,6));
    elseif v_error_temp < - pi
        v_error_temp =  2*pi + (COE_SP(i,6) - COE_NRL(i,6));
    else
    end
% Concatenate error to v_error array
v_error = [v_error (100/(2*pi))*v_error_temp];
end
```

Appendix B

CubeSat DS-1 Supporting Material

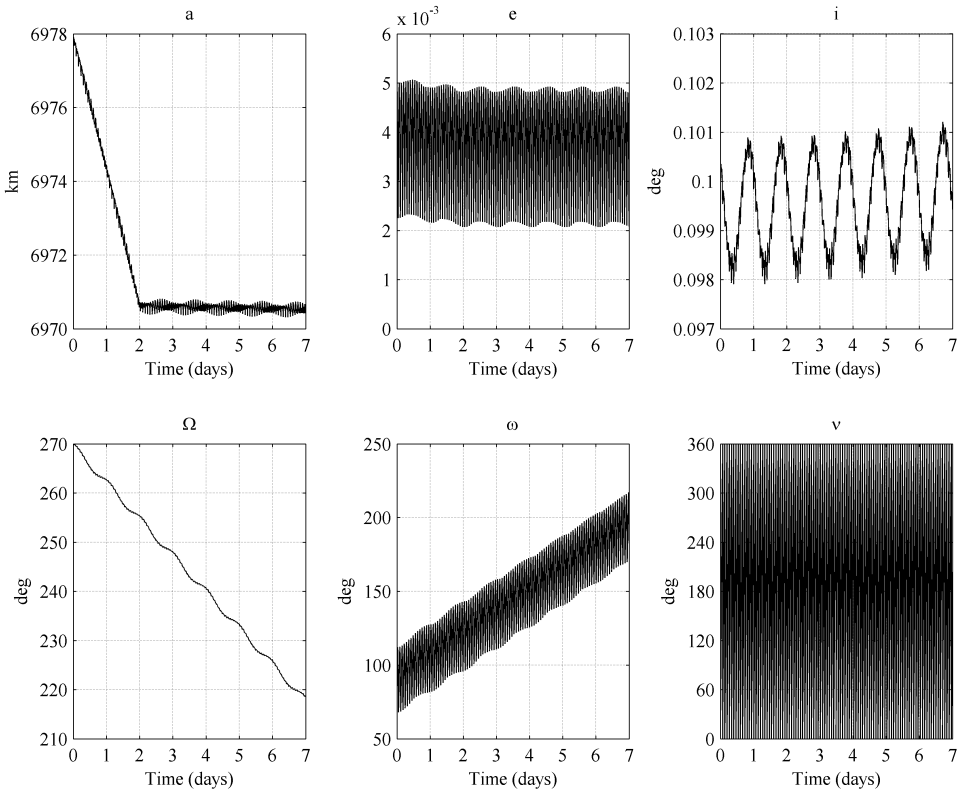


Figure B.1: Orbital Elements for OLE Mode

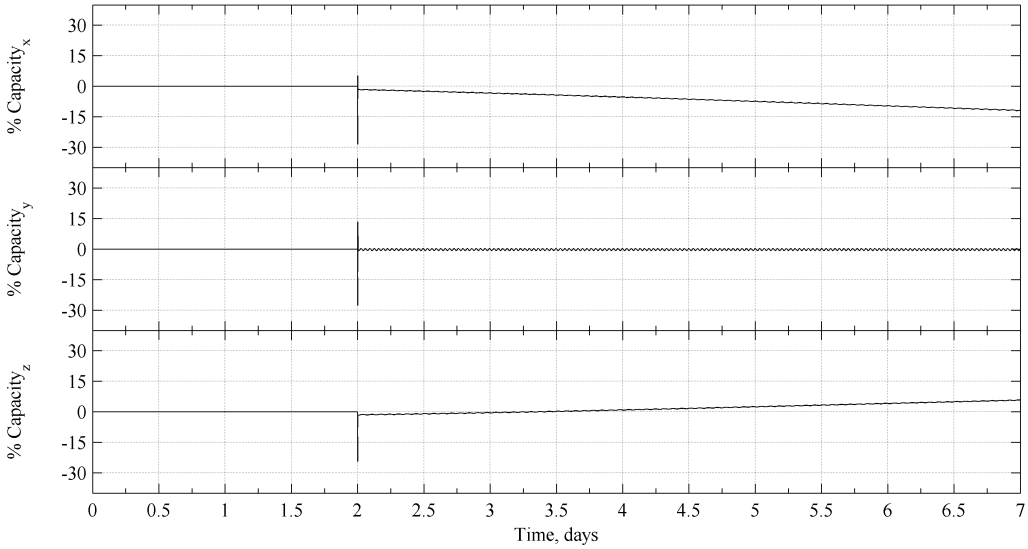


Figure B.2: 7 Day Angular Momentum Usage in OLE Mode

Appendix C

Torque-optimal Guidance Supporting Material

Listing C.1: Differential equations of motion

```
% Function AttitudeDae defines the equations of motion
function dae = AttitudeDae(sol,iphase);

% Call states (refer GPOPS documentation)
t = sol{1};
x = sol{2};
u = sol{3};
p = sol{4};

global Ixx Iyy Izz mu % Call global variables

% Normalize quaternion states prior to propagation
[x(:,4:7)] = quatnormalize(real([x(:,4) x(:,5) x(:,6) x(:,7)]));

r = sqrt(x(:,8).^2+x(:,9).^2+x(:,10).^2); % Calculate r

% Compute gravity gradient torque
T_gg_1 = 12*mu./(r.^3).*(Izz-Iyy).*(x(:,5).*x(:,4) - ...
         x(:,7).*x(:,6)).*(x(:,6).*x(:,4) + x(:,7).*x(:,5));
T_gg_2 = 12*mu./(r.^3)*(Ixx-Izz).*(x(:,6).*x(:,4) + ...
         x(:,7).*x(:,5)).*(x(:,7).^2 + x(:,4).^2-0.5);
T_gg_3 = 12*mu./(r.^3)*(Iyy-Ixx).*(x(:,7).^2 + ...
         x(:,4).^2-0.5).*(x(:,5).*x(:,4) - x(:,7).*x(:,6));

% Define attitude EoM
wx_dot = ((Iyy-Izz).*x(:,2).*x(:,3) + u(:,1) + T_gg_1)/Ixx;
wy_dot = ((Izz-Ixx).*x(:,1).*x(:,3) + u(:,2) + T_gg_2)/Iyy;
wz_dot = ((Ixx-Iyy).*x(:,2).*x(:,1) + u(:,3) + T_gg_3)/Izz;

q1_dot = 0.5*(x(:,3).*x(:,5) - x(:,2).*x(:,6) + x(:,1).*x(:,7));
```

```
q2_dot = 0.5*(-x(:,3).*x(:,4) + x(:,1).*x(:,6) + x(:,2).*x(:,7));
q3_dot = 0.5*(x(:,2).*x(:,4) - x(:,1).*x(:,5) + x(:,3).*x(:,7));
q4_dot = 0.5*(-x(:,1).*x(:,4) - x(:,2).*x(:,5) - x(:,3).*x(:,6));

% Define orbital state EoM
rx_dot = x(:,11);
ry_dot = x(:,12);
rz_dot = x(:,13);

vx_dot = -mu.*x(:,8)./r.^3;
vy_dot = -mu.*x(:,9)./r.^3;
vz_dot = -mu.*x(:,10)./r.^3;

% Assemble EoMs
dae = [wx_dot wy_dot wz_dot q1_dot q2_dot q3_dot q4_dot ...
       rx_dot ry_dot rz_dot vx_dot vy_dot vz_dot];
```

Listing C.2: Cost function

```
% Function brysonDenhamCost defines the cost function
function [Mayer,Lagrange]=AttitudeCost(sol,iphase);

% Call states (refer GPOPS manual)
t0 = sol{1,1};
x0 = sol{1,2};
tf = sol{1,3};
xf = sol{1,4};
t  = sol{2,1};
x  = sol{2,2};
u  = sol{2,3};
p  = sol{2,4};

% Define Mayer cost function - uncomment required cost function
Mayer = zeros(size(t0)); % No cost
% Mayer = tf; % Maneuver time

% Define Lagrange cost function
% Lagrange = zeros(length(t),1); % No cost
% Lagrange = ones(length(t),1); % Maneuver time (alternate)
Lagrange = sqrt((u(:,1).^2+u(:,2).^2+u(:,3).^2)); % Torque magnitude
% Lagrange = sqrt((1/3)*((u(:,1).^2+u(:,2).^2+u(:,3).^2))); % RMS torque
```

Listing C.3: Main function (180° yaw maneuver)

```
% This algorithm calculates the optimal guidance path
tic; % Start clock
clear all; clc; % Clear workspace and command window

addpath('SNOPT7') % Load SNOPT 7.2 NLP solver
```

```
gpopsInitialize % Initialize GPOPS variables

% Define initial states
w0 = [0;0;0]; % rad/s
q0 = [0;0;0;1];
r0 = [6744.09972813567;-1.65182865800867e-12;11.7706864857384]; %km
v0 = [1.88768722816998e-15;7.70707092143781;8.23658829943211e-19]; %km/s
x0 = [w0;q0;r0;v0];

% Define final states
wf = [0;0;0]; %rad/s
qf = [0;0;sind(180/2);cosd(180/2)];
rf = [6700.26562390112;769.036605065181;11.6939597859881]; %km
vf = [-0.875721091725920;7.65697885496423;-0.00153284702143234]; %km/s
xf = [wf;qf;rf;vf];

% Define state limits
xmin = [-0.3;-0.3;-0.3;-1;-1;-1;-1;-1E+4;-1E+4;-1E+4;-10;-10;-10];
xmax = [0.3;0.3;0.3;1;1;1;1;1E+4;1E+4;1E+4;10;10;10];

% Define control limits
umin = [-0.0001;-0.0001;-0.0001]; %Nm
umax = [0.0001;0.0001;0.0001]; %Nm

% Define global variables
global Ixx Iyy Izz mu
Ixx = 0.059; % kg*m^2
Iyy = 0.114; % kg*m^2
Izz = 0.126; % kg*m^2
mu = 398600; % km^3/s^2

% Define limits for the NLP (refer GPOPS manual)
iphase = 1;
limits{imin}{iphase,itime} = [0 100];
limits{imax}{iphase,itime} = [0 100];
limits{imin}{iphase,istate} = [x0 xmin xf];
limits{imax}{iphase,istate} = [x0 xmax xf];
limits{imin}{iphase,icontrol} = umin;
limits{imax}{iphase,icontrol} = umax;
limits{imin}{iphase,iparameter} = [];
limits{imax}{iphase,iparameter} = [];
limits{imin}{iphase,ipath} = [];
limits{imax}{iphase,ipath} = [];
limits{imin}{iphase,ievent} = [];
limits{imax}{iphase,ievent} = [];

% Provide initial guesses to the NLP solver (refer GPOPS manual)
solinit{iphase,itime} = [0; 2; 4; 6; 8];
solinit{iphase,istate}(:,1) = [0.0; 0.03; 0.05; 0.07; 0.09];
solinit{iphase,istate}(:,2) = [0.0; 0.03; 0.05; 0.07; 0.09];
solinit{iphase,istate}(:,3) = [0.0; 0.03; 0.05; 0.07; 0.09];
```

```
solinit{iphase,istate}(:,4) = [0;0;0;0;0];
solinit{iphase,istate}(:,5) = [0;0;0;0;0];
solinit{iphase,istate}(:,6) = [0;0.1;0.1;0.1;0.1];
solinit{iphase,istate}(:,7) = [1;0.9;0.9;0.9;0.9];
solinit{iphase,istate}(:,8) = [6745; 6745; 6745; 6745; 6745];
solinit{iphase,istate}(:,9) = [0; 0; 0; 0; 0];
solinit{iphase,istate}(:,10) = [12; 12; 12; 12; 12];
solinit{iphase,istate}(:,11) = [0;0;0;0;0];
solinit{iphase,istate}(:,12) = [7.7;7.7;7.7;7.7;7.7];
solinit{iphase,istate}(:,13) = [0;0;0;0;0];
solinit{iphase,iparameter} = [];

% Provide initial guess of control variables (refer GPOPS manual)
solinit{iphase,icontrol}(:,1) = [0;0;0;0;0];
solinit{iphase,icontrol}(:,2) = [0;0;0;0;0];
solinit{iphase,icontrol}(:,3) = [0;0;0;0;0];

% Define parameters required by GPOPS (refer GPOPS manual)
setup.name = 'TorqueOptimal_180Degree';
setup.funcs = {'AttitudeCost','AttitudeDae'};
setup.nodes = 60;
setup.limits = limits;
setup.solinit = solinit;
setup.connections = [];
setup.derivatives = 'complex';
setup.direction = 'increasing';
setup.autoscale = 'off';

% Solve
output = gpops(setup);
solution = output.solution;

solution{1,2}(:,4:7) = quatnormalize(real(solution{1,2}(:,4:7)));
toc; % Stop clock
```

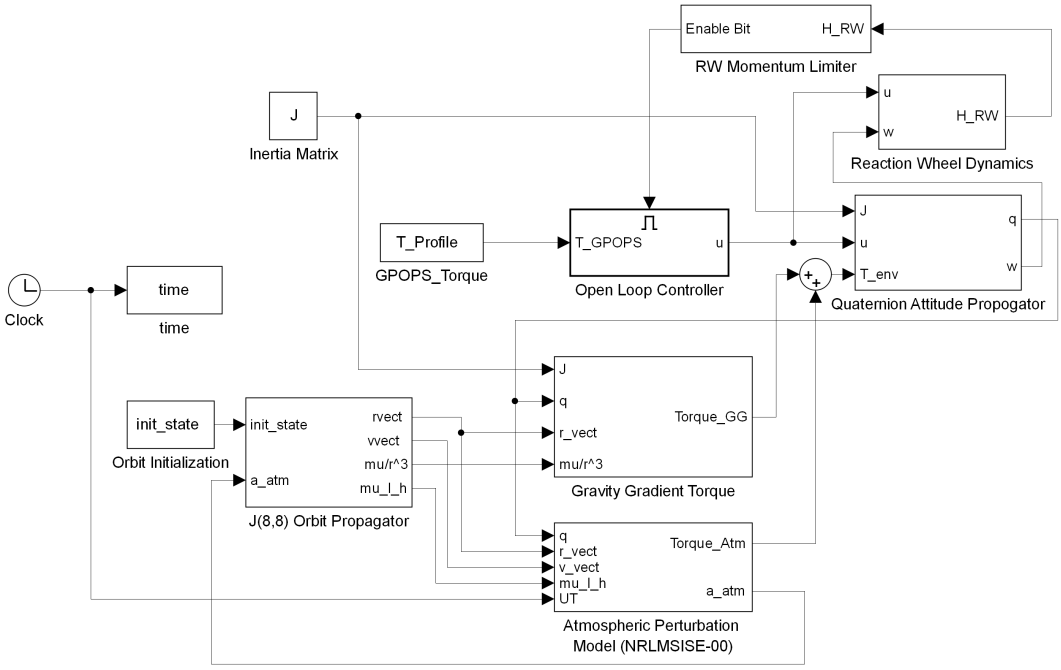


Figure C.1: Top-level Simulink Block Diagram of Real-world Propagator for Optimal Trajectory Validation

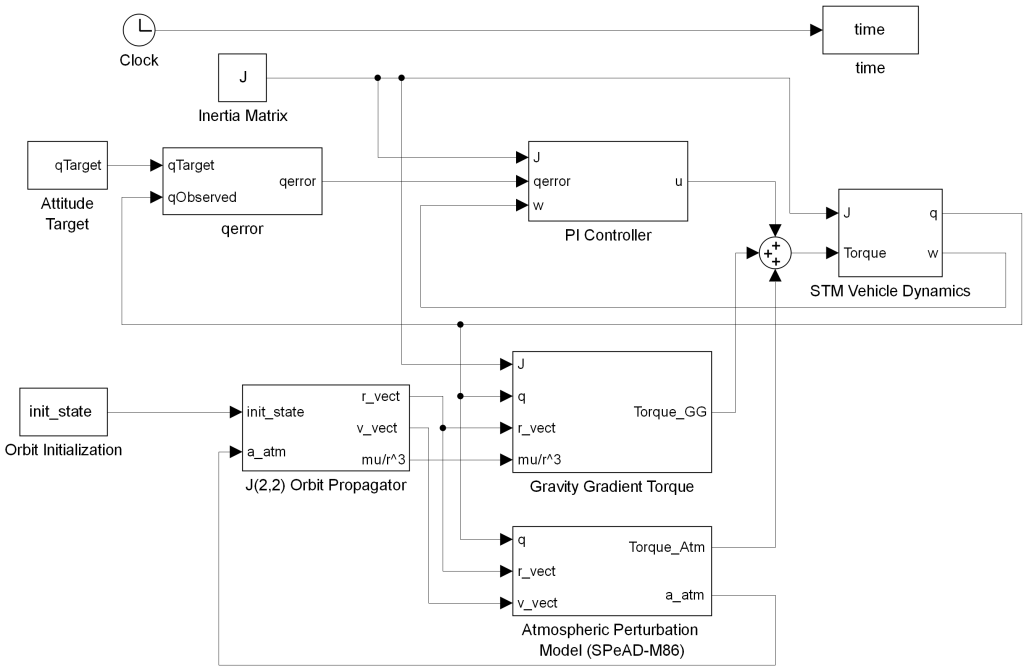


Figure C.2: Top-level Simulink Block Diagram of Onboard Propagator

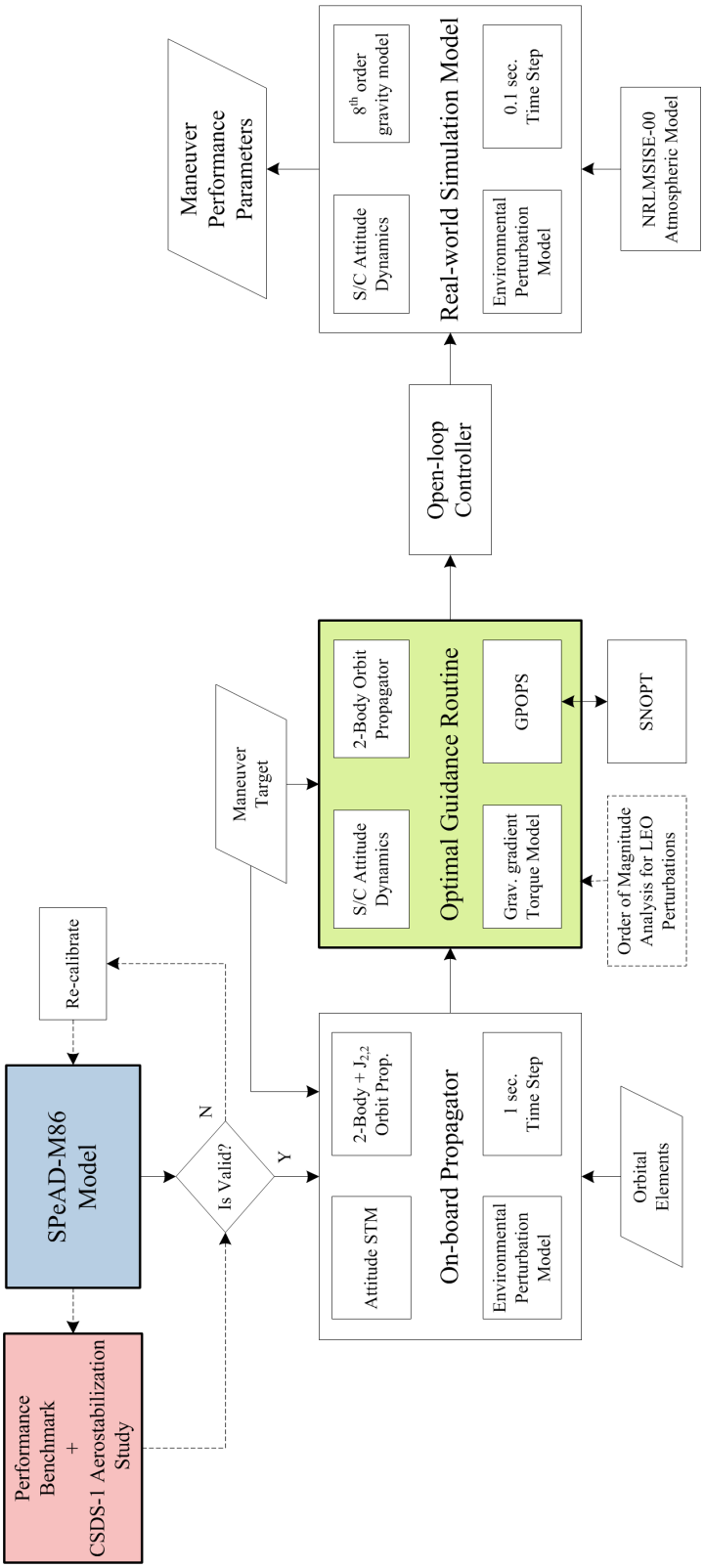


Figure C.3: Thesis Component Flowchart